Spring 4-15-2014

# Improving Recurrent Software Development: A Contextualist Inquiry Into Release Cycle Management

Syed M. Kamran
*Georgia State University*

**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, to copy from, or 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 the 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.

*Kamran M. Syed*

**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:

*Kamran M. Syed*
Georgia State University
Robinson College of Business
35 Broad Street NW
Atlanta, GA 30303

The director of this dissertation is:

*Dr. Lars Mathiassen*
Atlanta, GA 30326
Georgia State University
3344 Peachtree Road,
Atlanta, GA 30326

*Improving Recurrent Software Development: A Contextualist Inquiry Into Release Cycle Management*

BY

*KAMRAN M. SYED*

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree

Of

Executive Doctorate in Business

In the Robinson College of Business

Of

Georgia State University

GEORGIA STATE UNIVERSITY
ROBINSON COLLEGE OF BUSINESS
2014

**ACCEPTANCE**

This dissertation was prepared under the direction of the *Kamran M. Syed* 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 Doctoral of Philosophy in Business Administration in the J. Mack Robinson College of Business of Georgia State University.

H. Fenwick Huss, Dean

DISSERTATION COMMITTEE

Dr. Lars Mathiassen (Chair)

Dr. Wesley Johnston

Dr. Balasubramaniam Ramesh

# TABLE OF CONTENTS

CHAPTER I: INTRODUCTION....1
CHAPTER II: LITERATURE REVIEW....5
II.I SOFTWARE RELEASE LITERATURE....5
II.II SOFTWARE PROCESS IMPROVEMENT LITERATURE....6
CHAPTER III: ANALYTICAL FRAMEWORK....9
CHAPTER IV: RESEARCH METHODOLOGY....14
IV.I PROBLEM-SOLVING CYCLE....16
IV.II RESEARCH CYCLE....38
IV.III DATA COLLECTION....38
IV.IV DATA ANALYSIS....39
CHAPTER V: RESULTS....45
V.I DIAGNOSTIC PHASE....45
V.II ESTABLISHMENT PHASE....52
V.III ACTING PHASE....57
V.IV LEARNING PHASE....67
CHAPTER VI: DISCUSSION....72
VI.I IMPROVING RELEASE CYCLE MANAGEMENT AT SOFTWARE INC.....72
VI.II RELEASE CYCLE MANAGEMENT IN RECURRENT SOFTWARE DEVELOPMENT....76
VI.III A GROUNDED MODEL OF RELEASE CYCLE MANAGEMENT....81
CHAPTER VII: CONCLUSION....90
APPENDIX A: SHARED PLATFORM DOCUMENT....95
APPENDIX B: *SECURE-ON-REQUEST* RELEASE MANAGEMENT IMPROVEMENT PROJECTS – STATUS REPORT....146
APPENDIX C: CUSTOMER ADVISORY BOARD - MEETING ITEMS....148

APPENDIX D: OCTOBER 2013 RELEASE *SECURE-ON-REQUEST* RELEASE CHECKLIST.....150
APPENDIX E: DATA ANALYSIS SCHEMA (AS SEEN IN NVIVO).....151
APPENDIX F: *SECURE-ON-REQUEST* NEW RELEASE CYCLE MODEL.....152
APPENDIX G: *SECURE-ON-REQUEST* RELEASE MANAGEMENT ASSESSMENT AND IMPROVEMENT OPTIONS.....153
REFERENCES.....160

# LIST OF TABLES

Table 1: Research Design Summary (Mathiassen et al., 2012) ..... 4
Table 2: Key Analytical Constructs ..... 11
Table 3: Analytical Framework ..... 13
Table 4: Problem Solving Timeline ..... 17
Table 5: Release Management Practice-Based Assessment (Pre-intervention) ..... 20
Table 6: Improvement Projects Schedule ..... 22
Table 7: Improvement of *Secure-on-Request* Customer Relationship Project ..... 23
Table 8: Conceptual Definitions of the Six SaaS-Qual Factors (Benlian, et al., 2011; Barqawi, 2014) ..... 26
Table 9: Improvement of *Secure-on-Request* Requirements And Quality ..... 27
Table 10: Improvement of *Secure-on-Request* Release Cycle Project ..... 30
Table 11: Reoccurring Meetings for the New *Secure-on-Request* Release Model ..... 37
Table 12: Release Management Practice-Based Assessment (Post-intervention) ..... 37
Table 13: Data Sources ..... 39
Table 14: A Grounded Model of Release Cycle Management (RCM) in Recurrent Software Development ..... 87

# LIST OF FIGURES

*Number* *Page*

Figure 1: Contextualist Inquiry Diagram (Pettigrew, 1987) ..... 10
Figure 2: IDEAL Model (McFeeley, 1996) ..... 16
Figure 3: *Secure-on-Request* Service Blueprint at Software Inc. ..... 21
Figure 4: The New Secure-on-Request Release Model ..... 36
Figure 5: Data Analysis Activities ..... 43
Figure 6: *Secure-on-Request* new and fixed defects trends ..... 68
Figure 7: Conceptual Representation of Release Cycle Management (RCM) in Recurrent Software Development ..... 81
Figure 8: A Grounded Model of Release Cycle Management (RCM) in Recurrent Software Development ..... 88

# ABBREVIATIONS AND DEFINITIONS

CAB - Customer Advisory Board

CAR - Canonical Action Research

CPR - Collaborative Practice Research

GSU – Georgia State University

QA - Quality Assurance

RCM - Release Cycle Management

SPI - Software Process Improvement

TAM - Technical Account Manager

# ABSTRACT

*IMPROVING RECURRENT SOFTWARE DEVELOPMENT:*
*A CONTEXTUALIST INQUIRY INTO RELEASE CYCLE MANAGEMENT*

BY

*KAMRAN M. SYED*

*May 8th, 2014*

Committee Chair: *Dr. Lars Mathiassen*

Major Academic Unit: *J. Mack Robinson College of Business*

Software development is increasingly conducted in a recurrent fashion, where the same product or service is continuously being developed for the marketplace. Still, we lack detailed studies about this particular context of software development. Against this backdrop, this dissertation presents an action research study into Software Inc., a large multi-national software provider. The research addressed the challenges the company faced in managing releases and organizing software process improvement (SPI) to help recurrently develop and deliver a specific product, *Secure-on-Request*, to its customers and the wider marketplace. The initial problem situation was characterized by recent acquisition of additional software, complexity of service delivery, new engineering and product management teams, and low software development process maturity. Asking how release management can be organized and improved in the context of recurrent development of software, we draw on Pettigrew's contextualist inquiry to focus on the ongoing interaction between the contents, context and process to organize and improve release cycle practices and outcomes. As a result, the dissertation offers two contributions. Practically, it contributes to the resolution of the problem situation at Software Inc. Theoretically, it introduces a new software engineering discipline, release cycle management (RCM), focused on recurrent delivery of software, including SPI as an integral part, and grounded in the specific experiences at Software Inc.

# INTRODUCTION

The costs and time to create customized business systems software are often prohibitive (Carmel & Becker, 1995; Ncube, Oberndorf, Kark, 2008; Sawyer, 2000; Xu & Brinkkemper, 2007). As a result, the need for packaged business software has grown in recent years (Colomo-Palacios, Soto-Acosta, García-Peñalvo & García-Crespo, 2012). The common business model of the producers of software packages is to make one and then sell many copies (Xu & Brinkkemper, 2007). However, academic literature often lacks clarity in differentiating between software types, such as commercial off-the-shelf software (COTS), shrink-wrapped software or commercial software (Xu & Brinkkemper, 2007). By not fully exploring the deeper implications that emerge from considering that not all software development is the same, there remain gaps in the research. For example, an important area of packaged software that is not emphasized in literature is the recurrent nature of its development. Through using the term 'recurrent' we mean that the software is incrementally updated with improvements or new features, so new versions of the software can be released into the marketplace, ideally to fulfill or exceed the evolving consumer requirements.

Hence, studies into the recurrent development of software have the potential to explore new ground by exhibiting the unique aspects of these development processes and examining ways to improve them. That is the approach taken in the dissertation. Specifically, the dissertation examines how the recurrent development of software is managed and how the processes can be improved. The study draws on Xu and Brinkkemper's (2007) definition of packaged software as a ready-to-use product that is available to buy off-the-shelf from vendors,

and requires little in the way of modification. The definition is often used in talking about upscale enterprise software suites, such as customer relationship management (CRM) systems or enterprise resource planning (ERP).

Release management has been increasingly studied, within the software literature, mostly narrowly focusing on release management as separate activities, but also at times focusing holistically at the entire set of activities involved. Still, there are no studies that focus specifically on release management in the context of recurrent development of software. Similarly, software process improvement (SPI) has been studied extensively to drive improvements in software practices. There are a variety of SPI approaches available, mostly focused on process improvements as separate activities that support software development through interventions over time. There are a few studies of SPI as an emergent, integrated activity, but we found no studies focused on SPI in the specific context of recurrent software development.

Against this backdrop, we conducted a collaborative action research study with Software Inc., a large multi-national software provider. The study adopts two complementary perspectives, one grounded in SPI and engineering practices, and the other grounded in service delivery and customer interactions. This overall research design is described in detail in the Shared Dissertation Platform Document, Appendix A. Drawing on these complementary perspectives, through action research, we addressed the challenges the company faced in managing releases and in organizing SPI to improve the recurrent development and delivery of a specific product, *Secure-on-Request*. To factor in considerations to the specific environment at Software Inc. and to emphasize the particular characteristics of recurrent software development, we adopted Pettigrew's contextualist inquiry (Pettigrew, 1987 & 1990) as analytical lens. This

theoretical framework has previously been used to support action research into software practices (Frederiksen & Mathiassen, 2008; Napier et al., 2011), and it helped us organize a systematic inquiry into the context, content, and process involved in transforming the release management and process improvement at Software Inc.

On the basis of the above, the research focuses on the following research question: How can release management be organized and improved in the context of the recurrent development of software? This dissertation offers two contributions. Practically, it contributes to the resolution of the problem situation at Software Inc. Theoretically, it introduces a new release paradigm, release cycle management (RCM), focused on the recurrent delivery of software, including SPI as an integral part, and, grounded in the specific experiences at Software Inc. This action research, therefore, adds to the body of knowledge the concept of RCM which will be elaborated upon and precisely defined during the study. The empirical insights gained from our problem diagnosis, interventions and learning from Software Inc., are helpful to both practitioners and academic researchers. Overall, this dissertation relies on the style composition for action research (Mathiassen et al., 2012) summarized in Table 1. The different elements of this design will be dissected, described and further elaborated upon in the subsequent sections of the dissertation.

**Table 1: Research Design Summary (Mathiassen et al., 2012)**

| | |
|---|---|
| P (Problem setting) | Improve Software Inc.'s release practices |
| A (Area of concern) | Improving release management cycle in recurrent development of software |
| RQ (Research Question) | How can you organize and improve release management in the context of recurrent development of software? |
| F (Conceptual Framework) | Fi: Pettigrew's framework (1987 & 1990) for studying organizational change - emphasizing content, the context, and the process.<br>Fa: Models of recurrent development of software and IDEAL model (McFeeley, 1996) |
| M (Research Method) | Qualitative, action research study |
| CA (Contribution to A) | 1. Improved release management at Software Inc.<br>2. Empirical contribution to improving RCM in recurrent development of software<br>3. A grounded model of RCM in recurrent development of software |

# LITERATURE REVIEW

This chapter provides a review of two major streams of scholarly literature. First, the field of software release is reviewed, after which, the vast body of knowledge on SPI is examined. In the conclusion of this chapter, the research opportunity is presented.

## II.I Software Release Literature

The software release literature introduces a number of related practices. The literature recognizes specific release related activities, like software release management, which covers identifying, collecting, packaging, and distributing the components of a software item, such as executable programs, documentation, release notes, and configuration information (Ballintijn, 2005; Scott & Nisse, 2001). Van Der Hoek, Hall, Heimbigner, & Wolf (1997) defines software release management as: “The process through which software is made available to and obtained by the user.” Similarly, the literature covers release planning as the activity of deciding how to assign releasable product characteristics, such as features and requirements, to a planned sequence of releases of an evolving software product (Carlshamre, 2002; Regnell & Kuchcinski, 2011; Ruhe & Saliu, 2005; Svahnberg et al., 2010). The literature also highlights a number of approaches to release time estimation (Gaur & Oberoi, 2012). Related to release estimation, a number of researchers have attempted to conceptualize software prediction mathematical models to forecast the software release time (Qian, Yao & Khoshgoftaar, 2010). There are also studies focused on the technical aspects of release build and configuration management (Mazlan, Sefat, Selan & Lukose, 2013).

The specific release activities, like release management, release planning, release estimation, release build and configuration management have been well studied. Furthermore, there is an emerging literature that takes a broader, holistic view on software releases. For example, Taborda establishes an end-to-end release framework which ensures initiatives are planned and prioritized to streamline IT project portfolio execution and delivery in an enterprise management context (Taborda, 2012). Similarly, Humble and Farley lay out a detailed, holistic concept of release pipelines, in their study on improving release management (Humble & Farley, 2010). However, their study is only focused on the technical aspects of software release. While both these studies are focused on software releases from a holistic perspective, their context is different from recurrent development of software for the market. Therefore, the traditional software release literature lacks a unified concept of release that presents how all the moving parts fit together, including requirements management, development, testing, documentation, user acceptance and delivery in recurrent software development.

One of the goals of this dissertation is to address this gap in the literature by developing a holistic perspective of these different viewpoints about release, and assembling a multifaceted understanding of a recurrent software release, from the point it is first identified and defined as part of strategic planning, to its ultimate realization as a solution delivering additional benefits.

### II.II Software Process Improvement Literature

The SPI literature covers a wide variety of approaches and practices aimed at improving quality and reliability, employee and customer satisfaction, and return on investment in software development (Muller et al., 2010). SPI has been adopted by many organizations as a strategy to

enhance their capability to deliver quality software (Grady, 1997; Humphrey, 1989; Mathiassen et al., 2002). Although very successful cases have been reported (Diaz & Sligo, 1997; Haley, 1996; Humphrey et al., 1991; Larsen & Kautz, 1996), there is a critical debate about the approach (Bach, 1995; Bollinger & McGowan, 1991; Fayad & Laitinen, 1997; Humphrey & Curtis, 1991) and the feasibility and practicability of SPI initiatives (Bach 1995; Bollinger & McGowan 1991; Brodman & Johnson 1995; Curtis 1994; Fayad & Laitinen 1997; Herbsleb et al. 1997; Humphrey et al. 1991; Ngwenyama & Nielsen 2003).

SPI projects usually rely on well-known models of software process maturity, such as the Software Engineering Institute's Capability Maturity Model (CMM) (Paulk et al., 1993) and Bootstrap (Kuvaja et al., 1994). Critics claim that the models offer an overly rigid and limited view of software production and overlook the variety and complexities of software producing organizations (Bollinger & McGowan, 1991; Kohoutek, 1996; Mathiassen & Sorensen, 1996; Pries-Heje & Baskerville, 1999; Velden et al., 1996; Allison & Merali, 2007). Therefore, there is a need to investigate alternative or complementary approaches in the SPI field.

An increasing volume of research proposes advice to achieve SPI success. McFeeley (1996) discusses how to effectively organize learning cycles through the IDEAL model (i.e. Initiate, Diagnose, Establish, Act, Learn) for SPI. Mashiko and Basili (1997) and Ravichandran (2000) examine how SPI can benefit from software quality management ideas. Fichman and Kemerer (1997) discuss organizational barriers towards adoption of software process innovations. Abrahamsson (2000, 2001) discusses tactics to ensure and manage commitment from different stakeholders. Nielsen and Nøerbjerg (2001) emphasize social and organizational issues in SPI. Aaen (2002) suggests engaging software developers more actively in SPI, and

Borjesson and Mathiassen (2004) argue that it is important to balance practice pull and process push, and to spend more resources on deployment.

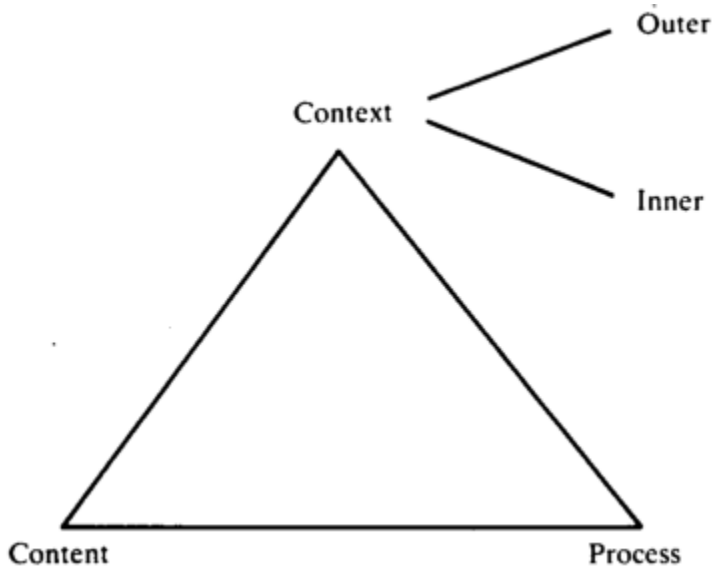
While these contributions suggest how to achieve SPI success, there are few studies that focus on the particular challenges and opportunities related to SPI in the context of the recurrent development of packaged software (Allison & Merali, 2007; Carmel & Becker, 1995;Sawyer, 2000; Xu & Brinkkemper, 2007). As a consequence, there is little known about how to improve release practices and how to leverage such processes in the wider context of SPI for the recurrent development of packaged software. Furthermore, there is a need to investigate the emergent nature of SPI, rather than consider it a deterministic activity. The emergent aspect of SPI would consider the design and action of the change process as being intertwined and shaped by their context in the recurrent development of software (Mathiassen, 1998; Truex, Baskerville & Klein, 1999).

In conclusion, the SPI literature has much to offer in terms of improving software quality, meeting stakeholder expectations and boosting efficiency, but it has not been applied to release processes in the context of recurrent software. Hence, there is a research opportunity to rethink release management and its relation to SPI in that context. Against this backdrop, this dissertation seeks to contribute to the literature of SPI and software release management for the recurrent development of software. Moreover, our aim is to make the empirical insights gained from our problem diagnosis, interventions, and learning from Software Inc., helpful to both practitioners and academic researchers.

# ANALYTICAL FRAMEWORK

We adopted Pettigrew's (1987 & 1990) contextualist inquiry framework to investigate the changes for improvement in the release cycle processes at Software Inc. 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 focused on how releases were managed and on how process improvement could be supportive at Software Inc. 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 release cycle processes, we were particularly interested in how the actors and social support elements of the context shape, and were shaped by, the process of improving release management. Finally, process refers to the actions and interactions between various interested parties as they attempt to transform practices. In our case, we focused on the actions and interactions related to the improvement in processes through the IDEAL model (McFeeley, 1996) within Software Inc. This dissertation used contextualist inquiry's core constructs (Table 2) to analyze the problems at Software Inc.

**Figure 1: Contextualist Inquiry Diagram (Pettigrew, 1987)**




Next, we will provide a brief account of the contextualist research approach in terms of its basic concepts, and the ensuing framework for guiding this research. This is necessary, not only for the sake of completeness of this dissertation, but also because, in a methodologically oriented work like this, it would be difficult to appreciate the value and validity of our results without having a basic understanding of the analytical approach used to arrive at them. In essence, the contextualist approach arises out of a conviction that, to be understood and studied effectively, organizations must be seen as "embedded" in and interacting with their social, cultural, political and historical context. The immediate effect of such a dynamic view of organizations is a profound shift of the researcher's attention and analysis away from mere "change" to a whole new kind of contextually driven, dynamic, analysis of the "process" of change in organizations.

**Table 2: Key Analytical Constructs**

| Constructs | Definition | Application |
|---|---|---|
| *Outer Context* | Outer context refers to the social, political, economic and competitive environment in which the firm operates. | *Outer context* represented the environment outside the *Secure-on-Request* unit. While *inner context* was the environment within the *Secure-on-Request* unit.<br>These constructs helped in understanding key issues and opportunities related to release activities and process improvement at Software Inc.<br>*'Why' of Change* |
| *Inner Context* | Inner context refers to the structure, corporate culture, and political context within the firm though which ideas for change have to proceed. | |
| *Content* | Content refers to the particular areas of transformation under examination. | Release activities and process improvement of *Secure-on-Request* at Software Inc.<br>*'What' of Change* |
| *Process* | The process of change refers to the actions, reactions and interactions from the various interested parties as they seek to move the firm from its present to its future state. | Improvement process guided by IDEAL model for transforming release activities and SPI of *Secure-on-Request*<br>*'How' of Change* |

In addition, Pettigrew (1987 & 1990) explicitly draws our attention to the relations or "interconnections" among the context, content and process concepts. Pettigrew's following comments are insightful: "The analytical challenge is to connect up the content, contexts and the processes of change over time to explain the differential achievement of change objectives. Perhaps the most critical connection is the way actors in the change process mobilize the contexts around them and in so doing provide legitimacy for change. Changes in the outer context can also be mobilized to fashion change …The contexts…are not inert or objective entities. Just as managers and other actors perceive and construct their own versions of those contexts, so do they subjectively select their own versions of the environment around them and seek to reorder the … change agenda to meet perceived challenges and constraints." (Pettigrew, McKee & Ferlie, 1988). These comments are enlightening because they clearly emphasize the need to study the drivers of change (context and process) and their interactions. They also remind us that the interactions involved need to be studied over a period of time and should examine the systems' space-across the organization's hierarchic levels- looking at subjective as well as objective aspects.

In terms of concretely applying the contextualist approach, we identified and examined multiple levels of the context involved, recognized the role of history, the present actors and the processes in the *Secure-on-Request* unit. In developing the contextualist analytical framework for our analysis, we also incorporated the various attributes of the recurrent development of *Secure-on-Request* as the 'content for change,' and the phases of the IDEAL cycle (McFeeley, 1996) as the 'process of change.' The resulting analytical framework (Table 3) helped us organize a systematic inquiry into the context, content, and process involved in transforming the release

management and process improvement at Software Inc. As such, the analytical framework was an ideal lens to study the transformation of release activities and the organizing of SPI to help Software Inc., recurrently develop and deliver *Secure-on-Request* to its customers and the market.

Consequently, the analytical framework, shown below, helped us form the concept of RCM during our interventions at Software Inc. RCM will be elaborated upon and precisely defined later in the study.

**Table 3: Analytical Framework**

| | | *Diagnosing* | *Establishing* | *Acting* | *Learning* |
|---|---|---|---|---|---|
| *Outer Context* | Competitors | | | | |
| | Market | | | | |
| | Customers | | | | |
| | Software Inc. At Large | | | | |
| *Inner Context* | People | | | | |
| | Technology | | | | |
| | Management | | | | |
| | Structure | | | | |
| | Culture | | | | |
| | Politics | | | | |
| *Content* | Recurrent Product Development (Business Strategy, Product Characteristics, Release Frequency) | | | | |
| | Release Cycle Process(Development, Testing, Documentation, Demonstration, User Acceptance, Delivery) | | | | |
| | Release Cycle Management (Planning, Monitoring, Improving, Communication) | | | | |
| | Release Cycle Organization (Roles, Technologies, Structures) | | | | |

# RESEARCH METHODOLOGY

Action research is a form of collaborative social research (Miles & Huberman, 1994). Our research at Software Inc., was conducted as an action research study to improve recurrent release management practices and software development processes for *Secure-on-Request* software. Our general research approach was collaborative practice research (CPR), a type of action research in which methodological pluralism and collaboration between researchers and practitioners is emphasized (Mathiassen, 2002). Through CPR methodology, we worked towards understanding the release management and software engineering practices at Software Inc., through interpretation, and improving the release management area by making interventions (Mathiassen, 2002).

Baskerville and Wood-Harper (1998) note that action research characteristics are orientated toward the research process rather than merely the outcome of the research. Action research methods are highly pragmatic in nature (Baskerville & Wood-Harper, 1996). Kurt Lewin (1952), who is often cited as the originator of action research, believed that knowledge is originated from problem solving in real-life situations. We believe the real-life problem at Software Inc., presented in this dissertation is exactly what Baskerville, Wood-Harper and Kurt Lewin have referred to. Our action research introduced changes to *Secure-on-Request*'s complex release management and SPI processes, and observed the effects of these changes at Software Inc. The social interaction that took place throughout this action research process was important as it helped to bring about organizational learning and change at Software Inc., to improve release management and SPI practices. By improving the release management and related

software processes at Software Inc., a real-life problem-solving situation, we generated knowledge in the form of empirical insights gained from our problem diagnosis, interventions, and learning which will help both practitioners and academic researchers. Hence, the outcome is an increased understanding of the social situation, practical problem solving and an expansion of scientific knowledge. The essence of this action research, laid in its objectives of advancing both the software release management and SPI theories in research, as well as facilitating the resulting organizational change at Software Inc. (Lee, 2003; Mumford, 2001).

Clark observes that: “for convenience it is useful to think of the practitioner as part of a set of actors who are oriented to solution of practical problems, who are essentially organizational scientists rather than academic scientists” (Clark, 1972, p. 65 in (Baskerville & Wood-Harper, 1996). As action researchers, we agreed to a set of rules to ensure a collaborative framework for action with those already working at Software Inc.  Shared Dissertation Platform Document (Appendix A) provides more details on the overall research approach used for this study.

**IV.I Problem-Solving Cycle**

In our problem solving cycle, we collaborated with Software Inc., to support release management innovation and proceed in a stepwise, iterative fashion, based on the approach described in the IDEAL model (Figure 2). This model, developed in 1996 by the Carnegie Mellon University Software Engineering Institute, presents a five-phase (Initiating, Diagnosing, Establishing, Acting, and Learning) cyclic approach to SPI (McFeeley, 1996). The IDEAL model can be seen as a specialized version of Susman and Evered's (1978) classical action research.

**Figure 2: IDEAL Model (McFeeley, 1996)**

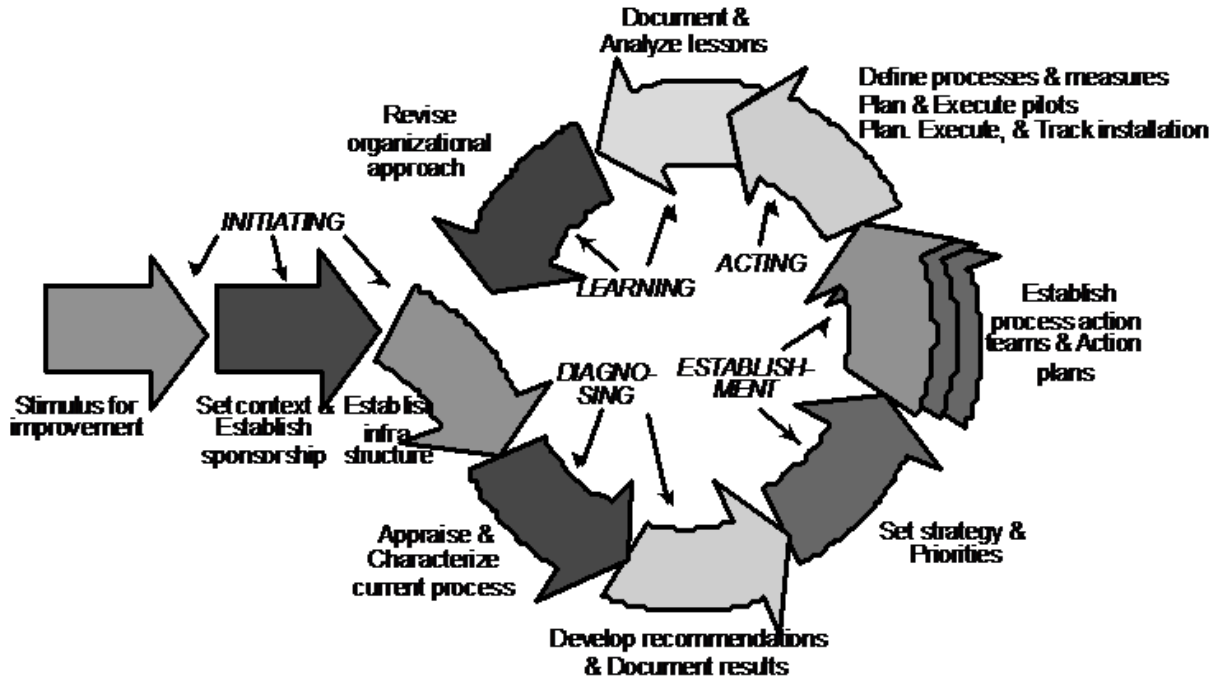


**Table 4: Problem Solving Timeline**

| Cycle Phase | Phase Timeline | Phase Overview |
|---|---|---|
| Initiation | January 5, 2013 – April 9, 2013 | Obtained commitment, set goals and established an improvement infrastructure |
| Diagnostic | April 9, 2013 – June 28, 2013 | Assessed current practices; developed and prioritized recommendations for improvements |
| Establishment | June 28, 2013 – July 2, 2013 | Created specific, focused improvement initiatives. Teams were established to deal with each of the recommended improvement areas from the diagnostic phases |
| Acting | July 2, 2013 – October 26, 2013 | Developed and implemented solutions for each improvement area. |
| Learning | October 26, 2013 - February 28, 2014 | Evaluated results of the initiatives. Improvements data were collected |

As the research project was organized according to the IDEAL model (McFeeley, 1996), this structure is also used in presenting the problem solving cycle (see Table 4). After initiating the project, we diagnosed existing strengths, weaknesses, and opportunities with respect to release practices. These insights fed an intervention cycle, focused on establishing improvement teams to recommend suggested changes, and then acting upon those suggested changes. The collaboration closed with a learning phase which asked identified stakeholders to reflect upon the initiative's impact and the effectiveness of the improvement organization. The Shared Dissertation Platform Document (Appendix A) contains an overview and more details on the

IDEAL model and the problem-solving cycle of this research. Next, we will focus on each phase in the IDEAL cycle:

1) The initiation phase is the initial step in the IDEAL model (McFeeley, 1996). In this phase, the Software Inc., senior management understood the need for the SPI, and along with Georgia State University (GSU) committed to the SPI program. From the release cycle perspective, we defined the context for SPI during this phase. Getting the Software Inc., management commitment and support was vital to bring about release management innovation in a way that would improve software engineering and management processes for the *Secure-on-Request* team. With Software Inc.'s solid management commitment, the study started on a strong footing. More details on the initiation phase are included in Shared Dissertation Platform Document (Appendix A).
2) The purpose of the diagnostic phase was to perform the baselining activity to get a picture of the current strengths and weaknesses in the release management area of *Secure-on-Request* within Software Inc. We believe release practices, viewed in a broader perspective, are the culmination of all the software engineering and management processes that are involved in a cycle of developing a new version of software. Release management intrinsically depends on and relates to these processes. Therefore, for our study at Software Inc., it made release management an obvious choice of area to start driving software engineering and management process improvement for the recurrent development of *Secure-on-Request*. During this phase, we investigated release management from a dual perspective. We focused on the release management activity itself, and used release practices as a lens to make sense of the *Secure-on-Request* recurrent development at large.

We reviewed *Secure-on-Request*'s organization structure and responsibilities, and evaluated baseline information needed against Software Inc.'s business drivers for SPI. We evaluated baseline information from the viewpoint of key stakeholders. We gathered information through perception-based as well as practice-based methods (Napier, Mathiassen, & Johnson, 2009). In the perception-based part of the assessment, we identified individuals from Software Inc., who are involved in the release process of *Secure-on-Request*, as well as internal and external customers. Participants' viewpoints were analyzed with a focus on strengths and weaknesses of the release management practices of *Secure-on-Request*. For our practice-based assessment, we selected release management principles identified in the release-management literature (Elephant, 2006; Team, 2006). We then benchmarked these principles and current release management practices at Software Inc. Based on the data collected (see Table 13) and observations, the research team assigned scores to Software Inc.'s release management practices, based on how they compared to the identified principles. The release management practice assessment and assigned scores are shown in Table 5 below.

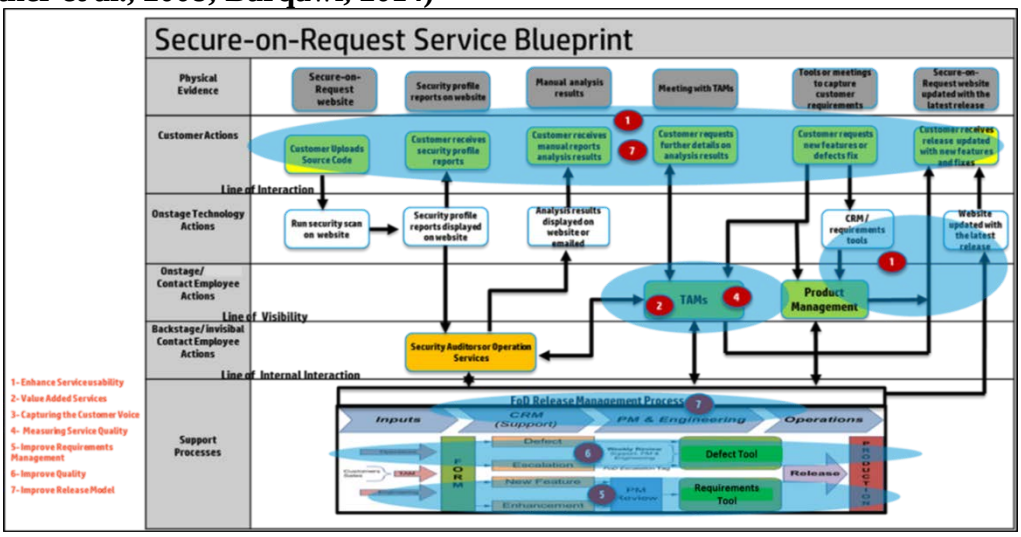
**Table 5: Release Management Practice-Based Assessment (Pre-intervention)**

| | Principle | Score |
|---|---|---|
| 1 | Define regular, targeted release dates | High |
| 2 | All deployments performed by a team independent of development team | High |
| 3 | Always have a tested back-out plan | Medium |
| 4 | Use a mature *Software Configuration Management* (SCM) process and tool to support the development of multiple releases in parallel | Medium |
| 5 | Test the deployment process at least once before deploying to live | Medium |
| 6 | Link all release documentation and scripts to your deployment unit | Low |
| 7 | Construct deployment units as early as possible | Low |
| 8 | Use an independent team to build all releases | Low |
| 9 | Automate as much as possible – use integrated tools for configuration, change management and deployment management | Weak |
| 10 | Have a documented Release Policy | Weak |

We presented the diagnosis and a portfolio of proposed improvements to the steering committee in June 2013 (Appendix G). As an outcome of the diagnostic phase, eight areas were identified for improvement, and these were: specifying and stabilizing requirements, prioritizing requirements across channels, managing technical debt, testing releases, managing release cycles, maintaining complete service information, communicating releases across customers and giving customers a voice. All these areas were interrelated and affect the release management of *Secure-on-Request*.

This dissertation also utilized the Service Blueprinting technique (Bitner et al., 2008; Barqawi, 2013). Service blueprinting revealed the complex context of *Secure-on-Request* in an easy way. Using service blueprinting for *Secure-on-Request*, we displayed possible areas for improvement and assigned the recommended project deliverables (during the establishment phase) for improvement, as it is illustrated in Figure 3.

**Figure 3: *Secure-on-Request* Service Blueprint at Software Inc. (Bitner et al., 2008; Barqawi, 2014)**




The information gathered in this phase was then used to initiate development of the strategic action plan that provided guidance and direction to the SPI program. More details for the diagnostic phase are included in Shared Dissertation Platform Document (Appendix A).

3) During the establishment phase, the steering committee developed a set of SPI strategic action plans that provided guidance and direction to the SPI program. These SPI strategic action plans were critical to provide clear guidance for the various process improvement actions that would be taken (McFeeley, 1996). The gaps identified during the diagnostic phase were prioritized, and strategies were developed for improvements, as explained in greater detail in Shared Dissertation Platform Document (Appendix A).

The SPI strategic action plans were approved by the steering committee in form of three projects: improvement of customer relations, improvement of requirements and quality, and improvement of release cycle. As a result, three project teams were formed. The three

projects were integrated into the baseline findings and the recommendations during the diagnosis phase. The project objectives and goals were well-defined. These projects provided clear engineering and business reasons for conducting the SPI program and were clearly and measurably linked to the organization's vision and business plan (McFeeley, 1996). Project schedules and milestones were determined, as shown in Table 6.

**Table 6: Improvement Projects Schedule**

| Projects Milestones | Target Dates |
|---|---|
| Project Start Date | July 2, 2013 |
| Midpoint Project Review | August 19, 2013 |
| Implementation Complete | October 26, 2013 |
| Lessons Learned | February 28, 2014 |

We know from SPI literature that organizations face problems with the implementation and deployment of SPI best practices. The majority of these problems belong to "people, group, team and community culture and behavior" (Dorrenbos & Combelles, 2004). The three project teams were made of individuals from cross-functional teams with diverse skills, both in technology and business. We used the broader perspective of the release management, presented in this action research, to address the problems Dorrenbos and Combelles have referred. The lens of release management provided the project teams with a shared understanding of evaluating and driving improvement of the broader engineering and management processes involved in the recurrent development of *Secure-on-Request* software. This approach helped us put a significant emphasis on the people and cross-team

collaboration aspects, and, therefore, the results from the projects were communicated easily at all the levels of the organization (Stelzer & Melis, 1999).

Support from the leadership team and operational preparedness were part of the three projects' deliverables committed by Software Inc. (Dyba, 2005; Niazi et al., 2006).

Improvement of Customer Relationship Project

The deliverables and assigned roles of the Improvement of Customer Relationship project are shown in Table 7.

**Table 7: Improvement of *Secure-on-Request* Customer Relationship Project**

| **Project Roles** | **Project Deliverables** | |
|---|---|---|
| - Project Manager: Release Manager<br>- Project Contributors: Business Owner, Product Manager, Technical Account Managers, Selected External Customers<br>- Project Consultants: Research team<br>- Project Sponsor: *Secure-on-Request* business owner | Enhanced Service Usability | - Identify ways to enhance the usability of *Secure-on-Request* website, from the end-user's perspective<br>- Effective and smooth communication of new features and releases to customers |
| | Value-Added Services | - Enhance TAMs team weekly status report |
| | Capturing The "Voice" of The Customer | - Early Adopters Program<br>- Customer Advisory Board (CAB)<br>- Web-based collaborative customer service software |
| | Measuring Service Quality | - Identify measurements that are related to service quality and establish a process for reporting them |

The steering committee understood that *Secure-on-Request* could be prevented from advancing until a common understanding of its functionality was established between the

engineering team and the customers. Therefore, the deliverables of this project were clearly identified for the engineering team to benefit from it. *Secure-on-Request*'s engineers must have a positive, collaborative relationship with their customers to improve the software engineering processes (Börjesson, Mathiassen, 2004; Mathiassen, Nielsen, & Pries-Heje, 2002). This view supports the broader perspective of release cycle processes presented in this research. The *Secure-on-Request*'s engineering team had to develop an understanding from their customers of how the application they were building was expected to function, when released. This need for engineers to better understand their customer expectations would improve engineering practices. This could be accomplished when release processes were considered as a driver of SPI.

As part of the project of improving the customer relationship, the research team working with Software Inc.'s key stakeholders recommended enhancing the service usability for *Secure-on-Request* customers. The team suggested that focusing on the usability features of the *Secure-on-Request* portal would enhance the service quality from the end-user perspective. Also, improving the release documentation process would result in the smooth communication of new features and releases to customers, and would consequently improve release management.

The research team recommended improving the TAMs team's weekly status report, which highlighted key information, such as customer concerns and Software Inc.'s responsiveness to the value-added services. The report was a tool used by management to deal with customers' issues, and it could also provide the engineering team with very important information about *Secure-on-Request* customers.

Capturing the “voice” of the customer was crucial to ensure the *Secure-on-Request* product team understood its customers’ expectations. It was also a key for improving the customer relationship with the *Secure-on-Request* engineering team. Capturing the “voice” of the customer enabled the *Secure-on-Request* engineering team to better understand the customers’ perspective. The Early Adopters Program was an initiative in which Software Inc., received feedback from customers about new product features prior to the formal release. In addition, the web-based customer service collaborative tool was a valuable source for *Secure-on-Request*’s engineering team to understand the needs of their customers.

For measuring service quality, the research team proposed that Software Inc., measure the *Secure-on-Request* service delivery processes against SaaS-Qual service quality factors (Benlian, et al., 2011; Barqawi, 2013). The conceptual definitions of the Six SaaS-Qual factors are shown in Table 8. These measurements will benefit the engineering team by providing more information about the *Secure-on-Request* business, which in turn will help to deliver better value solutions to customers.

**Table 8: Conceptual Definitions of the Six SaaS-Qual Factors (Benlian, et al., 2011; Barqawi, 2014)**

| Factor | Conceptual Definition |
|---|---|
| Rapport | Includes all aspects of an SaaS provider's ability to provide knowledgeable, caring, and courteous support (e.g., joint problem solving or aligned working styles) as well as individualized attention (e.g., support tailored to individual needs) |
| Responsiveness | Consists of all aspects of an SaaS provider's ability to ensure that the availability and performance of the SaaS-delivered application (e.g., through professional disaster-recovery planning or load balancing) as well as the responsiveness of support staff (e.g., 24-7 hotline support availability) is guaranteed |
| Reliability | Comprises all features of an SaaS vendor's ability to perform the promised services in a timely, dependable, and accurate fashion (e.g., providing services at the promised time, provision of error-free services) |
| Flexibility | Covers the degrees of freedom customers have to change contractual (e.g., cancellation period, payment model) or functional/technical (e.g., scalability, interoperability, or modularity of the application) aspects in the relationship with an SaaS vendor |
| Features | Refers to the degree the key functionalities (e.g., data extraction, reporting, or configuration features) and design features (e.g., user interface) of an SaaS application meet the business requirements of a customer |
| Security | Includes all aspects to ensure that regular (preventive) measures (e.g., regular security audits, usage of encryption, or antivirus technology) are taken to avoid unintentional data breaches or corruptions (e.g., through loss, theft, or intrusions) |

Improve Requirements and Quality Project

The deliverables and assigned roles of the *Improve Requirements and Quality* project are shown in Table 9.

**Table 9: Improvement of *Secure-on-Request* Requirements And Quality**

| Project Roles | Project Deliverables | |
|---|---|---|
| - Project Manager: Release Manager<br>- Project Contributors: Development Manager, Product Managers, Quality Assurance (QA) Managers<br>- Project Consultants: Research team<br>- Project Sponsor: *Secure-on-Request* business owner | Requirement Management Process | - Visualization of requirements (wireframes) using software tools.<br>- Validation of requirements through meetings and sessions and unifying statements of all stakeholders. |
| | Quality Improvement Process | - QA to put together a regression test plan<br>- Process to analyze escaped defects each release<br>- Scheduled weekly meetings with Dev to demonstrate new completed features to QA<br>- Single point of QA contact for the development<br>- Automation – Utilize performance testing to address performance business goals<br>- QA to develop end-to-end business scenario based testing |

Complex software like *Secure-on-Request* is more prone to defects (Kemerer, 1995). The complexity of *Secure-on-Request* influenced development defects (Banker, Slaughter, 2000), and also production defects after the release (Banker, Datar, Kemerer, Zweig, 1993; Banker, Davis, Slaughter, 1998). Development defects are

those discovered prior to release, while production defects are bugs found after the release (Harter, Kemerer, Slaughter, 2012).

On one hand, scholars like Juran (1959 & 1992), Deming (1992), and Crosby (1979) have long advocated process improvement as a means to improve quality in product development. Numerous studies have established a positive relationship between SPI and software quality (Herbsleb, Zubrow, Goldenson, Hayes, Paulk, 1997; Krishnan, Kellner, 1999; Li, Rajagopalan, 1998; Ramasubbu, Mithas, Krishnan, Kemerer, 2008). On the other hand, according to Lahtela & Jantti (2011) a well-defined release-management process can be pivotal to improving the quality of release planning, building, testing, and deployment activities. This will likely reduce the number of problems occurring after delivering the release to customers.

We find support to the argument presented in this research from the above literature. Quality is strongly tied to SPI and software release practices. Hence, in this project, using the broader lens of release management as a driver of SPI, we understood, evaluated and helped to drive improvement of the quality processes involved in the recurrent development of *Secure-on-Request* at Software Inc.

To improve the quality of *Secure-on-Request*, the research team recommended putting processes in place for:

- Regression testing to ensure that a change for the new release did not introduce new defects.
- Identifying and addressing common causes of both development and production defects.
- Holding weekly development demonstrations for the QA team.

- Establishing a clear line of communication between development and QA leaderships.
- Taking advantage of QA automation tools.
- Running end-to-end scenario-based testing, that depicted actual procedures used by most *Secure-on-Request* customers.

An accurate understanding of the customers' requirements was crucial for an effective release. Poorly understood requirements create uncertainty (Mathiassen et al. 2008). For better requirements management, the team recommended using specialized software tools for developing visual templates of requirements to help the *Secure-on-Request* development team to implement customer requirements. The team proposed that meetings be held to validate and align requirements coming from different stakeholders.

Improve Release Cycle Project

The deliverables and assigned roles of the Improve Release Cycle project are shown in Table 10.

**Table 10: Improvement of *Secure-on-Request* Release Cycle Project**

| Project Roles | Project Deliverables | |
|---|---|---|
| - Project Manager: Release Manager<br>- Project Contributors: Development Manager, Product Manager, QA Manager<br>- Project Consultants: Research team<br>- Project Sponsor: *Secure-on-Request* business owner | - Revised Release Model | - Change the release frequency from 30 days to 60 days. Longer release cycles would allow for process improvement. |
| | - Customer Communication Strategy | - Revised release frequency to be communicated to customers, and benefits of these changes to be explained |

Given that *Secure-on-Request* changes could occur on a continuing basis, one concern for release management at Software Inc., was determining when to issue a new release. The severity of the problems addressed by the release, and measurements of the fault densities of prior releases, affected this decision (Sommerville, 1995). Optimizing the release cycle of *Secure-on-Request* would improve the release management practices. The team recommended changing the release cycle from a 30-day to 60-day release model. This adjustment to the release model would allow changes to other areas in the release-cycle and contribute to maturing the software processes. For example, sufficient time would be allotted

for implementing the requirement and quality process improvements suggested in the previous project. The extra duration of the new release cycle would also contribute to the recommended documentation process changes (as stated earlier) that would subsequently improve customer communication and eventually make the release more successful. To improve customer communication during a release cycle, the research team also proposed a plan for communication to customers, involving product management and Technical Account Manager (TAM) teams.

The required stakeholders of the three projects agreed on the recommended improvement approach and execution plan (Appendix A, Table 4.3-2).

As stated above, improving the release model would positively impact all of the software engineering and management processes in the context of the recurrent development of *Secure-on-Request*. This project illustrated the core of the argument presented in this action research, that from a broad perspective, software release practices can be seen as the culmination of all the software engineering and management processes involved in one cycle of developing a new version of *Secure-on-Request*. In this sense, the release was a unique and important area that depended on, and was intrinsically related to, the other engineering and management processes, and as such it would drive their improvement.

4) During the acting phase, the three project teams started developing improvements and solutions to the process issues approved by Software Inc. The key problems discovered during the diagnosing phase were prioritized and selected during the establishing phase. Shared Dissertation Platform Document (Appendix A) has more details and key dates of the

acting phase activities at Software Inc. The acting phase ended on October 26, 2013. The following is an overview of our activities during the acting phase for each project:

Improvement of Customer Relationship

- *Enhanced Service Usability:*
  - To identify ways to enhance the usability of the *Secure-on-Request* portal, from the end user's perspective, the research team worked with the TAM team to provide a list of requirements that could enhance portal usability. The list was prioritized and communicated to the product management and engineering teams. As a result, most of the items on the list were placed on the product management roadmap.
  - Product managers took ownership of coordinating the documentation process to achieve effective and smooth release communication to customers. The documentation team and product managers started working early in the release cycle to review and identify documentation impact activities.
- *Value-Added Services:*
  - In order to enhance the effectiveness of the TAM team weekly status report, the research team discussed the summary report with management and TAMs. A summary section was added to the report, which included main items for quick review.
- *Capturing the Voice of the Customer:*

- Regarding the early adopters program, introductory meetings between PMs and customers that were identified as early adopters were completed. Customers reported positive feedback, and more meetings for discussing requirements and evaluating features were scheduled.
- The TAM management and the research team worked on the CAB initiative. Information and sample agendas were discussed, and a list of customers was identified. A CAB meeting was held in September 2013 at Software Inc., conference for customers.
- Several demonstrations of the web-based collaborative customer service tool were carried out by potential vendors. The solutions included live chat, ticketing, and knowledge-based management systems. A solution was chosen, and the development team implemented the integration of the tool within Secure-on-Request website.

o *Measuring Service Quality:*

- To identify measurements that were related to service quality and to establish a process for reporting them, the research team discussed service quality measures with TAM and product management teams. A list of measurements was recommended for measuring service quality, renewal rates, expansion (new customers) and open and closed tickets.

Improvement of Requirements and Quality

o *Requirement Management Process*

- To use visualization aids (screenshots, mockups, etc.) for requirements, a software tool was used by product managers to develop visualizations of requirements for the development, quality assurance and documentation teams.
- Requirements validation meetings started with all stakeholders, including product managers, TAMs, quality assurance, and development teams during the requirement gathering process. User acceptance criteria for requirements implementation was also put in place.

- *Quality Improvement Process*
  - Various initiatives were put in place to improve QA related processes. The QA team put together a regression test plan to ensure that a change for the new release did not introduce new faults. A process was put together to analyze escaped defects during a release. Weekly development demonstration sessions of the completed features were initiated for QA. A clear line of communication was established between development and QA leaderships. The QA team started putting together a plan of action to utilize automation tools. The QA team then started running end-to-end scenario-based testing, which depicted the actual procedures used by most Secure-on-Request customers.

Improvement of Release Cycle

- *Revised Release Model*

- The release frequency was changed from 30 days to 60 days. See Figure 4 for the new release model. Longer release cycles allowed for processes improvement, and consequently this helped to improve the Secure-on-Request quality. A release model was developed by the release manager and was agreed upon by all stakeholders. The first Secure-on-Request release following this model was made on October 19, 2013. Table 11 shows the set of meetings for the new release model.

- *Customer Communication Strategy*
  - The new release frequency was communicated with customers, and the benefits of these changes were explained by product managers and TAMs.

5) In the learning phase, we reviewed the implemented solutions and evaluated the outcome of the three improvement projects. Shared Dissertation Platform Document (Appendix A) has details and key dates of the learning phase activities at Software Inc. Our learning phase assessments included perception-based as well as practice-based methods (Napier et al., 2009) with a focus on evaluating the impact on the service-delivery process of *Secure-on-Request*. Our goal was to identify changes in each of the three project improvement areas, the effect on the processes, as well as the challenges that occurred during implementing the changes, and to make suggestions for improvement. For the practice-based part of the assessment, we used the norms and practices from release management literature that were identified in the diagnostic phase (Elephant, 2006; Team, 2006) and compared them to software release management practices at Software Inc., after the implementation of the improvement projects. The research team assigned scores based on data collected and

observations, and the assessment results were compared against those from the diagnosing phase. The resulting assessments are summarized in Table 12. An overall assessment of the improvement projects will be discussed in Chapter VI.

**Figure 4: The New Secure-on-Request Release Model**

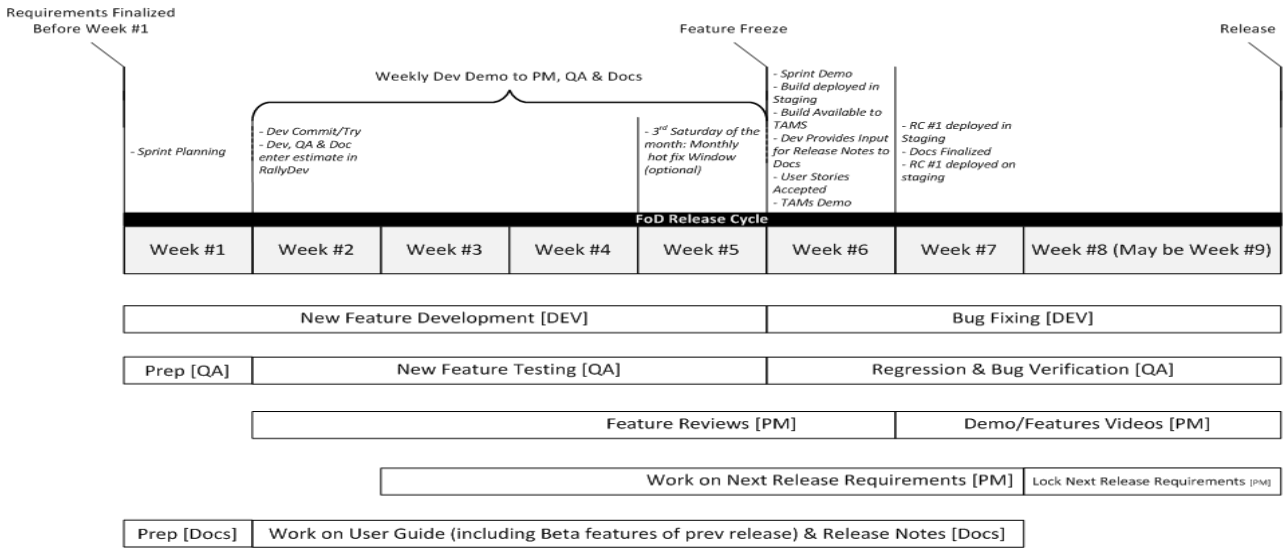


**Table 11: Reoccurring Meetings for the New *Secure-on-Request* Release Model**

| Meeting | Purpose | Schedule |
|---|---|---|
| Stakeholders Meeting | Gather and discuss input from key stakeholders for the requirements of the next release | Thursday of week #7 of previous release |
| Sprint Planning | Sprint team to negotiate the scope for the release with the product manager | Monday of the Week #1 |
| Product Manager and Docs Review | A requirements walkthrough by the product manager, for the documentation lead, engaging the documentation lead much earlier in the release cycle | Tuesday of week #1 |
| Weekly Development Team Demonstration | Weekly demonstration by the development team to product managers, quality assurance and the documentation teams for the completed new features during previous week. Week #2 thru Week #5. | Wednesdays of week #2 thru week #5 |
| Sprint Demonstration | Development team to show the stakeholders the work they have accomplished for the entire release cycle | Monday of week #6 |
| TAMs Demonstration | Product manager to show TAMs the new set of features included in the upcoming release | Tuesday of week #6 |
| Sprint Retrospective | A session for stakeholders to learn what worked and what did not work during the previous release cycle and subsequently make the necessary adjustments for the next release cycle based on the learnings | First Thursday after the release |

**Table 12: Release Management Practice-Based Assessment (Post-intervention)**

| | Principle | Score |
|---|---|---|
| 1 | Define regular, targeted release dates | High |
| 2 | All deployments performed by a team independent of development team | High |
| 3 | Always have a tested back-out plan | High |
| 4 | Use a mature *Software Configuration Management* (SCM) process and tool to support the development of multiple releases in parallel | Medium |
| 5 | Test the deployment process at least once before deploying to live | High |
| 6 | Link all release documentation and scripts to your deployment unit | High |
| 7 | Construct deployment units as early as possible | High |
| 8 | Use an independent team to build all releases | High |
| 9 | Automate as much as possible – use integrated tools for configuration, change management and deployment management | Weak |
| 10 | Have a documented Release Policy | High |

## IV.II Research Cycle

In parallel with the problem-solving cycle, the research unit at Software Inc., concentrated on adding new knowledge to recurrent development of software, SPI and software release streams of literature. This cycle was guided by the style composition for action research developed by Mathiassen, et al. (2012) (Table 1). We reviewed recurrent development of software, SPI and software release streams of literature, and this dissertation drew upon Pettigrew's contextualist inquiry theory(1987 & 1990). The research process was a collaborative and iterative process, focused on problem diagnosis, change, and reflection (Avison et al., 2001). Details on how our study satisfied the three methodological characteristics (Baskerville & Wood-Harper, 1996) and how it deals with the three dilemmas (Rapoport, 1970) are covered in the Action Research Design section in Shared Dissertation Platform Document (Appendix A). This document also covers in detail how Canonical Action Research (CAR) principles of action research were followed during our action study at Software Inc., to ensure rigor in our study. (Davison et al., 2004).

## IV.III Data Collection

We collected rich data from multiple primary and secondary sources (Myers, 2008), all through our collaborative study. Using the guidelines from Yin, (2008) and Miles and Huberman (1994), the principle data sources included semi-structured interviews, and problem solving cycle documentation. We identified key individuals from Software Inc., to be interviewed for our study. We conducted approximately one-hour face-to-face interviews, as well as phone interviews. All interviews were conducted in English, and detailed notes were taken. All of the

interviews were recorded. During the course of our data collection, we used triangulation (Miles & Huberman, 1994) to counterbalance any insider bias (Coghian, 2001). Table 13 shows the primary and secondary data sources we used in our research.

**Table 13: Data Sources**

| Primary Data Sources | Secondary Data Sources |
|---|---|
| **Meetings:**<br>- Release Management Meetings (Weekly)<br>- Bi-Weekly Scrums<br>- Release Planning and Demos<br>- Daily Customer Escalation Calls | **Release management documentation tools:**<br>- Rally Dev - Requirements tool<br>- Bugzilla - Defect Management tool<br>- Zoho – Customer Relationship Management tool |
| **Semi-structured interviews:**<br>- Professional Services<br>- Sales<br>- Quality Assurance<br>- Product Management<br>- Operational Services<br>- Development<br>- Business Unit Owner<br>- Technical Account Management<br>- Project Management Office<br>- External Customer | |

## IV.IV Data Analysis

To pursue the contextualist approach, we needed to apply its characteristically content and process-oriented mode of analysis to the collected data, to see what major issues and problems emerged vis-à-vis the release cycle processes. Here, we felt a clear need to further operationalize the contextualist approach, especially its analysis aspects, to devise a practical way of analyzing

data to identify specific issues or challenges of possible empirical interest and significance. We developed a comprehensive coding scheme for this purpose (Appendix E). The overall rationale, and the general mechanics, of this analysis technique are explained below.

First, Pettigrew's notion of 'content,' as "the what" of change (Table 2 & 3), sensitized us to look for specific entities that are subject to possible change, as well as the particular nature of that change. As the principal arena of observable changes of interest, we focused our attention particularly on the release cycle processes (development; testing; documentation; demonstration; user acceptance and delivery), release cycle organization (roles, technology and structure) and release cycle management (planning, monitoring, improving and communicating) activities within the inner context of the *Secure-on-Request* unit. Given our interest in more of a planned future of the *Secure-on-Request* unit, we looked for both the "reactive change" in response to environmental (outer context)pressures, and the planned "design change" that may be intrinsically desirable from Software Inc., and its customers perspective on the *Secure-on-Request* unit.

Second, the notion of "process," as "the How" of change (Table 2 & 3), prompted us to examine significant "actions, reactions and interactions" of any "actors" in terms of how they caused or affected any changes in the *Secure-on-Request* unit. In either case, starting either from the content end or from the process end, what we were looking for was a "significantly coupled chain" of actors, their actions, and the changes that they caused or influenced.

We then scanned through the data to see if interesting themes emerged. We focused on the changes in the *Secure-on-Request* unit that were expected to be either important or controversial, and then identified the possible actions and the relevant actors that were likely to

impact those changes. In the other direction, we also scanned the entire data (inner and outer context) for any actors that seemed to reveal a deductive influence, and we critically examined their possible actions to see if a significant change process could be substantiated.

A major strength of this approach was its built-in objectivity, enhanced by the logical and objectively grounded reasoning, which the researcher must follow to systematically trace and validate a possible chain before declaring it as important.

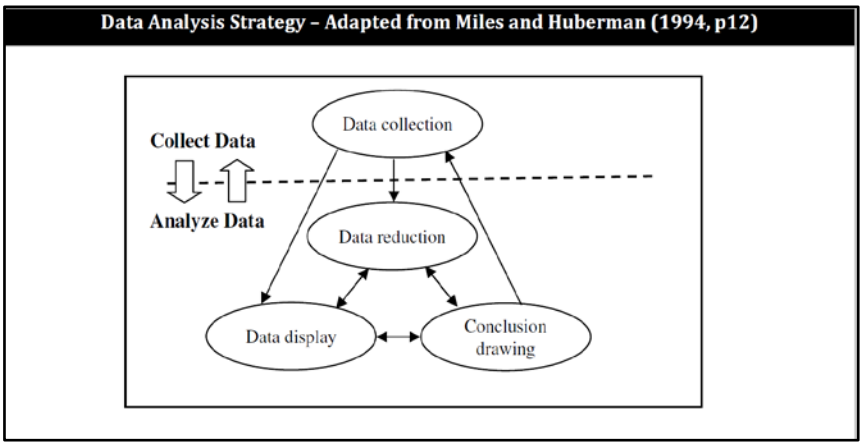
Furthermore, transcribed interviews and meetings, researchers' notes, email communications, and system performance data were reviewed multiple times. This analysis was completed according to the data analysis procedures proposed by Miles and Huberman (Miles & Huberman, 1994) for qualitative data analysis. Despite the more abstract nature of qualitative research (by comparison to quantitative methods), rigorous approaches to data analysis were developed which provided solid evidentiary support to the conclusions and insights. There were methods to organize process, analyze and evaluate information from the qualitative data acquired through well-designed research. Data was analyzed by entering into NVIVO software (Appendix E). It was then coded and reviewed. The assessment data was further reviewed through the research framework, as stated in the previous section. Key research themes were coded in the transcriptions of the interviews and meetings, researchers' notes, email communications, and system performance data—all of which were imported to NVIVO (Appendix E). Subsequent to the coding process, data analysis began. According to Miles and Huberman (Miles & Huberman, 1994), coding is helpful for the interpretation phase. We began coding with an initial list. However, as the process continued, the initial coding list was enhanced to facilitate sense making. Fundamentally, the key objective of the data analysis is to address the research

question. With that in mind, the data analysis processes in this research were aligned with the three distinct components, as defined by Miles and Huberman (Miles & Huberman, 1994): data reduction, followed by data display, and finally conclusion drawing and verification (Figure 5).

### IV.IV.i Data Reduction

Data acquired during the research was continuously extracted and filtered through the analytical lens and the general research themes. As such, Miles and Huberman (Miles & Huberman, 1994) specifically describe data reduction as, “the process of selecting, focusing, simplifying, abstracting, and transforming the data that appear in written-up field notes or transcriptions.” The description of this phase—reduction—says it all. As in other phases, the analytical lens for the research provided the backdrop so the ‘data reduction’ occurred within a context rather than autonomously. The coding process sharpened, sorted, focused, discarded, and organized collected data, which made it relevant to the research question and as a foundation for the remaining data display, results and discussion sections. Significant portions of the transcribed interviews and meetings, researchers’ notes, email communications, and system performance data were marked and reviewed for inclusion in the subsequent analysis and presentation.

**Figure 5: Data Analysis Activities**




### IV.IV.ii Data Display

Data display is the second flow of data analysis recommended by Miles and Huberman (1994). Like data reduction, the processes of creating data displays are an iterative process occurring throughout, and following, the data collections process. Classification and organization characterized this phase, where data was displayed through a variety of formats. These presentations helped us to view the data in a systematic structure that enabled pattern observations and sense making. The data display phase allowed us to perceive greater insights that might not have surfaced in the more detached data reduction phase. Data display through tables, charts, models and matrices (Table 3, 7, 9, 8 & 11; Appendix F) revealed patterns that helped us to draw conclusions.

### IV.IV.iii Conclusion Drawing and Verification

Interwoven with data reduction and data display were the conclusion drawing and verification processes. Before, during and after the data collection process, we drew conclusions by noting regularities, patterns, explanations, possible configurations, and propositions from

available data. These conclusions were held lightly in the beginning. However, during the course of this research, through conclusion drawing and verification, sense making and meaning evolved stronger and stronger through substantiating the insights. Hence, conclusions became increasingly explicit and grounded throughout the process (Miles & Huberman 1994, p11). Verification occurred as the data was reviewed through iterations and reflection. The key focus was to maximize objectivity and develop sound arguments for conclusions. It was important during this phase to assess inconsistent and contradictory data. Miles and Huberman (Miles & Huberman, 1994) refer to these as “surprises” and confirm the necessity of “checking the meaning of outliers” and of “using extreme cases.”

# RESULTS

In this chapter, through the empirical results of our study, we describe how Software Inc. organized and improved release management to help recurrently develop and deliver *Secure-on-Request* to its customers. In adherence to Pettigrew's contextualist approach (1987 & 1990) we identify aspects of the process, context, and content (Table 2 & 3) for each phase (Diagnostic, Establishment, Acting and Learning) of the transformation to the *Secure-on-Request* release activities and the organization of SPI. (Appendix A, Table 4.0; Table 6).

## V.I Diagnostic Phase

### V.I.i Process

Going into this phase, it was important to understand how the people in the organization viewed the *Secure-on-Request* unit (Appendix A, Table 4.2-2). Such insights would help us to anticipate challenges and structure our approach to tackle the challenges at Software Inc., and it would serve as input to tailor release cycle processes later during the IDEAL cycle (Figure 2). If change was to happen, we had to consider employee's beliefs, issues and concerns. An important first step was, therefore, to understand the culture as it existed and how the employees believed it needed to change.

In the diagnostic phase, we established the foundation for the later phases in the process. The goal was to understand the current practices and challenges in software release management within Software Inc. We assessed existing software release practices related to *Secure-on-Request* to establish a baseline for interventions (Appendix A, Table 4.2-3). We collected data

between March 2013 and June 2013 (Appendix A, Table 4.2-1), including twelve semi-structured interviews, several meetings with Software Inc.'s, stakeholders, and a review of performance data extracted from the internal tracking systems (Table 13). Our assessment included perception-based methods from the interviews and meetings with Software Inc.'s stakeholders. It also included practice based methods in which we looked at performance data and reported results that were extracted from the tracking systems (Appendix G). We also reviewed the release management literature to understand relevant industry practices. Hence, for the practice-based component of the assessment, we selected norms and practices that were identified in the release-management literature (Elephant, 2006; Team, 2006) and compared them to current release practices at Software Inc. (Table 5).

In the perception-based part of the assessment, we identified individuals who were involved in the release process of *Secure-on-Request*, as well as internal and external customers (Appendix A, Table 4.2-2). The research team created an interview guide that discussed objective and subjective information about release management and service delivery related to *Secure-on-Request*. The research team met and analyzed the interviews to reflect upon emerging themes. Participants' viewpoints were analyzed with a focus on the strengths and weaknesses of current release management and service delivery practices (Appendix G).

During the course of the study, the steering committee was kept informed of the activities through weekly status reports and periodic status meetings. The research team documented the assessment findings in a complete diagnostic report (Appendix G), and a steering committee meeting was held on June 20, 2013 to present and discuss the findings and overall

recommendations. The meeting served to share the insights, it provided important feedback, and it helped prioritize actions during the establishing phase (Appendix A, Table 4.2-4).

**V.I.ii Content**

Release cycle processes in the *Secure-on-Request* engineering and product team were previously mostly ad hoc and chaotic. Release predictability for schedule, scope and quality was weak. As a result, the release cycle which occurred every month did not operate in a stable environment and most of the work was performed informally. Heterogeneous functions and features that had been added to the software due to its large and diverse customer base made the code complicated and vulnerable. In addition, monthly releases did not allow enough time for requirements analysis, testing, documentation and customer communication. For instance, the release at the time of the diagnoses in March 2013 had several poor quality features. As a result, the development team spent considerable time after the release fixing defects that had been mostly reported by customers, this was at the expense of working on the next release cycle. According to the development manager:

> *"The volume of development work in each release cycle that goes to fixing defects that comes out from the previous release is very high, I would say around 25%."*

Due to the short release cycles and the business ambition to release more features to beat the competition, there would not be enough time left in the release cycle after the development team had finished building new features. Often, the quality assurance team would get 1-2 days to test three weeks of development work. This was not enough time. According to the quality assurance manager:

> *"We don't have enough time between the end of the release and the time we put it out to get full quality regression tests done."*

As a result, the product would be released with poor quality due to minimal testing. This would cause a surge of customer escalations immediately after the release, which would force the development team to start fixing bugs from that release. This would trigger a series of frequent unplanned releases to rectify the quality issues.

The same time constraint, as mentioned above, would also impact documentation issues during every release cycle, as the product manager noted:

> *"Release notes and user guide documentations have been a real challenge because we have monthly release cycles. How can you write documentation if you are actually writing code the night before the release goes out? It is pretty hard"*

The release cycle planning process was also very weak. We discovered that unclear requirements caused confusion and rework. Requirements prioritization within and between new features was a major challenge. The product manager recognized the challenge of requirements prioritization by saying:

> *"Our maturity and our ability to move forward with the prioritization process isn't 100% there, and we all agree that is not what we want it to be in the long term."*

A friendly and comfortable relationship existed between *Secure-on-Request*'s business owner and most of the development team members. The business owner had worked with several of the members for over ten years in a previous organizational setup, and had developed close acquaintance with them over time. Since the product management and quality assurance teams were new and still settling in, the close relationship between the development team and the business owner strongly influenced key decisions during a release cycle. The business owner felt the product management team was too new to be fully functional. He said:

> *"The junior product manager has been around for a year now, but she doesn't know even as much as the senior product manager and he is new to the organization and the software."*

In the absence of fully functional product management and quality assurance teams, the long-term product vision and product quality were not always considered in releasing new features. Due to this dynamic, the development team was more concerned and involved with day-to-day crises. The business owner was  pushing the development team to catch up with the competition in the market, but much of their time was spent reacting to crises, at the expense of focusing on the long-term goals, such as building a solid roadmap for the product or developing a good understanding of the customers' expectations and needs. The business owner was "shooting for the moon," while the development team lacked attention to the long-standing benefits. For the most part, the development team tricked itself into thinking they were being productive. To a great extent, being busy made them believe they generated good results.

There were no mature tracking mechanisms and defined standards for the release cycle. As a result, there was a lack of visibility of planned features in a release cycle, among the team members. In addition, there were no processes in place for assessing processes and improving them. In short, the release cycle processes were unpredictable and were reacting to, rather than shaping, the business environment.

**V.I.iii Context**

*Secure-on-Request*'s high frequency releases meant new features were often made available to the customers. However, for some customers the monthly release cycles made it difficult keeping up with the frequent updates. According to the product manager:

> *"Frankly, the customers can't absorb these frequent updates and changes, and in the process we haven't been giving the customers enough time to know the service is changing."*

For some customers, the rapid release cycles were a problem because it disrupted the habits of their users, requiring changes in behavior largely due to major modifications in the interfaces of the product. The consequential change processes were complicated and at times costly. For example, in some cases the customers' IT unit had to test their systems to ensure the changes in the *Secure-on-Request* release did not break their internal processes. Sometimes the IT unit had to redevelop the glue code between the components to make their local systems connect to *Secure-on-Request*. In addition, there was very little or no information shared with customers by the *Secure-on-Request* product management team about new features and changes to existing features that would be included in the next release cycle. In most cases, customers would find out about changes after the release when their local processes were impacted. These kinds of surprises would make customers very frustrated. The product manager said:

> *"Customers have said things like: 'you guys just released all that stuff and we were not expecting it, we are glad you are doing all that kind of stuff, but we want advance notice.'"*

Also, in the absence of a reliable prerelease communication, customers did not always understand the added value in a release. The product manager stated:

> *"Lack of certain usability features is seen as defects by customers, but this is not how we see it."*

In addition, the Technical Account Managers (TAMs), who were the liaison between the existing customers and the product team, felt that because of its close relationship with the business owner, the engineering team was not giving appropriate priority to the issues that the TAM team identified in the customer feedback. Most of the time, this made TAMs frustrated. One TAM shared with us:

> *"Some engineering team members believe that what TAMs do in reality is all academic."*

As a result, TAMs were not very engaged in the engineering and product release processes. When we asked one of the TAMs about the release process, his response was:

> *"To be honest, I am sitting here trying to think, what is the Secure-on-Request release process? Maybe that's one of the weaknesses right there."*

The business owner and some of the other members of the business group believed beating competitors in the market and winning new customers would bring more revenue to the *Secure-on-Request* team than customer retention. However, TAMs, who represented existing customers at Software Inc., believed customer retention was equally important for success. This difference in perspective, at times, caused tension in the organization.

Due to his closeness to the business owner, at one point, the development manager suggested the development team saw TAMs as a distraction, based on the nature of their requests to his team. It revealed a problematic relationship between the development and TAM teams. Development manager said:

> *"The TAMs are actually more of a problem for us."*

In addition, the high release frequency increased the presence of bugs due to weak engineering processes. This adversely impacted the software quality and reliability which again negatively affected the customers' perception of the service.

There was no organizational learning. The success of *Secure-on-Request* largely depended on the individual heroism of key team members. The know-how of the software could easily be lost if an engineer left the company. This made the organization people-dependent. For example, one engineer shared his view about one of his colleagues:

> *"He is the guy who has all the knowledge so everybody tends to go to him. However, the knowledge needs to be distributed."*

Although monthly releases helped Software Inc., quickly catch up with competition in the market, it also contributed to constant deadline pressures. As a result team members worked in a fast-paced environment, which at times was chaotic because the expectations were high and the resources were limited. The engineering and product teams worked overtime to achieve results. This situation made attrition risks very high. The quality assurance manager, like many others in the team, was stressed due to the chaos around him:

*"I am trying to make this work with the environment that we have and it is stressful."*

Despite these challenges, trust and support among team members was high. As reflected in the 'High' rating in the release management practice assessment (Table 5), the team members were technically strong and experienced, and consequently, managed to negotiate quality issues one way or another. Managers and developers were committed to doing the best job they could under difficult circumstances.

### V.II Establishment Phase

Based on the GSU diagnostic report (Appendix G), the steering committee committed to continue working with the GSU research team for the next few months to change release practices in the *Secure-on-Request* unit (Table 6).

#### V.II.i Process

In the establishment phase (Appendix A, Table 4.3-1), we prioritized the issues that Software Inc., would address (Appendix A, Table 4.3-2), and we developed strategies for reaching solutions. We completed the detailed process improvement plan, based on the agreed strategy, and designed plans to execute it (Table 6, 7, & 9). The suggested improvement strategy

was implemented through three dedicated project teams with clearly identified deliverables and timelines. The steering committee members agreed to form three teams to work on three improvement projects: customer relations, software quality, and release cycle. The steering committee approved the overall plans for the improvements identified in the diagnostic phase.

**V.II.ii Content**

Thanks to the diagnosing phase (Appendix A, Table 4.2-3 & Table 4.2-4) we saw an improvement in the stakeholders' awareness, during the establishing phase. They were aware of the benefits of building a reliable release cycle (Appendix F), and they desired to become more disciplined in the software processes. The stakeholders now had a much better understanding of benefits, such as an improvement of productivity, time efficiency, product quality, customer satisfaction and increasing staff morale by better managing the *Secure-on-Request* release cycles.

Through the three improvement projects the steering committee members decided to increase the length of release cycles to sixty days (Appendix A, Table 4.3-2). This change would provide the much needed time for key activities, such as detailed requirements analysis, quality assurance, documentation and prerelease customer communication. This change also relaxed the development team members. According to a development engineer:

> *"This longer cycle will give us more time to develop better functionality in the core capability of the product and give us better focus."*

The change would involve the quality assurance team early in the release cycle to support development of test cases based on requirements. The new release model would strengthen collaboration between functional teams about requirements, test cases, test results, and defect correction (Table 9).

To finalize requirements for the October 2013 release cycle, steering committee members decided key stakeholders would share their input for requirements with the product manager in a standing meeting that would be held three weeks before the start of the new release cycle (Table 11). During the same meeting, the requirements would be prioritized mainly through consensus. However, in case of disagreement, the product manager and business owner would make the final decisions. After this step, requirements would be explicated and effectively shared across development, quality assurance and documentation teams. It was also decided that the product manager would, if needed, approve any changes to requirements in the middle of a release cycle. Furthermore, steering committee members agreed to make use of wireframes—a common practice to ensure effective communication between technical and business teams.

It was further decided that for better communication, development team would hold weekly demonstration sessions of the newly developed features for key stakeholders (Table 11). The weekly demonstrations would stress the importance of executable software as proof of progress. Each weekly demonstration would verify the system architecture, adherence to requirements and stakeholder needs, as well as the software quality. In addition, it would emphasize real progress during the release cycle by producing demonstrable results. It would give stakeholders visibility into the real progress, not the perceived and subjective view of progress. This would be possible because a working version of the system would be available for inspection each week during the development cycle, emphasizing an important concept. By practicing incremental development, the teams would stay focused on results.

The other key area of focus during this phase was emphasis on some key roles. The diagnosing phase had revealed that roles for TAMs, quality assurance engineers and product

managers needed to change to better manage the *Secure-on-Request* release cycle. Therefore, the process improvement projects focused on ways to make these roles stronger. Through their established relationships with customers, TAMs could play a more effective role to grow business with existing customers by working closely with the product management and engineering teams to provide improved solutions. In this regard, TAMs who believed there was a greater potential for generating more revenue from the existing customers needed to get support from the business owner (Table 7). The quality assurance team was still very new and needed to establish a strong presence with the rest of the functional teams to improve the much needed quality of the product (Table 9). Finally, the product managers could improve the requirements management practices to help the engineering team deliver better solutions. For example, the business owner thought the engineering team could benefit a great deal through the use of visuals like screenshots during requirements specification, especially for more complex requirements:

> *"In detailing our requirements, there should always be a picture or a screenshot (wireframe) of what it should look like if it is a customer-facing interface, so there will be no confusion."*

**V.II.iii Context**

After the diagnosing phase, stakeholders started to believe in building a more relaxed culture, which would provide the necessary focus and stimulus to the engineering team to continuously improve software processes during release cycles. The steering committee was committed to improve the chaotic culture in the engineering team. The product manager shared with us his thoughts on the existing culture:

> *"A lot of our guys are cowboys, cowboy developers, consultants who just want to figure out a way to hack it together and make it work. We want to resist doing it this way and fall off the wagon. I mean okay, we need this functionality, but we have to follow the*

*process. We need to put the requirements for it in our internal tracking tools first and then look at it there with other requirements and do it the right way."*

As a result, the improvement program was gathering more social support in the organization. The business owner had involved more people in the strategic planning. This helped create the suggested improvement strategy and carried it forward in a collaborative manner. The social commitment to the action research study showed a willingness to break the traditions and consider alternative ways of thinking. The steering committee members were open to direction, criticism, and new ways of thinking.

However, as we will share ahead, organizational politics and some resistance for change would prevent Software Inc., from easily realizing the SPI benefits. The biggest challenge was the close relationship which existed between the business owner and the development manager. For example, it was very common for them to bypass the regular flow of communication related to a routine release issue, thus keeping other important product functional groups like TAMs, quality assurance and product management out of the loop. This would later result in surprises. The quality assurance manager pointed to some of these dynamics:

*"The development manager is trying to please everyone (implying business owner). I think this is probably putting his team under tremendous pressure. Although he is a hard worker, everybody are hard workers, we cannot release high-quality products under this pressure. And, we can't keep this crisis management for the next six months. We have to do something about it."*

Related to the above, the development manager said:

*"Volume workwise, I would say 60% of requests to my team come through the front door and 40% come through the backdoor."*

Much of the "backdoor" requests to the development team came from the business owner due to his closeness with the development team. The business owner agreed that he gets involved in low-level details in the product changes:

*"I am pretty intimately involved in the details of the product. I have been in the market since 2004, so it has been quite a while, and I know the product very well. I know the competitors."*

This interplay between the business owner and the development team was setting a major tone in the organizational politics of the group.

### V.III Acting Phase

The acting phase began in July 2013 with the kickoff meetings of the three improvement projects (Appendix A, Table 4.4; Table 6).

#### V.III.i Process

In the acting phase, the GSU research team focused on the improvement projects to address the areas for improvement identified during the diagnosing phase (Appendix A, Table 4.3-1 & Table 4.3-2). The strategy and prioritization, as well as deliverables, were agreed upon in the establishment phase. The research team and steering committee members held a kickoff meeting for each improvement project. At the kickoff meetings, the teams were given a set of objectives and deliverables. The teams were provided with draft project plans along with expected delivery dates (Appendix B). Numerous meetings were held between research team members and improvement teams to work on the deliverables and assess progress. An interim status meeting for the steering committee was held on August 19, 2013, where a status update on the three projects was presented and progress was discussed. The project team members provided deliverables for review by October 19, 2013. This phase was closed on October 26 2013 a week after the first 60-day release went live.

**V.III.ii Content**

The execution of the improvement projects was started during this phase (Appendix A, Table 4.3-2). The new release cycle (Appendix F) helped achieve better product quality, as it engaged the quality assurance team to work early, gathering important information during design and development. This meant that the tests were more proficient, had better coverage, and resulted in fewer builds. The development manager was happy with the progress seen in the quality assurance team:

> *"I think we did a very good job, and you can tell that the quality did get improved. I mean we did do the regression test through some of that stuff and a lot of that made a real difference with the customers. I think even [business owner name] said that he didn't have a big issue with defects this time."*

Similarly, the business owner, who had always been critical of the product quality, had this to say:

> *"We are getting a lot better at QA."*

Weekly demonstrations conducted by the development team provided an early preview to the stakeholders of the new features as they were being developed, and it also provided an opportunity for the development team to receive early feedback (Table 11). The development manager felt the weekly demonstration was helping:

> *"I think they have been very helpful and the fact that I wasn't here last week is testament to how helpful they are, because it irritated people that I wasn't at the demo last week. You really don't know how important these things are until you miss one and people are irritated, so I can only say that the weekly demos are extremely help to QA and other stakeholders."*

Product management provided wireframes to the engineering team, for bigger features, which not only allowed for a clearer way to communicate the working of the new features, but

also provided a way for the product manager to develop a more informed and profound thinking process while the new feature was still being designed. According to the business owner:

> *"Improvements in the requirements management area made a huge difference. The wireframes help us stabilize and know what we are getting beforehand. A great example was a requirement that we did, we all thought we knew what we wanted. We wireframed it out, and when we had it in at the UAT, we realized it is not what we wanted. It wasn't fault of QA and it wasn't fault of development. It is just when you start clicking around it gets too confusing, and thankfully we could fix it before the release."*

As discussed, the product manager found the screenshots useful too:

> *"We are using wireframes as a rule in place, to provide more visual examples. So where possible, provide visuals, even above user stories. This is also about sharing of understanding of requirements. It's not only a question of getting them specified, but to reasonably specify them so the programmers and testers, and for that matter everyone, understand the same thing. So a picture is better than words."*

A requirements walkthrough by the product manager, for the documentation lead, engaged the documentation lead much earlier in the release cycle (Table 11). This change helped in three ways. Firstly, it made the product manager the center point of contact for the requirements, which was what stakeholders wanted to see. Secondly, it detached the development team and the documentation lead to a greater extent, which helped the development team to focus on the development tasks. And finally, the product manager and documentation lead contact ensured that the documentation was slanted more towards the customers' perspective. The development manager was happy with this setup, and he saw value in it:

> *"It would bring the escalations down because a lot of them (customer escalations) are about how the system works"*

Furthermore, the new release model (Appendix F) allowed holding retrospective meetings after each release, which provided a feedback mechanism to apply valuable learnings from the previous release cycle to the next. The business owner expressed his appreciation of the retrospective meetings:

*"I think one of the most valuable things that we have done during the improvement program, is holding the retrospective sessions that we have started after each release. It focuses on what worked and what did not work during the release, and we just did one, for example, one of the problems that we have is, who makes the release go/no-go decision? It used to be my decision, and then we said no we need to push this; the product manager needs to make a decision, so we changed this a little bit. That kind of a change is always going to happen, because we evolve."*

Moreover, TAMs, who kept their fingers on the pulse of customers (Appendix C) and try to cater to their needs on regular basis, suggested important product improvements to engineering, which the engineering team incorporated. By listening, learning, and responding to customers through TAMs, this interaction strengthened the engineering team's understanding of the customers' needs and expectations, which contributed to a successful release in October 2013 (Table 7). One of the new features incorporated into the release, based on TAM's input, was the ability for the customers to open support tickets and chat with an expert in the event of an issue. This feature was implemented under the customer self-service model, where the idea was that customers could be more self-sufficient in support release issues, and in the process, allow TAMs to focus on other priorities. According to the business owner:

*"These features freed up 15% of TAMs time. It allowed them to be more strategic with the customers, so they are not just supporting people. TAMs should spend less time on individuals working on the other systems and spend more time trying to make customers trying to increase the utilization of the service."*

The product manager was happy to see these new features implemented by the engineering team, despite challenges:

*"It's not perfect, but now customers can create tickets, they can use chat, all that stuff that we wanted to do is there in one full scoop. So I think we exceeded the expectations."*

One of the TAMs was thrilled to see a feature which he championed with the product manager and the engineering team:

> *"One of the biggest improvements that we have done is trying to capture all the data that is necessary on the scan form in the portal, and now there is no floating data. It will help build customer profiles to introduce more intelligence in future. I think we did a very good job with that. I would give it is a 9 out of 10."*

A significant observation during this phase was the mutual reinforcement of the three improvement projects (Appendix A, Table 4.3-2), where the activities of one project supported the deliverables of the other two projects. For example, the new release model (Appendix F) provided a fitting framework for the key deliverables of the other two projects. The requirements management and the quality improvement deliverables seamlessly corresponded with the new release model (Table 9). As an example, the new release model allowed the quality assurance team the additional time to introduce regression testing during the release cycle. Introducing regression testing was one of the key deliverables of the quality improvement project. Similarly, in accordance with the new release model, product managers could now start working on the requirements for the next release cycle much earlier. This helped the deliverables of the requirements improvement project, due to the additional time built into the new release model to account for the release readiness activities. Some of the deliverables for the customer improvement projects were also facilitated by the new release model, such as improving the customer prerelease communication by informing them of upcoming new features to customers three weeks before the release date (Table 7).

In addition, the synergies from the three improvement projects also helped instill a culture of continuous process improvement in the engineering team. For example, the engineering team felt that having a separate environment would allow a preview of new features to key business stakeholders, like TAMs, earlier in the release cycle, and hence, this would provide them with valuable feedback. Having early feedback during the release cycle was critical for a smooth

release, as it would significantly reduce the element of surprise to the business when the new version of the product was made available to customers. In addition, such a dedicated environment for "user acceptance testing" would provide a more stable platform for feedback, since a shared environment with other engineering teams could result in availability issues due to simultaneous engineering activities, which could disrupt business stakeholders during their preview activities. The development manager was delighted and relieved with this new setup:

> *"This release cycle has helped us in managing expectations with the business that has eased things tremendously, like in that previous release cycle, the business had different expectations and then the release day came and they were like what was this feature doing here, because they hadn't seen the new version earlier. Now we definitely solved that problem with the UAT environment and getting this out to the business a little bit sooner, the communication to the business is much better, and I do think this time we are on the same page with the business, mentally. We have ended things a little bit early now and giving stakeholders the visibility to the product earlier, which had a great impact to the business."*

The quality assurance lead had his own reasons to feel relieved with the new environment:

> *Due to the UAT environment, valuable feedback came from the product managers and the TAMs. Especially that the TAMs were able to jump in and do testing in the UAT environment. I mean, the TAMs were there constantly working on it, so it gave me a little feeling of comfort that my team wasn't the only pair of eyes to test it. And it made the live-chat feature one of the quietest features to be tested because they spent a long time working on it."*

Another process innovation instance was seen when the engineering team started tracking, and providing visibility about the final release activities. This was made possible by circulating the status of the release checklist to all stakeholders (Appendix D). This communication helped tie together all the teams involved in the release cycle and allowed a consolidated focus on the final release cycle activities. The business owner saw value in the release checklist:

> *"One of the things that we have done a good job of, is putting a defined schedule together for every release. So putting like a graph with a time limit that says here are all the*

*things that are going to happen, and we update it and send it out and it says get JAD done, get XYZ done, and get security testing done."*

The quality assurance lead found this release checklist very beneficial too:

*"I really like the release checklist with status as we near the release date. By sending everyone email showing when we are reaching specific goals, it is very helpful for me because this way I find which steps are affecting QA."*

Everyone was happy working with the new release model (Appendix F) as it allowed more time for requirement analysis, testing, documentation and customer communication. Below are some related quotes:

*"I like the 60 day cycles. It allows us to take bigger changes at once. We don't have to break up our changes into releasable components. I think if you get more than two months, you start to run into problems in that you have big giant branches that you can put back together. Two months seems like you are doing decent size work but you are not going crazy." - Development Engineer*

*"Going for two month cycles has been huge. It helped us get rid of a lot of chaos. I think it has made a major improvement on the quality of life of the staff because you know you can't run forever. Going into a fast walk, and going into a two month release is much easier for them." – Business Owner*

*"The move to sixty days proved to be a great one in my opinion." – Product manager*

*"I am really happy with going to the 60 day release cycle." – Quality assurance lead*

There were a few instances where passive change resistance was observed. One area of such a resistance was seen while improving the requirements management practices (Table 9), an area led by the product manager. The GSU research team sensed her passive resistance when she started missing her deadlines for the same deliverables multiple times. Specifically, her deliverables were to provide screenshots of the new features to the engineering team. These new features were included in the next release. It was not until the GSU research team interceded to provide her support in overcoming her resistance before she provided her deliverables. As a result, it caused a delay of a few weeks in her deliverables. The improvement changes required the product manager to come out of her comfort zone, as it required her driving the requirements

management tasks more actively with members of the engineering team and various people in business operations side. She had associated this change with the loss of her existing comfort zone, her skillset, and her prestige within the organization. She did not resist the change itself, so much as she resisted the uncertainties, fear and discomforts associated with it. However, once the GSU research team provided her support in delivering the requirements, it addressed her fear of the unknown.

Another area of resistance was observed when a stronger role of the TAMs started to emerge as a result of the improvement projects. Their role had strengthened because the key stakeholders realized the need to retain the existing software customers happy in order to continue to grow business with them (Table 7). Part of this realization stemmed from the requirement to place greater importance on incorporating the feedback from existing customers during the development of new product features (Appendix C). This made the input from the TAMs key in the development process. The shift to give more attention to the TAMs did not align favorably with business managers who wanted to focus on winning new customers. The business owner felt that the TAM's priorities would conflict with other business initiatives, since both required engineering resources. The business owner told us:

> *"In one of the releases TAMs were screaming because they wanted something, and I came in and say you don't need it."*

However, TAMs successfully backed their claims with strong data that showed there is a serious revenue upside in selling more services to existing customers. This helped some of the skeptical business managers to understand the business value behind the claims the TAMs were making, and they started to come to terms with TAM's stronger role, which focused on doing more business with existing customers. Below, is what the business owner shared with us at a

much later point, which showed how much Software Inc., benefited from the TAM's efforts to keep their customers happy in order to earn more business with them:

> *"We also get customers that we know are one-time customers, and they are like we want to buy single assessment I got an audit, I am not going to come back. We had several of these customers that had basically a 3 million dollar contract with us, but we didn't know if we will be renewing with them, but they went from 3 million to 11 million just to expand their coverage."*

**V.III.iii Context**

The mutual reinforcements from the three projects (Appendix A, Table 4.3-2) created a powerful thrust in moving the change (improving the release cycle processes) forward with significant force, like a powerful river flowing directly to open sea, letting nothing block its path. For example, it was getting difficult for the business owner and development teams to sustain their close interaction, as it would negatively interfere with the deliverables of the improvement projects, and personally, it would look embarrassingly awkward for them. In the new emerging reality, the release cycle actions were required to move forward through a formalized channel, respecting the roles of the functional teams. As a result, we saw a swift change in the team culture. The culture was now more relaxed.

Also during this phase, smoother relationships started to develop between various actors (Appendix A, Table 4.2-2). The development manager, who during the diagnosing phase saw little value from the product manager, had a changed perspective:

> *"The product manager's ownership of the requirements is pushing forward, and he has been very helpful. He has been running to us with these requirements, and he started pushing these requirements, and he is going to do more work… I do believe that the product manager is the one to count on to be the source of requirements and not those fifty sources for requirements, I think the right relationship is there now, and I do believe that he is someone that I can start to count on to be my source of requirements… I mean*

*we are starting to build up a little bit of trust there…I think he is starting to produce better documents for us to follow and it will make a difference."*

Similarly, the quality assurance lead was happier with his team's relationships with the development team:

*"I am happy with all that development has done in terms of getting better and better. The development manager is making good attempts in making good communications. We are having meetings outside of the regular 'QA-Dev' meetings if issues come up."*

The *Secure-on-Request* business was expanding, both in terms of new customers and revenue. During this phase, *Secure-on-Request* closed the biggest single deal in terms of money with an existing customer. In addition, there were also many more significant deals being made. As a result, the *Secure-on-Request* business was exceeding expectations in terms of revenue. This new wave of success of the *Secure-on-Request* software was largely due to customers seeing value in the product. Mostly, this value came in the form of the recent features added and changes made to the product by the engineering team, with guidance from product management team. As a result, the continued business success of *Secure-on-Request* was energizing the engineering team by making them feel that they were directly contributing to the success. This was creating an overall environment of pride and a spirit of, *"let's do even better."* The product manager attributed part of this success to the engineering by saying:

*"Well, I think we're as well-positioned as anyone in the industry, from a competitive perspective. I think part of it is our growth. We don't win every deal, but we beat our numbers revenue-wise, we won some very big deals, and in part because of our ability to turn pretty quickly on features and functions and requirements, and I think that a lot of folks feels that we're pretty nimble. I mean one customer said, 'I've been asking for this thing from your competitor for a year and you guys did it in, you know, two months.'"*

### V.IV Learning Phase

The learning phase began in October 2013 and ended in February 2014, when the initial GSU and Software Inc., collaboration ended (Appendix A, Table 4.5). During this phase, Software Inc., started to focus mostly on practicing software processes that the company had developed over the previous phases.

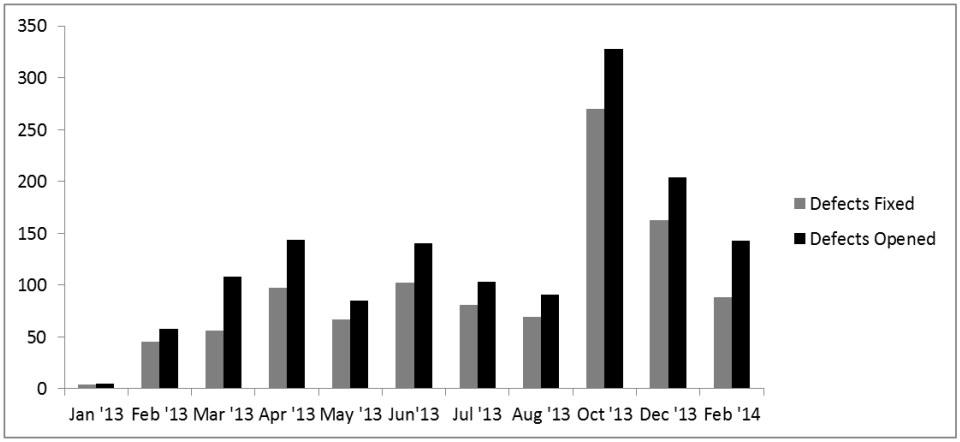
#### V.IV.i Process

Even though there was a distinct learning phase, through the IDEAL model (McFeeley, 1996), learning also happened during the whole research collaboration period. After the collaboration period, we evaluated the whole process and reviewed the proposed solutions, as well as the impacts of the three improvement projects (Appendix A, Table 4.3-2). We also carried out the practice-based assessment and reviewed performance data extracted from Software Inc.'s internal tracking systems. After reviewing the results and assessing the strategies that we used for the process improvement project, we interviewed several employees about the impact of the initiatives (Appendix A, Table 4.2-2). Having successfully introduced a number of improvements in the release cycle (Appendix A, Table 4.2-4), the team validated and analyzed what had been done. The GSU research team, the steering committee and the three improvement project teams (Appendix A, Table 4.3-2) had worked well together and had achieved initial success. The team effort had a reinforcing effect on establishing more effective release cycle processes.

**V.IV.ii Content**

There had been a positive impact on all aspects of the process as a result of the undertaking. Almost all of the targeted areas for improvements were successfully achieved. The biggest achievement was the transition to the 60-day release model (Appendix F). This model provided the *Secure-on-Request* functional teams a foundation to successfully manage the release cycle activities. Better software was being produced as defects were reduced (Figure 6) and tracked back to the source, allowing effective, preventive action to be taken to avoid reoccurrence.

**Figure 6: *Secure-on-Request* new and fixed defects trends**




It was determined that the effort had been an overall success, which led to improvements to the release cycle processes, as reflected in the October 2013 release, which was the first to benefit from the interventions. The product manager reflected on the October release:

> *"We just had the first release of it. We don't know yet what the full impact is, but it's been quiet. Quietness is goodness, typically, in this space. Because we released all these capabilities in October and we had nothing blow up. No one's calling saying, 'This is not working, we need help!...' Overall, we've achieved improvements in communication between the teams. I think we achieved improvement in overall quality across the board. We've got better defined requirements. We've stuck more rigidly to the schedules. And*

*then I think we did a better job on managing requirements and trying to provide a little more granularity with respect to the release plans."*

The business owner, reflecting on his thoughts about the final outcome of the improvement projects, stated:

*"I am thrilled in terms of achievements. We have achieved five times more than I thought we would achieve. I think the outcome has been extremely positive, you know. I don't know about anything that we didn't achieve, and I didn't think we wouldn't achieve, so I don't know about anything that we missed during the process."*

The quality assurance lead was excited about the overall outcome:

*"Overall, I keep my expectations lower around here, but I was kind of surprise that some of the things ended up working out very well. I mean, in general I am happy. So in my expectation I kept them low, but I am pleasantly surprised."*

Some of the improvements started in the October 2013 release would require more time to be further optimized during future release cycles, as those improvement areas are closely related to the cultural aspects of the team members. According to the product manager:

*"There's still the kind of interactions that happen outside the process that we're trying to work on where we ask 'why do we do that?' 'Well, so-and- so called me, and we needed to go do it.' This still happens to some degree, but I think that overall it's improved a lot. And we need to keep improving it by holding each other accountable."*

The only area of improvement which could not be satisfactory met was the introduction of automated testing, due to organizational priority issues (Appendix A, Table 4.2-4). However, the team did plan to follow up on this improvement during the future release cycles.

### V.IV.iii Context

In general, the people we interviewed considered the improvement program very successful. One common theme was a heavy emphasis on the continuous process improvement going forward. There was a commitment to an organizational culture of continuous awareness, responsiveness to feedback, and openness to improvement. When the *Secure-on-Request* team

knew how to improve, then they would improve. During the improvement program, we made changes, observed effects, and started to digest the change. Now, Software Inc., needed to turn this pattern into a habit. If they did, they would hit a plateau from which they could absorb more feedback and identify new opportunities. This is how the project manager reflected on it:

> *"My biggest concern is we slip back into bad habits. It's still hard. We've still got to keep doing it. It's hard to keep it going. It's like that huge wheel, that rock that grinds the wheat, you've got to keep pushing that thing because it doesn't just keep spinning. We implemented four or five major improvements, but that's not the pinnacle of where you want to be, and so, as we move forward, we have to keep getting crisper on our timeframes, on our requirements, on our release plans, and so on."*

On a similar note the business owner reflected:

> *"Now it's not perfect. We can still continue to improve that. Make sure we don't fall off the wagon. So it's like, you know, alcoholics are still alcoholics, they have to make sure they stay on the wagon."*

Another important area for learning was the engineering team's ability to strike a delicate balance between working on product features, which was tied to business revenue, and working on improving the product maturity by being focused on improving the technical architecture of the software. This balance played a key role towards the outcome of the improvement projects. The general tendency from the engineering teams is to be more biased towards doing more technical changes, which creates some level of tension with the business teams. However, according to the business owner:

> *"There is no pressure from the development manager to do that. I think he understands that business revenues are more important sometimes, so he gets a person to balance it, but he also knows how to scream if there is like, 'Hey we are getting into a problem here,' so I don't feel like we have a problem balancing."*

The business owner was a very powerful person in the organization. His leadership style would heavily influence on the organizational culture. When asked about his personal learnings from the improvement program, his response was:

> *"One of the personal learnings is that we have to take more time to stop and think how we are doing things. We need to stop and say, great let's think about the next 45 days because that is like our next focus, and think about the last quarter and about what we wanted to achieve in it and we didn't achieve, and next quarter what you want to achieve and how you want to achieve."*

One process improvement specialist, the author, is still employed full-time by the company, as the process improvement specialist. Moving forward this role will be focused on driving the continuous processes improvements in the organization, as referred to above, to build on the improvement work started in this research collaboration.

In conclusion, the findings in this chapter revealed how Software Inc. adopted the release cycle management (RCM) approach that organized and improved release processes to help recurrently develop and deliver *Secure-on-Request* to its customers. The RCM approach provided a holistic view of the *Secure-on-Request* release, focusing on how all the moving parts fitted together in effectively managing the release cycle, and how it provided a basis of continuous SPI at Software Inc. This chapter presented an important basis to thoroughly understand RCM. As a result, RCM at Software Inc. made a compelling case in managing releases, as well as providing an efficient approach for SPI in recurrent software development. The concept of RCM will be further elaborated upon and precisely defined in the next chapter.

# DISCUSSION

As discussed earlier, a review of the literature found no studies that empirically examined the implementation of SPI into software release processes in the recurrent development of software. Moreover, we found an emerging literature that looks at the release processes with a holistic perspective. However, there is no literature with such a viewpoint of release processes in the recurrent development of software. In an effort to address these gaps, the action research at Software Inc., drew upon Pettigrew's contextualist inquiry (1987 & 1990) and uses the IDEAL cycle (McFeeley, 1996) for improvement. The research explored the holistic concept of software release, focusing on how all the moving parts fit together to effectively manage a release cycle, and how the new release concept can provide a basis of continuous SPI in the recurrent development of software. In the following pages, we discuss organization and improvement of RCM at Software Inc., and the subsequent contributions to theory based on a grounded model of RCM in recurrent software development.

## VI.I Improving Release Cycle Management at Software Inc.

### VI.I.i Release Cycle Management

By applying the concept of RCM, we helped Software Inc. take an approach that focused on release as a common thread running through strategic planning, execution, delivery and operations processes, each of which aimed to manage the success of the release by imposing governance over its evolution (Tarboda, 2012). Therefore, as the release transitioned through a series of states in a cycle (Appendix F), the successive release milestones became shared goals that brought together the different perspectives of managers, developers, testers, TAMs, and end

users (Appendix A, Table 4.2-2). This approach seamlessly linked the entire array of software engineering processes, synthesizing and capturing the essence of the release in a manner that was relevant and appropriate to the stakeholders during a release cycle (Table 11). Creating highly cohesive and integrated processes proved to be the key to smoothly move forward the software delivery to their customers (Table 7).

**VI.I.ii Problem Solving**

Before our research and intervention at Software Inc., monthly releases did not allow enough time for requirements analysis, testing, documentation and customer communication (Appendix A, Table 4.2-3). Now, the new release model (Appendix F) has allowed the required time for these activities. Another problem that we encountered was that requirements were unclear, causing confusion and additional work (Table 9). Requirements prioritization within and between new features were a major challenge. Now, requirement prioritization takes place through a structured process involving a series of meetings between product manager and stakeholders (Table 11). Mockups, feature design meetings and weekly demonstration sessions are also now conducted by the development team to ensure requirements specification processes are effective (Appendix A, Table 4.2-4). Previously, the product would be released with poor quality due to minimal testing, causing a surge of customer escalations immediately after the release. Now, more time is available for quality assurance activities to run software testing. Furthermore, moving the user acceptance milestone to an earlier time period in the release cycle, allowed stakeholders to access the software well in advance before the formal release date (Appendix F). It also allowed stakeholders to provide more timely feedback about the software quality to the engineering team. Before our interventions, documentation was a real challenge

(Appendix A, Table 4.2-3). During the monthly release cycles, the development team was writing new code until very late for the documentation to be completed in a timely fashion. Under the new model, the documentation activity is started early in the release cycle, and the documentation team now works directly with the product manager through requirements. This new arrangement allows feature development and documentation processes to be carried out in parallel, whereas, they were previously conducted sequentially. Before our interventions, the product management team did not have a good understanding of the customers' expectations and needs. Now, due to the creation of a customer advisory board (Appendix C), the product management team has a better understanding of customers' expectations and needs (Table 7). Finally, in the old release model, the software code was still being changed very late in the cycle. As a result, the final release content was often unknown until the very end. Therefore, the customers could not be communicated to, well in advance, about the content of the new release. Now, due to the introduction of the feature-freeze milestone in the release cycle (Appendix F), the content of the outgoing release is known well in advance of the release date, allowing timely communication to customers, listing the features to be included in the new version of the software.

**VI.I.iii Continuous Improvement**

Our post-implementation interview laid the foundation for continuous improvement by revealing that our interventions set the groundwork for the *Secure-on-Request* team to achieve future collective accomplishments (Table 12; Appendix A, Table 4.2-2). The *Secure-on-Request* team has learned how to use rudimentary tools for sustaining change. Although the GSU research team has handed over the baton to have the team at Software Inc. carry it forward for

the next lap, sustaining the positive results of the interventions is their next challenge. Furthermore, the managers want to establish RCM KPIs for three levels of monitoring and reporting: executive, manager, and operations. Finally, learnings stimulated by the interventions are still emerging, and it is not well defined how new RCM practices (Appendix F) are identified, presented, and diffused across the organization. As the product manager stated: *"I think you have to work harder at it (sustaining the change) earlier on. So, you probably know after two or three cycles that it becomes a second nature. But if you start to take it (change for improvement) for granted, it will become lax."*

**VI.I.iv Software Quality**

Figure 6 provides relevant indicators of quality for *Secure-on-Request* releases since April 2013. The spike of new defects opened during the October 2013 release cycle (the first release made after closing of the IDEAL cycle (McFeeley, 1996) at Software Inc.) reflects that the longer release model (Appendix F) provided a more open situation for the quality assurance team to report new defects and acknowledge development's fixing of those defects during a cycle. Furthermore, over time fewer and fewer new defects were reported which indicated stabilization of the software. This is represented by the declining trend in the number of new defects opened (Figure 6) since the October 2013 release. In addition, we also see from the figure that the development team consistently fixed a significant portion of all reported defects during each release cycle after the interventions. As a result, it contributed to significantly improving the quality of *Secure-on-Request*, a view also confirmed by different stakeholders in their post-intervention interviews.

### VI.I.v Stakeholder Assessment

To learn how the new RCM was perceived at Software Inc., we conducted post implementation interviews (Appendix A, Table 4.2-2). The interviews revealed that the new requirement practice was effective (Table 9), quality was improving (Figure 6), the new release cycle (Appendix F) made the chaotic culture more relaxed, customers believed the product team met their request for new features with agility (Appendix C), the engineering team was doing a great job balancing technical debt and working on the new features in the software. Engaging the documentation team earlier in the release was helping to effectively capture software information about the new features in the release, and, the quality and timeliness of prerelease communication to the customers was also improving (Table 7).

## VI.II Release Cycle Management in Recurrent Software Development

Our study contributes to the existing body of knowledge by providing new insights into the area of recurrent software development, software release practices and SPI. Specifically, the study adopted an analytical lens based on Pettigrew's contextualist inquiry (1987 & 1990) to explore how RCM can be organized to facilitate SPI in the recurrent development of software.

The literature has been focused on software release as an isolated activity (Ballintijn, 2005; Carlshamre, 2002; Gaur & Oberoi, 2012; Mazlan, Sefat, Selan & Lukose, 2013; Qian, Yao & Khoshgoftaar, 2010; Regnell & Kuchcinski, 2011; Ruhe & Saliu, 2005; Scott & Nisse, 2001; Svahnberg et al., 2010; Van Der Hoek, Hall, Heimbigner, & Wolf, 1997). This dissertation focuses on release practices in recurrent software development as a common thread, running

through the stages of strategic planning, execution, delivery and operations, with the aim of managing the success of a release and imposing governance over its evolution.

Furthermore, software releases in recurrent development represent a continuous process of planning, monitoring, improving and communicating of software engineering activities within and between target releases. RCM provides a powerful way to identify and apply the required changes to development and management activities to improve the software. SPI normative models are often criticized for the inflexibility of their pre-defined actions and their underlying deterministic assumptions about implementation (Allison & Merali, 2007; Bollinger & McGowan, 1991; Kohoutek, 1996; Mathiassen & Sorensen, 1996; Pries-Heje & Baskerville, 1999; Velden et al., 1996). To address these concerns, the challenge is to understand change driven by SPI, not as a predictable or designed causal outcome, but as an emergent process developed from the relationship between people and their context (Allison & Merali, 2007). An emergent view of SPI helps to understand the way the actions intertwine to inform each other and how they are shaped by their context. While the literature recognizes the emergent nature of software development practice (Mathiassen, 1998; Truex, Baskerville & Klein, 1999), the dynamics of emergence is under-explored (Allison & Merali, 2007). This dissertation contributes to the literature by using a contextualist lens to elucidate the dynamics of emergence by integrating SPI into RCM.

Based on an analysis of our collaboration with Software Inc., our study adds to existing knowledge by extending our current understanding of release cycle processes in recurrent development environments. As explained below, our study focuses on the role of RCM in realizing SPI in such environments.

First, we offer new insights into the contextual characteristics of recurrent software development literature (Carmel & Becker, 1995; Colomo-Palacios, Soto-Acosta, García-Peñalvo & García-Crespo, 2012; Ncube, Oberndorf, Kark, 2008; Sawyer, 2000; Xu & Brinkkemper, 2007). By referring to a contextualist perspective (Pettigrew, 1985, 1987), this action research investigated the challenges around the recurrent development of *Secure-on-Request* by focusing on how releases of such software are managed and on how process improvement can be supportive. In the process, our study revealed how the articulations between the different contextual elements continually unfolded and built up again through the complex interplay between actors in the *Secure-on-Request* unit.

The contextual challenges from within the *Secure-on-Request* unit (inner context challenges), included recent acquisition of additional software, complexity of service delivery, new engineering and product management teams, low software development process maturity. Outer contextual challenges included the commercial pressure that shaped their RCM processes. In the midst of these challenges, through our action research, we saw that the software processes at Software Inc., were enacted through a constant process of negotiation between the engineers, TAMs, and the managers. The research provided rich insights into how the different competencies, characteristics and experiences of the software team shaped their actions for improvement and how those actions reinforced and altered the context at all levels. Hence, the interplay between the "content" and the "context" in this action research added rich insights to the literature by focusing on how the *Secure-on-Request* release was managed and on how process improvement supported it.

Second, based on the analyses of our collaboration with Software Inc., our study adds to our current understanding of release by focusing on how all the moving parts in recurrent development of software fit together. The holistic view of the release is discussed in the literature (Humble & Farley, 2010; Taborda, 2012). Expanding on this research, our study applies the holistic perspective of the release within recurrent software development. We combined a contextualist inquiry framework (Pettigrew, 1985, 1987) and an IDEAL cycle approach (McFeeley, 1996) to closely analyze how the holistic view of release at Software Inc., helped improve their software processes.

We saw at Software Inc., that it was the significance of the delivery milestone that provided the impetus to understand how the release came into being. As the release went through different states, our contextualist (Pettigrew, 1987 & 1990) inquiry helped us to understand how the successive release milestones became shared goals that brought together the different perspectives of managers, developers, testers, and TAMs. This shift in release perspective, in turn, benefited both the *Secure-on-Request* unit and its customers.

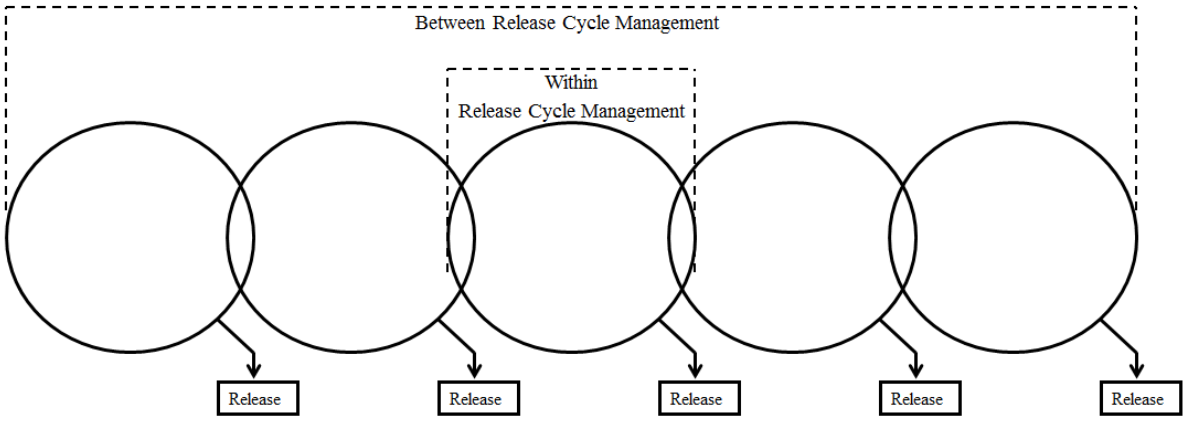
Third, software RCM is a continuous process of planning, monitoring, improving and communicating software engineering activities between the target releases. SPI literature acknowledges that the activities of developing and delivering software and improving the related processes, are not mutually exclusive, even though they are normally considered as separate (Allison & Merali, 2007). Our study adds to existing research by explicating how well-organized RCM can provide powerful means to identify and apply changes for improvement to the development and management activities, in order to recurrently deliver software to existing customers and the market.

At Software Inc., when someone saw a clear purpose in introducing a new technique, for example the building of a dedicated user acceptance testing (UAT) environment for stakeholders, or revising a current method, for example the documentation lead started working directly with the product manager instead of the development manager, they were prepared to apply their own resources and the team's resources to its introduction. Hence, improvements occurred through the ongoing practice and improvisations of the practitioners as they identified and sought to solve perceived problems, or as they found and shaped an external solution to solve a problem previously identified. Therefore, RCM can naturally blend with SPI, forming a very powerful phenomenon to continuously focus on developing software and improving the relevant processes.

Finally, we offer a conceptual understanding of RCM and how it unfolds over time (Figure 7). In a setting of recurrent development (Carmel & Becker, 1995; Colomo-Palacios, Soto-Acosta, García-Peñalvo & García-Crespo, 2012; Ncube, Oberndorf, Kark, 2008; Sawyer, 2000; Xu & Brinkkemper, 2007), because of its repetitive occurrence, a release ceases to be a milestone and instead takes on more of a process focus. The interpretation of a release as a process that guides the development of a work-in-progress creates a new awareness of the release as a relevant management construct across the entire product life cycle (Taborda, 2012). This makes the conventional release management a more regular activity, requiring its own processes associated with the continual incremental delivery of evolving software. Based on this notion, this dissertation contributes to literature by developing a new release paradigm in the recurrent development of software called release cycle management (RCM), which can be defined as:

*Software release cycle management is a continuous process of planning, monitoring, improving and communicating of software engineering activities and its organization, within and between the target releases, where release priorities need constant adjustment based on the learnings in the cycle and changing business strategies.*

**Figure 7: Conceptual Representation of Release Cycle Management (RCM) in Recurrent Software Development**




Between target release cycles, RCM also provides a strong basis for incremental improvements in software processes, which are continuous, concerted and cumulative.

In conclusion, our analyses suggests that our Pettigrew's contextualist inquiry (1987 & 1990) offered a powerful approach, to clarify the software improvement process in a recurrent software development environment, and also to expand knowledge as it relates to RCM.

## VI.III A Grounded Model of Release Cycle Management

The emerging research on release management highlights a new trend within software engineering. In a first study, Louis Taborda (2012) presents a new paradigm for software releases for evolving businesses. He refers to this paradigm as enterprise release management (ERM).

Through this paradigm, he takes a holistic view of change that offers a synthesis of traditional management approaches, including project and change management, enterprise architecture, and development practices like configuration and release management. His study establishes an end-to-end release framework that ensures initiatives are planned and prioritized to streamline portfolio execution and delivery. Benefits of this release-centric approach include reduced execution and operational risk, improved demand management and optimized release throughput. Taborda's study offers a fresh enterprise perspective that addresses strategic change and the release life cycle, providing managers with the tools they need to chart and track the course of their business.

Similarly, in a second study, Jez Humble and David Farley (2010) lay out a detailed concept of release pipelines, in a holistic sense, in their study on improving release management. Through their concept of pipelines, they present a pattern that can be implemented to model an end-to-end path to the release of software. They summarize this pipeline as: "in essence, an automated implementation of your application's build, deploy, test, and release processes." Hence, this emerging line of research provides valuable insights by recognizing release as the common thread running through strategic planning, execution, delivery and operations, where each process aims to manage the success of the release and impose governance over its evolution.

In contrast, this study provides insights into how RCM allows software practitioners to integrate a number of familiar software engineering management disciplines into a holistic view in a recurrent software development setting. The richness of the release concept across the recurrent software product development is at the heart of RCM that taps the concept's emerging

relevance to management. It is the goal of RCM to attempt a unification of different viewpoints about release and to assemble a multifaceted understanding of the release from the point that it is first identified and defined, as part of strategic planning, to its ultimate realization as a solution that delivers the promised benefits.

Accordingly, our study advances the discussion of SPI by revealing how RCM can be used as a vehicle to achieve continuous process improvements in the recurrent development of software. Considering the importance of the contextual factors, we adopted Pettigrew's contextualist inquiry (1987 & 1990) as an analytical perspective to make sense of the rich data from Software Inc., during the problem solving cycle. As a result, we developed a detailed account of how software engineering practices were improved by focusing on the RCM processes. Pettigrew's contextualist inquiry (1987 & 1990) provided insights into how various engineering practices were improved by focusing on the RCM processes. Specifically, Pettigrew's contextualist approach (1987 & 1990) helped us to identify the contextual factors playing out during the improvement projects at Software Inc. This theoretically informed analysis revealed the underlying approaches, tensions and intricacies involved at the various stages of the improvement phases.

Based on the empirical account presented in this dissertation, we offer a grounded process model of how the RCM processes were organized and improved at Software Inc., (Figure 8 & Table 14). This model reveals how software RCM is a continuous process of planning, monitoring, improving and communicating software engineering activities, within and between target releases. Based on the learnings gained in changes to both the cycle and the business

strategy, priorities need constant adjustment, as software RCM creates a continuous and cumulative process of incremental improvement.

Below we provide brief details of the roles involved in this model:

1) *Business Executive*: This role sponsors the recurrent releases. The person in this role looks for releases to deliver business value and expects to avoid frustrating delays that impede the progress of the business strategies.
2) *Product Manager*: This role collects and analyzes requirements to flesh out the software strategy and drive the solution design. This role is responsible for aligning business and technology strategies and identifying alternative solutions, while ensuring business and technology impacts are understood across the increasingly complex and interdependent contexts.
3) *Operations Staff*: These roles are responsible for maintaining smooth operations in a software engineering unit by providing the necessary support functions to the core roles of software engineering.
4) *Engineer*: This role applies the principles of engineering to the design, development, maintenance, testing, and evaluation of the software.
5) *Manager*: This role applies the technological problem-solving skills of engineering, combined with the organizational, administrative, and planning abilities of management in order to deliver partially or completely recurrent releases from conception to completion.

6) *Business Operations Staff*: These roles interface with the engineering software unit from *Outside Context* with business operational perspectives for the software to be successful in market.
7) *Customer*: This role purchases the software (or the services derived from the software) and/or is the user of the software (or the services derived from the software).
8) In addition to the above roles, through the argument presented in this dissertation, a new role in a recurrent development of software unit emerges, which we will refer to as the *Release Cycle Manager*. Traditional software engineering units have roles like project managers and dedicated roles for SPI. The *Release Cycle Manager* role can substitute these roles in recurrent development. When recurrent releases are seen from the broader perspective, as argued in this dissertation, a single role needs to lead and oversee the improvement processes. Such a role can be the orchestrator of the release cycle processes. In short, this role is the go-to person. As a project has a definite start and end date, unlike the recurrent development of software, a traditional project manager role will not be suitable to manage such releases. Hence, the more effective role of the *Release Cycle Manager* emerges.

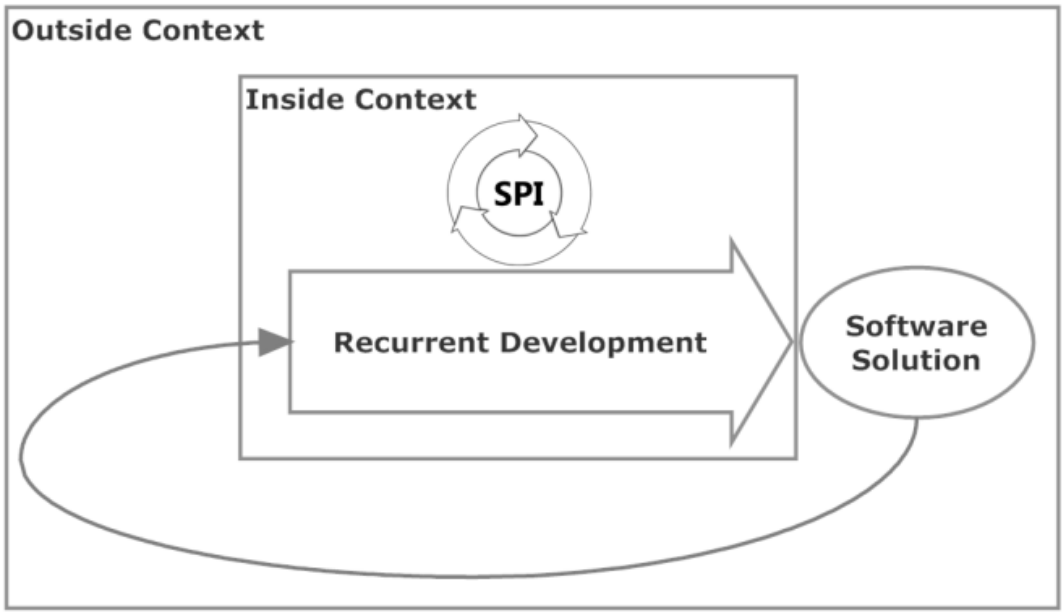
In retrospect, based on the empirical accounts of our analysis and previous literature, our grounded model presents software development as being conducted in a recurrent fashion, where the same product or service is continuously being developed as a consequence of updates and feedback from customers, defects in the previous release cycle(s), market factors, new customer demands and other technical and non-technical requirements. The model focuses on how releases of such software are managed and how SPI is an emergent and integrated activity in

such a setting. The contextualist perspective in the model takes into account the sensitivity to the environment and helps highlight the particular characteristics of recurrent software. Therefore, the model has a built-in systematic inquiry into the context (outside context and inside context), content (recurrent development of software), and process (SPI as an integral part of RCM), which optimizes the RCM and the process improvement in the recurrent development of software.

**Table 14: A Grounded Model of Release Cycle Management (RCM) in Recurrent Software Development**

| Entities | Sub- entities | | Description | Application from Software Inc. | Roles Involved |
|---|---|---|---|---|---|
| **SPI** | - Improvement Goals<br>- Improvement Organization (formal, informal) | | A continuous process to improve software development by leveraging the ongoing recurrent development processes | RCM (also) acting as SPI | - Managers<br>- Engineers<br>- Business Executives<br>- Customers<br>- Release Cycle Manager |
| **Context** | Outside Context | - Social<br>- Political<br>- Economical<br>- Competitive Environment | The inside and outside environment in which the software unit operates | Environment around *Secure-on-Request* unit | - Managers<br>- Engineers<br>- Business Executives<br>- Support Staff<br>- Customers<br>- Operations Staff<br>- Release Cycle Manager |
| | Inside Context | - Structure<br>- Corporate Culture<br>- Political Context<br>- People | | | |
| **Recurrent Development** | - Release Activities (Requirements, DEV, QA, Documentation, Demonstrations, User Acceptance, Delivery)<br>- Release Management (Planning, Monitoring, Adaptation, Communication)<br>- Release Organization (Roles, Technology, Structure)<br>- Release Frequency<br>- Engineering Services | | The same unit recurrently produces incremental versions for the market | Software engineering and management processes in *Secure-on-Request* unit | - Managers<br>- Engineers<br>- Business Executives<br>- Operations Staff<br>- Release Cycle Manager |
| **Software Solution** | - Software Characteristics<br>- Business Strategy<br>- Support Activities | | A software-intensive arrangement satisfying the specific needs of a particular market segment | *Secure-on-Request* | - Business Executives<br>- Business Operations Staff<br>- Customers<br>- Release Cycle Manager |

**Figure 8: A Grounded Model of Release Cycle Management (RCM) in Recurrent Software Development**




In the conclusion, we will elaborate on some methodological observations about the model as a whole, in terms of its significance and value for further research. Firstly, given the wide variation of the entities within the model, we expect that different researchers will find it useful in different ways, depending on their research questions and research objectives. Secondly, we should note that once we adopt the overall contextualist perspective, the entities within the model emerge easily, based simply on our familiarity. Finally, while the ultimate value of this research approach lies in what the analyses tell us regarding the value of RCM in the recurrent development of software, we must not forget that the model in Figure 8 and Table 14 represents a valuable result, from the use of the contextualist approach.

We believe our research is unique and is a valuable contribution, in particular, as we were unable to identify any other major efforts within the existing software literature that attempt to create a comprehensive model of RCM in the recurrent development of software.

# CONCLUSION

Drawing upon Pettigrew's contextualist inquiry (1987 & 1990) through an action research at Software Inc., we addressed the challenges the company faced in managing releases and organizing SPI to help recurrently develop and deliver a specific product, *Secure-on-Request*. The study has brought to light a number of interesting insights to theory:

1) We offer detailed insights into the specific characteristics of recurrent software development.
2) We suggest RCM as a comprehensive framework for understanding and managing recurrent software development.
3) We demonstrate how SPI may be integrated into RCM to support recurrent software development.
4) We offer a conceptual understanding of RCM and how it unfolds over time.
5) We provide a grounded model of RCM by focusing on how releases of such software are managed and how process improvement can be supportive, through a contextualist approach.

As a testament to the robustness of our action study and the approach we used, stakeholders at Software Inc. have reported that our intervention directly resulted in improvements to the release cycle processes. The resulting insights may have significant implications for both academics and practitioners alike.

There are, however, important limitations of the proposed approach. Most importantly, the research was carried out within one particular organization, Software Inc. Therefore, before

adopting the contextual approach suggested in this study, software managers should carefully consider the conditions that shaped our investigations. If these conditions are considerably different in their own organization, managers should consider alternative approaches. If sufficiently similar conditions exist, we encourage them to adapt the proposed approach. This requires careful consideration of the specific software engineering practices they intend to improve. We have not suggested a procedure to be followed. Rather, we have outlined how the content, context, and process of implementing RCM may be approached, supported by concepts and frameworks from the literature and lessons from Software Inc. Hence, effective adoption of this contextual approach requires appreciation of the specific history of the software engineering processes and careful examination of the experiences, skills, and other resources available to improve its benefits. Accordingly, we propose the following lessons for managers:

Lesson 1: Organize the initial assessment to improve existing software engineering practices based on the IDEAL framework (McFeeley, 1996). The IDEAL-based process allowed us to be flexible and adapt to emerging issues and events at Software Inc. It helped us to gain a rich understanding of software engineering practices through triangulation of data from different sources and analyses. Finally, the IDEAL framework's (McFeeley, 1996) focus on diagnoses, learning and active involvement of key stakeholders helped us understand the problems at Software Inc., rather than promoting solutions based on general models of best practices. In this way, we relied extensively on stakeholders within the organization, and our contextual inquiry was, in this sense, problem-based rather than model-based.

Lesson 2: View release holistically. At Software Inc. a release-centric approach helped us capture the essence of the release in a manner that was relevant and appropriate to the

stakeholders across the recurrent software development cycle. For example, the conventional focus of the release on the sharp end of delivery was changed to place greater emphasis on the early stages of a release life cycle. Instead of simply managing the technical integrity of the final solution, release management increasingly encompassed earlier phases, such as release planning, where requirements were prioritized and assigned to current and future planned releases. For this reason, we suggest recurrent development of software units to seriously consider benefiting from the role of the Release Cycle Manager who can oversee all of the activities in the suggested unified fashion during a release cycle under the same umbrella.

Lesson 3: Design improvement through planned and adaptive change. The process of improvement needed to account for reactive, reflective changes when the processes were to be improved, not just extemporized. We promoted sustainable development of the processes by integrating the experiences of the developers, their learning through action, and also through the sharing of that learning experience. The learning processes that informed the SPI activity were ongoing, not simply delivered via training. It was when a need was clearly answered, often serendipitously, within a training event that it was incorporated into the practice. Changes in the process-in-use at Software Inc., were seen to occur through different forms of innovation. Finding a way to facilitate this level of inventiveness within the software process is an important lesson learned from this study.

Lesson 4: Linking SPI to business objectives. The interventions at Software Inc., were not coupled with the business objectives. Indirectly the objectives were taken into account through the software management team's awareness of the business priorities. However, to have identified specific business goals, for instance, to reduce the cost of reuse, would have enabled

the tasks to be better aligned to these goals, and the benefits of the SPI project would have been evident to the steering committee. At Software Inc., the sales continued to grow, and their market leadership was strengthened. To support this market-oriented perspective, we needed to develop an agile approach to SPI through RCM so that the process improvement reflects the needs of the given context. An agile approach to SPI would be responsive and flexible to local needs, and it would encourage innovation in the process, build SPI innovation around those who are motivated, encourage self-organizing competent teams, and promote sustainable development of the processes.

Lesson 5: Ensure commitment and active participation on all levels of software management. The strong commitment and active participation of the business owner was instrumental during this process. Also, his engagement helped our collaborative relationships to managers, engineers, and TAMs. These relationships at Software Inc., played a major role in identifying new software engineering approaches and in implementing the new program as an integral part of the management and organizational context.

SPI has been well researched, but perceived challenges persist. In terms of further research, our study demonstrates a fresh understanding of process improvement through RCM. From this theoretical perspective, it is anticipated that a more agile and blended view of SPI with day-to-day software engineering processes is required if organizations are to leverage the emergent nature of the process improvement activity. Continued efforts could validate and further develop the proposed contextual approach to improve software engineering processes through RCM in settings that are different from the one at Software Inc., for example, in smaller software

organizations or software organizations that are have been in existence for a longer period of time.

By combining the real-world experience of those involved at every process of recurrent software release with academic concepts and frameworks, this study has closely followed the principles of "engaged scholarship." As Van de Ven (2007. p.9) states, engaged scholarship is: "a participative form of research for obtaining the different perspectives of key stakeholders (researchers, users, clients, sponsors, and practitioners) in studying complex problems." A singular sphere of knowledge alone, from within a software development organization would not have provided the required depth of knowledge to effectively examine the release cycle processes in the *Secure-on-Request* unit at Software Inc., and to recommend solutions. Through a commitment to an engaged scholarship model, this comprehensive study across every process, from conception to final delivery, has provided the deep, multi-dimensional knowledge needed to provide a unique understanding of the problems and solutions in recurrent software release cycles.

# APPENDIX A: SHARED PLATFORM DOCUMENT

## Improving Processes and Services in a Software Unit: An Action Research Study into Release Cycle Management

**Neda Barqawi and Kamran Syed**

**J. Mack Robinson College of Business**
**Georgia State University**

# A1.0 PROBLEM SETTING

As part of its corporate business strategy, Software Inc. has decided to develop and reposition its on-line security testing solution, *Secure-on-Request*. This Software-as-a-Service (SaaS) application enables an organization to test the security of its software quickly, accurately, affordably, and without installing additional software. This action research investigated the challenges around the recurrent release management and the continuous service delivery functions of *Secure-on-Request* at Software Inc. The release management team of the application faces four significant problems: (1) the recent acquisition of the software; (2) the complexity of service delivery; (3) a new engineering and product management team; and (4) software engineering process immaturity.

## A1.1 Recently Acquired Software

Software Inc. inherited *Secure-on-Request* through a recent acquisition. The company plans to develop and reposition this SaaS to realize its full potential. There were issues with *Secure-on-Request* stemming from before the acquisition: the original design needed rethinking, parts of the system were difficult to use, and the system's use of resources was less than optimal. Overall, the software is complicated, and its components need better alignment and consistency. As a result, the SaaS is somewhat fragile and until recently, the engineering team would not modify its core. Instead, they built everything around it for new functionality, and consequently the advancement of *Secure-on-Request* has been severely limited.

This innovation challenge is a predicament for the production group. The group is facing difficult to manage technology at a time when Software Inc. faces serious challenges from startup companies that threaten its market position with new, innovative technology. In this situation, Software Inc. needs to find ways to respond to customer needs and market demands as quickly as its smaller competitors. The company's best option is to adopt more agile approaches and business technology systems that respond nimbly to both changing market conditions and competitive challenges.

"Security testing as a service is a way for enterprises to reduce upfront costs and to augment limited internal resources when undertaking a software security program. This technology area is growing and will have a significant impact on the application security market over the next 12-18 months." — Joseph Feiman, Ph.D., Research Vice President and Gartner Fellow

### A1.2 Complexity of Service Delivery

Secure-on-Demand is a complex, SaaS-based security-testing solution. Each customer application submitted for security analysis is unique. A team of experts conducts a thorough audit of each application for security vulnerabilities and provides a comprehensive and accurate analysis. This service tests a variety of technologies (21 different development languages) for back-end, web, mobile or cloud-based applications. It encompasses the testing of thousands of applications, security expert teams located on

four continents, services provided to sixteen diverse industries including civilian and defense agencies, and companies of various sizes.

### A1.3 New Engineering and Product Management Team

Due to the repositioning of *Secure-on-Request*, Software Inc. has formed several new teams to support the recurrent release of the software. These teams, each with a specific function, include engineering development, quality assurance, product management, program management, and infrastructure operations. These functional teams are heterogeneous with unique skills and knowledge. Across these teams, there are disparities in commitment due to competing priorities. In this complex organizational set-up, the newly formed teams face two critical issues: establishing appropriate collaboration patterns and effective processes, and developing the capability to recurrently release new versions of the SaaS to market.

### A1.4 Low Software Engineering Process Maturity

Processes for recurrent release-management and related activities are mostly ad hoc. On the whole, software development is performed informally without proper documentation. As a result, the release-management function does not operate in a repeatable fashion. Due to this less than optimal software-development lifecycle maturity, the release-management team must work overtime to meet set deadlines and customer expectations. There are some mature tracking mechanisms and defined standards in

place. However, quality issues are mainly addressed by individual team members that are technically strong and experienced. As a result, the degree of predictability in schedule, budget, scope and quality is not high and the success of a release depends upon the heroism of a few key team members. Moreover, because there are no effective mechanisms for organizational learning, the know-how of the software can easily be lost if an engineer leaves the company.

### A1.5 Actors

The key functional leaders associated with this challenging situation include the head of the program management office, the development manager, the product manager and the business owner of the services provided by the application. Each of these people faces different but overlapping problems.

The head of the program management office is frustrated by the low visibility, weak predictability, and inefficient processes in delivering quality software to the market. He believes that these problems make it difficult to quickly and flexibly respond to problems and address the needs of end-users. Fluctuating and conflicting requirements is a problem for the development manager. The business owner of the service delivery of the software application is unhappy with the quality and the speed at which solutions are being delivered. The product manager feels he is sucked into day-to-day issues due to weak engineering processes which do not allow him sufficient time to focus on customer needs. Together, these players seek intervention to improve this problematic situation.

Toward this end, we agreed to conduct an action research study with the above-mentioned individuals as collaborators.

We consider release management a good starting point for intervention to improve Software Inc.'s capabilities related to *Secure-on-Request*. Release management is the nub at which all of the above-described functions meet. The release-management area oversees end-to-end software engineering functions including requirement gathering, planning, designing, developing, testing, and coordinating deployment activities in the Software Development Lifecycle (SDLC). Looking at release management from the perspective of the product management and engineering teams provided a rich, internal picture emphasizing software engineering and management. At the same time, looking at the release-management function from a customer-perspective provided an external, service-oriented view. Hence, release management served as a platform for addressing the observed portfolio of problems, and drove improvements both in software process improvement and service innovation.

# A2.0 RELEASE CYCLE MANAGEMENT

Software release management is defined as “the process through which software is made available to and obtained by the user” (A. Van Der Hoek, Hall, Heimbigner, & Wolf, 1997). It includes the typically recurrent identification, packaging, and distribution of the elements of a product such as an executable program, documentation, release notes, and configuration data (Ballintijn, 2005; Scott & Nisse, 2001). The term “release” refers to the distribution of software outside of the development activity, and this includes internal releases as well as outside customers (Scott & Nisse, 2001). A well-defined release-management process can be the crux of increased quality of release- planning, building, testing, and deployment activities. This will likely reduce the number of problems occurring after delivering the release to customers (Lahtela & Jantti, 2011).

The fact that *Secure-on-Request* was inherited through acquisition might be part of the problem in the release-management process. High-tech companies acquire commercial off-the-shelf software components as a strategy to achieve efficient new product development (Brown & Eisenhardt, 1995; Kakola, Koivulahti-Ojala, & Liimatainen, 2009; Meyer & Seliger, 1998). Companies try to shorten the cycle of new product development while reducing cost and improving product quality and service delivery of their products in order to succeed in the global markets of software-intensive products and services (Kakola et al., 2009; Krishnan, 1994; Prasad, 1994). In general, software release management is further complicated by the increasing tendency for software to be assembled as a “system of systems," constructed from pre-existing,

independently created systems. Both developers and users of such software are affected by these trends (André Van der Hoek & Wolf, 2002)

Releasing a large software application is a complex procedure. In the case of *Secure-on-Request*, this complexity is heightened by the number of customers that use the service. A diverse and large customer base indicates a need for a substantial number of features to be included in the service. Furthermore, as the service evolves over time to incorporate the changing needs of customers, the release takes a great deal of effort and tends to be error-prone (Ballintijn, 2005). Delivering features that reliably meet customer requirements is an essential part of the release-management process; low-quality releases affect customer operations and the long-term relationship with their software providers (M. Kajko-Mattsson & Yulong, 2005). On-time delivery is equally critical to customer satisfaction (Prasad, 1994). Creating a robust software-release model and an effective release-management process will benefit business by reducing general cost and enhancing customer satisfaction (Rana & Arfi, 2005) .

Release management involves technical and management activities that take a release from a set of requirements to the final-delivery stage of the software (Danesh, Saybani, & Danesh, 2011). New management of the *Secure-on-Request* team adds challenges to the release process, since software typically result from the efforts of multiple individuals and teams (Otte, Moreton, & Knoell, 2008). Managing the work of multiple teams requires careful planning to ensure the quality of every part of the application. Meeting deadlines and documenting milestones is equally important. A

release manager can be appointed to coordinate the teams and to identify problems that might affect the software-release process (C. Jensen & Scacchi, 2005).

Release managers play the diverse role of interacting, planning and coordinating with different stakeholders, as well as understanding technical issues (C. Jensen & Scacchi, 2005; Michlmayr, Hunt, & Probert, 2007) .

Software quality and the success of release management hinge on having the right processes in place. Managers and developers must be provided with accurate information and guidelines to improve decision-making processes, plan and schedule activities, predict bottlenecks, allocate resources, and optimize implementation of change requests (Basili et al., 1996). Van der Hoek et al. (1997) noted that release management is "a poorly understood and underdeveloped part of the software process," and they pointed out several pertinent issues. Because efficient management of new-release production can improve software quality and customer satisfaction, the release-management process is crucial to the success of large software projects (Danesh et al., 2011) .

Software release management has garnered substantial academic and practical interest. We categorized the reviewed articles into four areas: standardization and development of models, process improvement, software quality, and customer and business perspectives. Standardization was the focus of several studies on software release management (Ballintijn, 2005; Biswas, 2007; M. Kajko-Mattsson & Yulong, 2005; Ramakrishnan, 2004; A. Van Der Hoek et al., 1997; André Van der Hoek & Wolf, 2002). Two studies identified specific issues in software-release management, offered a

list of requirements and proposed a prototype for a software release management tool called “SRM.” The tool was designed to aid both customers and developers in the software-release management process (A. Van Der Hoek et al., 1997; André Van der Hoek & Wolf, 2002). Several studies examined the overall release process. These studies identified problems and practices for release-management processes and offered practical suggestions (Bjarnason, Wnuk, & Regnell, 2010; Danesh et al., 2011; Erenkrantz, 2003; Kakola et al., 2009; Lahtela & Jantti, 2011). Release management has also been looked at in terms of release-quality (Boote et al., 2007; Michlmayr, 2005; Prasad, 1994; Rana & Arfi, 2005). For instance, Michlmayr (2005) found that improvement of release management impacted on quality issues facing open-source development. This research identified problems in release practices, and developed ways to improve release management in free-software projects. Finally, release management has been investigated from business and customer perspectives (B. B. Jensen, Lyngshede, & Søndergaard; M Kajko-Mattsson & Meyer, 2005; Krishnan, 1994). Krishnan (1994) presented an economic model to evaluate the tradeoffs involved in software-release decisions, and discussed techniques to achieve optimal software-release time (Krishnan, 1994) .

Research on software release management is limited. Consequently, no major improvements have been seen in tools and processes used in this area. Furthermore, it has been suggested that software-release processes have been “ad hoc and homegrown” in nature (Wright, 2009). Fierce market competition is now demanding a transformation of development strategies that provides timely product introduction and responsiveness to

customer need (Krishnan, 1994; Pratim Ghosh & Chandy Varghese, 2004). Therefore, we are proposing an action research study at Software Inc. on software rerelease management. Improvements in both software processes and service-delivery quality are targeted results. The theory and practice of release management is likely mainly instrumental in nature when focusing on the activity itself, that is, the perspective is of a first-order nature. We also zoomed in on and explored release management on a second-order level, that is, as an approach to organizational learning and innovation. In addition, we looked at release management from both an internal (engineering orientation) and external (customer orientation) perspective. Accordingly, our study contributed to the software organization and release-management literature regarding development of high-reliability capability, and to the SaaS and service-innovation literature regarding enhancing service-delivery quality by improving the release-management process. This knowledge will be of both practical and academic interest, as currently, significant resources are being expended on the software-release management process.

# A3.0 RESEARCH METHODOLOGY

## A3.1 Engaged Scholarship

To achieve deep insight into the process, we applied the principles of engaged scholarship, implying "negotiation and collaboration between researchers and practitioners in a learning community; such a community jointly produces knowledge that can both advance the scientific enterprise and enlighten a community of practitioners" (Van de Ven (2007), p.7).

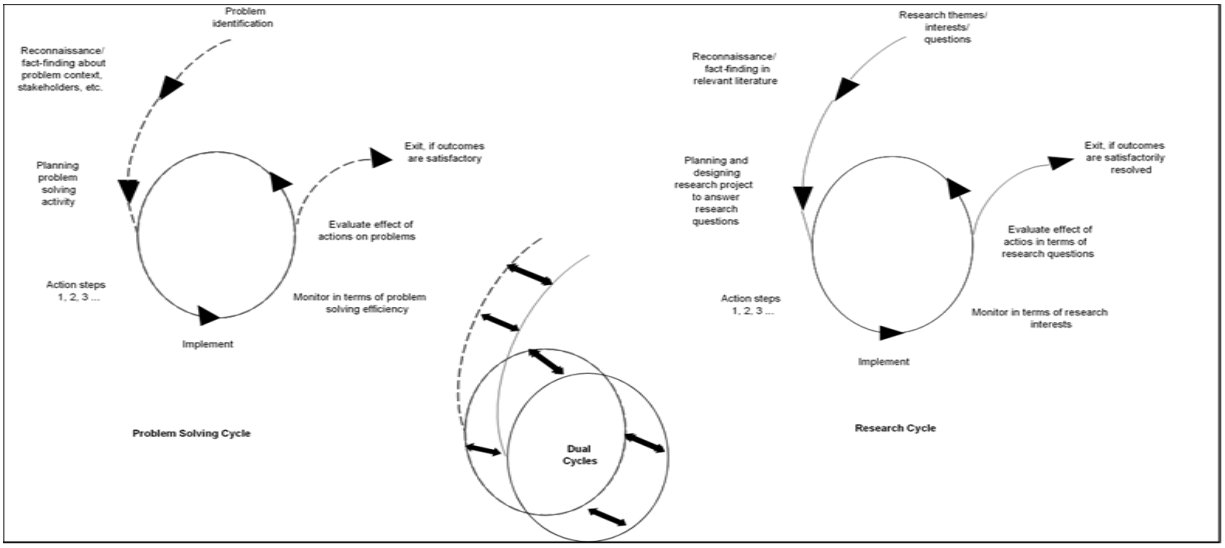
Van de Ven describes engaged scholarship as a participative form of research for obtaining the views of key stakeholders to understand a complex problem. By exploiting differences between these viewpoints, he argues that engaged scholarship produces knowledge that is more penetrating and insightful than when researchers work alone. Four alternative forms of engaged scholarship are defined by Van de Ven: (1) informed basic research with stakeholder advice that is undertaken to describe, explain or predict a social phenomenon; (2) co-produced knowledge with collaborators entailing a greater sharing of power and participation between researchers and stakeholders; (3) policy, design and evaluation research undertaken to develop knowledge related to design and evaluation of policies, programs and models for addressing practical and professional problems; and (4) action and intervention research for solving a client's problem while at the same time, contributing to the academic body of knowledge (Van de Ven, 2007). Of the four forms of engaged scholarship, we adopted action research for a number of

reasons: we had unlimited access to Software Inc., we had close relationships to the leadership of *Secure-on-Request*, we wanted to actively contribute to addressing the problems faced by the *Secure-on-Request* teams, and, we assumed such interventions would provide new valuable insights into release management and service provisioning in recurrent software practices. As a result, we adopted a clinical intervention approach to diagnose and resolve a portfolio of problems in a specific client context.

Action research was introduced by Kurt Lewin, and it makes use of intervention within challenging social situations as a means of developing scientific knowledge (Lewin, 1951; Rapoport, 1970). Rapport described action research as aiming "to contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable ethical framework" (1970, p. 499). Several action research approaches have been developed by subsequent scholars. Susman and Evered developed what has become known as Canonical Action Research (CAR) by expanding the work of Lewin and Rapoport to develop a client-system infrastructure and a multi- phased cyclical process for action research consisting of diagnosing, action planning, action taking, evaluating, and specifying learning (Davison, Martinsons, & Kock, 2004; Susman & Evered, 1978). McKay & Marshall, 2001 further developed the cyclical process of action research and introduced the two simultaneous cycles of research and problem-solving. McKay and Marshall's dual cycle framework enables researchers to diagnose problems and develop solutions in the problem-solving cycle while working closely with key stake holders. The

research cycle allows researchers to focus on developing and evaluating theory, while they start with an initial area of research interest and adopt the appropriate theoretical framework (McKay & Marshall, 2001). Figure 3.0 illustrates the two cycles and the exchange of information between them.

**Figure 3.0: Dual Cycle Model of Action Research at Software Inc. (McKay and Marshall 2001)**




## A3.2 Action Research Design

Our action research study aimed to simultaneously support the *Secure-on-Request* repositioning effort at Software Inc. and contribute to the body of scientific knowledge (Avison, Baskerville, & Myers, 2001; Baskerville & Wood-Harper, 1996). The general research approach is collaborative practice research (CPR). It is an action research methodology that advocates methodological pluralism and collaboration between

researchers and practitioners (Mathiassen, 2002). CPR methodology goal is to understand practice through interpretation, and to improve practice through interventions (Mathiassen, 2002). CPR suggests ways to achieve the right balance between relevance and rigor, requiring a dedicated effort involving both research and organizational work. Throughout our study we facilitated collaboration and managed the different agendas involved (Mathiassen, 2002). CPR disciplines complemented our action research approach, and allowed for collecting data systematically in addition to applying methods of interventions appropriately (Mathiassen, 2002).

We followed McKay and Marshall (2001) and organized our research into two parallel cycles: the problem-solving cycle and the research cycle. We adopted the IDEAL model (McFeeley, 1996) to guide our activities in the problem-solving cycle. Moreover, to ensure applicability and accuracy, we followed the five principles and associated criteria for Canonical Action Research (CAR) suggested by Davison et al. (2004). In Section 5, we provide a detailed account of how these principles were applied to our research at Software Inc.

Our action research was collaborative and iterative and focused on problem diagnosis, change, and reflection (Avison et al., 2001). Three methodological characteristics apply across the action research cycles (Baskerville & Wood-Harper, 1996). First, the researcher is actively involved with expected benefits for both the researcher and the organization. In our case, one of the researchers is the release manager of the project we are studying at Software Inc. His organization benefited from the ideas

developed during the problem-solving cycle through the enhancement of the knowledge base of their release management process. Second, immediate application of the knowledge obtained, and cyclical process linking theory and practice. As we moved forward with our activities, we applied the knowledge gained. Finally, the cyclical process should link theory and practice. Most participants were, to some extent, involved in all aspects of the action research cycles.

Rapoport (1970) identified three characteristic dilemmas of action research: ethics, goals and initiative. He suggested that a resolution in the science direction could lead away from action and vice versa. He also argued that "good" action research selectively combines elements of both directions. We were on the look-out for these dilemmas in our research with Software Inc. Examples of ethical dilemmas include researcher reactions to the client, managing confidentiality of participants, being approached by a competitor of a client, and personal involvement in the client's organization (Rapoport, 1970). Since one of the researchers is a manager at Software Inc., we were conscious of his dual role as researcher and employee of the client for whom we conducted the study. We consider that working with two other researchers and other stakeholders, and triangulating the data, will reduce the risks associated with dual allegiance. The discrepancy between practice and academic goals is the second dilemma identified by Rapport. We managed this dilemma by applying the recommended style composition practices (Mathiassen, Chiasson, & Germonprez, 2012), identifying the dual cycles of action research (McKay & Marshall, 2001), and recognizing the role duality as

an insider action research project raised by (Coghian, 2001). Initiative, which in this context concerns the solving of a client's problem as opposed to the pursuit of knowledge for knowledge's sake, is the third dilemma identified by Rapoport (Rapoport, 1970). The combined effort of multiple stakeholders when conducting engaged scholarship and action research provided the proper platform for us to deal with this dilemma.

# A4.0 PROBLEM-SOLVING CYCLE

We worked in a collaborative, stepwise, iterative fashion as we engaged in the problem-solving cycle to support the release-management and service-delivery processes at Software Inc. To guide our activities in the problem-solving cycle, we adopted the IDEAL model (McFeeley, 1996). This model is an approach for innovating software practices and was developed in 1996 by the Carnegie Mellon University Software Engineering Institute (McFeeley, 1996). The IDEAL model (Initiating, Diagnosing, Establishing, Acting, and Learning), illustrated in Figure 4.0, is very similar to the CAR five-phase cyclical approach (diagnosing, action planning, action taking, evaluating, and specifying learning) developed by Susman and Evered (1978). Enacting the phases of the IDEAL process guided our activities in the problem-solving cycle as well as provided opportunities to make research contributions as we studied the change processes over time.

**Figure 4.0: IDEAL Model (McFeeley, 1996)**

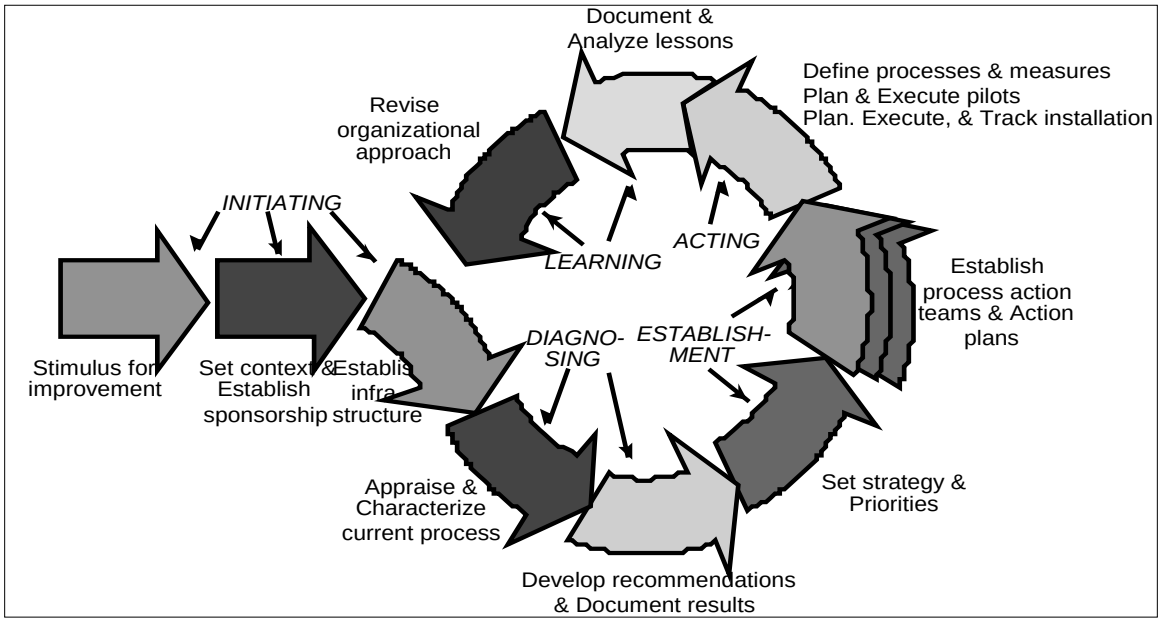



**Table 4.0: IDEAL Model Phases (McFeeley, 1996)**

| | |
|---|---|
| **Initiation phase** | Obtaining commitment, setting goals and establishing an improvement infrastructure |
| **Diagnostic phase** | Assess current practices; develop and prioritize recommendations for improvements |
| **Establishment phase** | Create specific, focused improvement initiatives. Teams are established to deal with each of the recommended improvement areas from the diagnostic phases |
| **Acting phase** | Develop and implement solutions for each improvement area. |
| **Learning phase** | Develop plan based on the results of the initiatives. Improvements data are collected and new evaluation is prepared |

### A4.1 Initiation Phase

In the initiation phase, we created an initial improvement infrastructure and established the "mutually acceptable ethical framework" (Rapoport, 1970) that served as the foundation for our study. We also secured a commitment from Software Inc. to work on the possible improvement areas (McFeeley, 1996). Table 4.1: Initiation Phase Key Dates provides a summary of key dates during the initiation phase at Software Inc. The research team received Institutional Review Board approval (IRB) on March 8 2013. The research team created a memorandum of understanding (MOU) which functioned as the researcher-client agreement (RCA) (Davison et al., 2004) for the study. The MOU defined the initial roles and responsibilities of both Software Inc. and the research team. It also clarified the dual objectives of contributing to research and practice, and provided an overview of project outcomes. Subsequently, we obtained approval for the improvement plans as well as a commitment for resources to accomplish future tasks.

**Table 4.1: Initiation Phase Key Dates**

| ***Date*** | ***Activity*** |
|---|---|
| **January 5, 2013** | Email sent to Software Inc. senior manager regarding possible collaboration |
| **January 12, 2013** | Invitation to collaboration meeting with Software Inc. senior management |

| | |
|---|---|
| **March 08 , 2013** | IRB Approval for Protocol Application Number: H13290 |
| **March 11, 2013** | The *Memorandum of Understanding* was shared and agreed to by Software Inc. |
| **March 15, 2013** | First meeting for the project steering committee |
| **April 09, 2013** | Starting Diagnostic Phase : First diagnostic interview was conducted |

### A4.2 Diagnostic Phase

In the diagnostic phase, we established the foundation for the later phases in the process. The goal of the diagnostic phase was to understand the current practices and challenges related to software release management and service delivery within Software Inc.

We assessed existing software-release and service-delivery practices related to *Secure-on-Request* at Software Inc. and established our baseline. We collected data between March 2013 and June 2013 to assess current practices from the viewpoint of key stakeholders at Software Inc. (Table 4.2-1: Diagnostic Phase Key Dates). Our diagnostic work included 16 semi-structured interviews, several meeting with Software Inc. stakeholders, and a review of performance data extracted from Software Inc. internal tracking tools and systems. Our assessment included perception-based methods constructed from our interviews and meetings with Software Inc. stakeholders (Napier, Mathiassen, & Johnson, 2009). It also included practice-based methods, derived from a review of release-management and service- delivery practices in the literature. Finally,

we analyzed the performance data and reported results extracted from the main tracking systems of Software Inc.

**Table 4.2-1: Diagnostic Phase Key Dates**

| ***Date*** | ***Activity*** |
|---|---|
| **April 09, 2013** | Starting Diagnostic Phase : First diagnostic interview was conducted |
| **April 10, 2013** | Meetings with product management team of *Secure-on-Request* started |
| **April 11, 2013** | Meetings with software development team of *Secure-on-Request* started |
| **May 22, 2013** | Last interview for initial diagnosis was completed |
| **June 05, 2013** | Release-management standards assessment completed |
| **June 10, 2013** | Service-quality standards assessment completed |
| **June 14, 2013** | First draft of diagnostic report completed |
| **June 20, 2013** | Steering committee meeting to share and discuss diagnostic findings |
| **June 28, 2013** | Establishment phase begins: First meeting to plan improvement projects |

For the practice-based part of the assessment, the research team selected norms and practices that were identified in the release-management literature (Elephant, 2006; Team, 2006), and compared them to current release practices at Software Inc. We also selected service-delivery principles identified in the service-science literature (Karpen, Bove, & Lukas, 2012; Schneider & Bowen, 2010; Vargo & Lusch, 2004), and compared

them to current service-delivery practices at Software Inc. The research team assigned scores based on data collected and observations, as it will be illustrated in the individual dissertation documents for the research team members (Barqawi, 2014; Syed, 2014)

In the perception-based part of the assessment we identified individuals from Software Inc. who were involved in the release process of *Secure-on-Request* as well as internal and external customers (Napier et al., 2009). The research team created an interview guide that discussed objective and subjective information about the release cycle and service-delivery processes related to *Secure-on-Request*. The research team conducted semi-structured interviews with the individuals listed in Table 4.2-2: Diagnosing Interview Sources.

**Table 4.2-2: Diagnosing Interview Sources**

| ***Group*** | ***Role*** | ***Count*** |
|---|---|---|
| Software Development | *Manager*<br>*Engineer* | *2* |
| *Quality Assurance* | *Manager*<br>*Engineer* | *2* |
| *Product Management* | Manager<br>PM | 2 |
| Project Management | Manager<br>Release Manager | 2 |
| Internal Customers | *Business Owner*<br>*Professional Services*<br>*Sales*<br>*Technical Account Managers* | *6* |

| | | |
|---|---|---|
| External Customers | *Manager*s | 2 |
| | Total | 16 |

The research team met and analyzed the interviews to reflect upon emerging themes on release-management and service-delivery practices related to Secure-on-Demand. Participants' viewpoints were analyzed with a focus on strengths and weaknesses of current release- management and service-delivery practices. The identified areas for improvement are illustrated in Table 4.2-3. We will expand on these identified areas in the research team members' individual dissertation documents (Barqawi, 2014; Syed, 2014), as it relates to their research focus.

**Table 4.2-3 Identified Possible Areas for Improvement at Software Inc.**

| ***Area*** | ***Identified Issues*** |
|---|---|
| Specifying and Stabilizing Requirements | • *Unclear requirements cause confusion, rework, delayed releases and adverse effects on our ability to ensure software quality.*<br>• *Inadequate verification of requirements quality*<br>*"In detailing our requirements there should always be a picture or a screenshot (wireframe) of what it should look like if it is a customer facing thing, so there will be no confusion"* |

| ***Area*** | ***Identified Issues*** |
|---|---|
| Prioritizing Requirements Across Channels | • *Expectations are high, release timeline is short, and resources are limited*<br>• *Too many inputs for requirements for detailed analysis due to time constraint*<br>• *Prioritization within and between new features development, escalations, fixing defects and technical debt are major challenges*<br><br>*"Our maturity and our ability to move forward with the prioritization process isn't still 100% there, and we all agree that is not what we want to be in the long term"* |
| *Managing Technical Debt* | • *Inherent product maturity issues*<br>• *Deadline pressure due to short release cycle*<br>• *Lack of unit test, peer code review, definition of "done"*<br>• *Technical debt often results in escalation of customer problems*<br><br>*"We definitely have some technical debt, and I would say moderate quality, it is not high quality, I think it is important to say that our technical debt in January was much higher than it is now"* |

| ***Area*** | ***Identified Issues*** |
|---|---|
| Testing Releases | • *New quality assurance team and new management. Continue to mature quality assurance processes*<br>• *Unclear and changing requirements adversely affect ability to ensure software quality*<br>• *Lack of visibility of planned features for releases: adding features late in the sprint creates challenges for QA*<br>• *Frequency of releases is affecting the time allowed for better testing for and stabilization of the software*<br><br>*"We don't have enough time between the end of the release and the time we put it out to get full quality regression tests done"* |
| Managing Release Cycles | • *Monthly releases help catch up with competition in market*<br>• *Monthly releases does not allow enough time for requirements analysis, testing, documentation and customer communication*<br><br>*"Frankly the customers can't absorb this frequent updates and changes, and in the process we haven't been given the customers enough time to know it is changing"*<br><br>*"We could do a 90 day cycle that could give us more time to provide more components and focus on the core capability of the application"* |
| Maintaining Complete Service Information | • *Information about features in new releases is not effectively communicated to TAM's and customers*<br>• *Release frequency is not allowing enough time for generating complete service information*<br><br>*"Release notes and user guide documentations, have been a real challenge because we have a monthly release cycles and how can you write documentation if you are actually writing codes the night before it goes out, it is pretty hard"* |

| ***Area*** | ***Identified Issues*** |
|---|---|
| Communicating Releases Across Customers | • *Release process is unclear for internal customers*<br>• *Technical account managers feel the need to "hedge" their communication to avoid failure to meet customers' expectations*<br>• *Customers require early notice of new features released*<br>• *Engineering work closely with Technical account managers, Beta is an initiative in this direction, Recent UI changes made to help*<br><br>*"Customers commented on one of latest releases as the following: you guys just released all that stuff and we were not expecting it, we are glad you are doing all that kind of stuff, but we want more notice"* |
| Giving Customers a Voice | • *Servicing large and diverse customer base allows for developing heterogeneous functions and features*<br>• *A need for better way to understand and address customer expectations and needs*<br>• *Fixing problems without changing the user interface making it difficult for customers to appreciate the enhancement*<br><br>*"Lack of certain usability features is seen as defects by customers, but this not how we see it"* |

During the course of the study, the steering committee was kept informed of the activities through weekly status reports and periodic status meetings. The research team documented the assessment findings in a complete diagnostic report, and a steering committee meeting was held on June 20, 2013 to describe the findings and overall recommendations. Table 4.2-4 illustrates the list of improvement options and recommendations shared with the steering committee during that meeting.

**Table 4.2-4 Suggested Improvement Options at Software Inc.**

| ***Area*** | ***Improvement Options*** |
|---|---|
| Release Frequency | *Move from 30 day to 90 day release model* |
| Service Requirements | • *Allow more time for requirements analysis*<br>• *Ensure key stakeholders agree on requirements and how they are prioritized*<br>• *Ensure requirements are explicated and effectively shared across developers, QA and documentation*<br>• *Ensure requirements changes are managed explicitly and shared effectively*<br>• *Use Wireframes to ensure effective communication between technical and business people*<br>• *Early demo of feature for key stakeholders* |
| Software Quality | • *Allow time for testing by reducing release frequency*<br>• *Involve QA early in the process to support development of test cases based on requirements*<br>• *Strengthen collaboration between development and QA about requirements, test cases, test results, and defect fixing*<br>• *Introduce automatic testing to free resources from mundane testing, provide quick feedback to developers, and focus on high-priority issues* |
| Customer Relationships | • *Help customers build knowledge and competence by maintaining complete service information and scheduling monthly customer webinars*<br>• *Gain better insight into customer needs and expectations by integrating support capability directly in the portal and scheduling quarterly on site reviews with customers*<br>• *Improve communication of releases across TAMs and customers by providing updates and notifications in the system on new features upon application access*<br>• *Continue assessments with key people, TAM's and customers to create stronger basis for improving customer relationships* |

### A4.3 Establishment Phase

In the establishment phase, we prioritized the issues that Software Inc. would address and we developed strategies for reaching solutions (Table 4.3-1: Establishment Phase Key Dates).

**Table 4.3-1: Establishment Phase Key Dates**

| ***Date*** | ***Activity*** |
|---|---|
| **June 28, 2013** | Establishment phase begins: First meeting to plan improvement projects |
| **July 1 , 2013** | Meetings with steering committee members to agree on strategy and deliverables of improvement projects |
| **July 2, 2013** | Acting phase begins: Kick-off meetings for improvement projects started |

We completed the detailed process-improvement plan based on the agreed-upon strategy, and designed plans to execute it. The suggested improvement strategy were implemented through a number of dedicated project teams with clear timelines and identified deliverables. The steering committee members agreed to form three teams to work on three improvement projects: customer relations, software quality, and release cycle. The details of these improvement projects will be discussed in the individual dissertation documents for the research team members (Barqawi, 2014; Syed, 2014). Table 4.3-2 shows an overview of the three improvement projects approved by the steering committee members. The steering committee was responsible for approving the overall plans for the improvements identified in the diagnostic phase.

**Table 4.3-2 *Secure-on-Request* Release Management and Service Delivery**

| ***Project Name*** | ***Project Roles*** | ***Project Deliverables*** |
|---|---|---|
| Improve Customer Relationship | • Project Manager: Release Manager<br>• Project Contributors: Business Owner, Product Manager, Technical Account Managers, Selected External Customers<br>• Project Consultants: Research team<br>• Project Sponsor: *Secure-on-Request* business owner | • Enhanced Service Usability<br>• Value Added Services<br>• Capturing The Voice of The Customer<br>• Operational Preparedness<br>• Implementation Plan<br>• Leadership Team Commitment |
| Improve Requirements And Quality | • Project Manager: Release Manager<br>• Project Contributors: Development Manager, Product Managers, QA Managers<br>• Project Consultants: Research team<br>• Project Sponsor: *Secure-on-Request* business owner | • Requirement Management Process<br>• Requirement Specification Formats<br>• Development–Test Exchange Process<br>• Development–Test–Documentation Management<br>• Operational Preparedness<br>• Implementation Plan<br>• Leadership Team Commitment |
| Improve Release Cycle | • Project Manager: Release Manager<br>• Project Contributors: Development Manager, Product Manager, QA Manager<br>• Project Consultants: Research team<br>• Project Sponsor: *Secure-on-Request* business owner | • Revised Release Model<br>• Customer Communication Strategy<br>• Operational Preparedness<br>• Implementation Plan<br>• Leadership Team Commitment |

### A4.4 Acting Phase

In the acting phase, we positioned the improvement projects agreed on at Software Inc., to address the areas for improvement identified during the diagnosing phase (Table 4.4: Acting Phase Key Dates). The strategy and prioritization as well as deliverables were agreed upon in the establishment phase. The research team and steering committee members held a kick-off meeting for each improvement project. At the kick-off meetings, the teams were given a set of objectives and deliverables. The teams were provided with draft project plans along with expected delivery dates. Numerous meetings were held between research team members and improvement teams to work on the deliverables and assess progress. An interim status meeting for the steering committee was held on August 19, 2013, where a status update on the three projects was presented and progress was discussed.

**Table 4.4: Acting Phase Key Dates**

| ***Date*** | ***Activity*** |
|---|---|
| **July 2, 2013** | Acting phase begins: Kick-off meetings for improvement projects started |
| **July 2 , 2013** | Kick-off meeting for improved customer relationship project |
| **July 3, 2013** | Kick-off meeting for improved requirements and quality project |
| **July 5, 2013** | Kick-off meeting for improved release cycle project |
| **August 19, 2013** | Interim status meeting for steering committee members |

| | |
|---|---|
| **September 30, 2013** | Deliverables from project teams due |
| **October 26, 2013** | Learning Phase begins: acting phase completion meeting |

The project team members provided projects deliverables for review on September 30, 2013. The completion meeting to close this phase was conducted on October 19, 2013. The details and key outcomes for each project are included in the individual dissertation documents for the research team members (Barqawi, 2014; Syed, 2014).

### A4.5 Learning Phase

In the learning phase, we reviewed the implemented solutions as well as evaluated the outcome of the three improvement projects (Table 4.5: Learning Phase Key Dates). Our learning phase assessments included perception-based as well as practice-based methods (Napier et al., 2009) with a focus on evaluating the impact on the release cycle and service-delivery process of *Secure-on-Request*. our goal was to identify changes in each of the three project improvement areas, the effect on the processes as well as the challenges that occurred during implementing the changes, and suggestions for improvement. For the perception-based assessment, we conducted fourteen semi-structured interviews with the key stakeholders. Each interview was around 45 minutes, and was recorded, and later transcribed. Our goal was to determine how different stakeholders perceived the overall value of the improvement projects implemented, their

satisfaction with their own level of involvement, as well as suggestions for future improvement. For the practice-based part of the assessment, we used the norms and practices from release management and service-delivery literature identified in the diagnostic phase (Elephant, 2006; Team, 2006; Karpen, Bove, & Lukas, 2012; Schneider & Bowen, 2010; Vargo & Lusch, 2004) and compared them to software release management service-delivery practices at Software Inc. after implement the improvement projects. The research team assigned scores based on data collected and observations, and the assessment results were compared against those from the diagnosing phase as it will be illustrated in the individual dissertation documents for the research team members (Barqawi, 2014; Syed, 2014). The resulting assessments and findings were summarized. An overall assessment of the value of the improvement projects will be discussed in details the individual dissertation documents for the research team members (Barqawi, 2014; Syed, 2014).

**Table 4.5: Learning Phase Key Dates**

| ***Date*** | ***Activity*** |
|---|---|
| **October 26, 2013** | Learning Phase started |
| **November 14, 2013** | First learning phase interview was conducted |
| **December 5, 2013** | Last learning phase interview was completed |
| **February 28, 2014** | Release-management standards assessment completed |
| **February 28 , 2014** | Service-quality standards assessment completed |

# A5.0 RESEARCH CYCLE

The research cycle for this study was guided by the style composition for action research developed by Mathiassen, et al. (2012). Our research explored software release management, software improvement, and software-as-a-service and service-science streams of literature. The study employed Pettigrew's contextualist inquiry theory (Pettigrew, 1985) to analyze how release cycle management can be improved in the context of recurrent development of software. Additionally, the study adopted Service-dominant logic as a theoretical framework (Vargo & Lusch, 2004) to analyze how the release management process can be organized to improve Software Inc.'s ongoing value co-creation with its customers. Our research process was a collaborative and iterative process highlighting problem diagnosis, change, and reflection (Avison et al., 2001). Furthermore, our study satisfied the three methodology characteristics that were described across action research cycles (Baskerville & Wood-Harper, 1996). First, the researcher is actively involved with expected benefits for both the researcher and the organization. In our case, one of the researchers was the release manager of the project we are studying at Software Inc. We expect that as a manager, his organization will benefit from the suggestions developed during the problem-solving cycle and add to the understanding of their release-management process. Secondly, we linked theory and practice through immediate application of the knowledge obtained, and by following the cyclical process. Using our research at Software Inc., we applied knowledge gained as we moved forward to the next set of activities.

We followed CAR principles of action research to guarantee rigor as we conducted our study and depicted the research cycles (Davison et al., 2004). As explained in Section 3 on the adopted action research design, the authors provided specific questions and criteria for each principle (Davison et al., 2004) to guide the study.

**A5.1 Data Collection**

Action research and qualitative research require rigorous documentation, data collection, and documentation methods (Avison et al., 2001; Miles & Huberman, 1994). Our study employed several sources for data collection, which include interviews, meetings, field observations, researchers' notes, and unlimited access to Software Inc. internal systems reports and process documentation. For our diagnostic phase, we identified key individuals from Software Inc. to be interviewed for our study. We conducted sixteen one-hour face-to-face as well as phone interviews. All interviews were conducted in English, and detailed notes were taken. All interviews were recorded. During the course of our data collection, we used triangulation (Miles & Huberman, 1994) to counterbalance any insider bias (Coghian, 2001). Table 5.1 outlines the specific primary and secondary data sources for our data collection phase. Data collection methods for the study are discussed in more detail in the individual dissertation documents for the research team members (Barqawi, 2014; Syed, 2014).

**Table 5.1: Primary and Secondary Data Sources**

| ***Primary Data Sources*** | ***Secondary Data Sources*** |
|---|---|
| Meetings:<br>• Release Management Meetings (Weekly)<br>• Bi-Weekly Scrums<br>• Monthly Release Planning and Demos<br>• Daily Customer Escalation Calls | Release management documentation tools:<br>• Requirements Management tool<br>• Defect Management tool<br>• Customer Relationship Management tool |
| Semi-structured interviews:<br>• Professional Services<br>• Sales<br>• Quality Assurance<br>• Product Management<br>• Operational Services<br>• Development<br>• Business Unit Owner<br>• Technical Account Management<br>• Project Managers<br>• External Customer | |

**A5.2 Data Analysis**

Analysis was performed using a variety of qualitative data analysis techniques and followed the guidelines suggested by Miles and Huberman (1994). We used Pettigrew's contextualist inquiry theory and its adopted constructs (Pettigrew, 1985) in analyzing the data related to the study of release management focused on the internal software process improvement at Software Inc. We also used Service-dominant logic as framework (Vargo & Lusch, 2004, 2008) in analyzing the data related to the service delivery practices of *Secure-on-Request*. Additionally, our study followed the qualitative

data analysis strategy offered by Miles and Huberman (1994). They propose three concurrent flows of activities: data reduction, data display, and conclusion drawing and verification. These activities were enacted continuously throughout the data collection process as it is explained in more detail in the individual dissertation documents for the research team members (Barqawi, 2014; Syed, 2014).

Our team of researchers independently analyzed the interviews and meetings transcripts and used triangulation throughout the data analysis to offset potential for insider-bias related to the role held by one of our research team members in Software Inc. (Coghian, 2001). Qualitative data analysis software (NVIVO) was used to classify, tabulate, and visualize the data. We used the constructs and concepts from the adapted theoretical framework to analyze and code our data. Data analysis strategy and outcome of the study will be discussed in more detail in the individual dissertation documents for the research team members (Barqawi, 2014; Syed, 2014).

# A6.0 PRINCIPLES OF CANONICAL ACTION RESEARCH

We followed the principles of CAR to ensure rigor as we conducted our study at Software Inc. Davison, Martinsons and Kock write that CAR is directed by five principles: 1) researcher-client agreement; 2) cyclical process model; 3) theory; 4) change through action; and 5) learning through reflection (2004). The authors provide criteria for each principle that we followed to ensure the rigor and relevance of our study (Davison et al., 2004).

Following the principle of Researcher-Client Agreement (Davison et al., 2004), we provided a framework for our research by communicating the overall objectives of the study and by explaining the roles of research team members. The Memorandum of Understanding on Research Collaboration (MoU) that we initially shared with Software Inc. clearly stated the objective of the research project. Software Inc. committed the time and resources needed to complete the study. The business owner of the product *Secure-on-Request* at Software Inc. became the sponsor of the project and helped identify the roles of the steering committee as well as those of the problem-solving project's team members. Key deliverables and evaluation criteria were communicated to all stakeholders. Software Inc. also agreed to our data collection methods including interviews, meeting attendance, and data and reports from internal systems and internal communications. Table 6.1 lists the evaluation of the principle of Researcher-Client Agreement criteria of our study.

**Table 6.1: Criteria for the Researcher-Client Agreement**

| ***Principle 1 – Criteria for the Researcher - Client Agreement*** | ***Applied to Software Inc.*** | |
|---|---|---|
| 1a – Did both the researcher and the client agree that CAR was the appropriate approach for the organizational situation? | No | No explicit agreement with Software Inc., but we followed the CAR principles to guide our research effort. |
| 1b – Was the focus of the research project specified clearly and explicitly? | Yes | Our MoU with Software Inc. clearly stated the objective of the study: Improving processes and services in a software unit: An action research study into release management. |
| 1c – Did the client make an explicit commitment to the project? | Yes | Software Inc. committed to the project the time and resources needed to complete the study. |
| 1d – Were the roles and responsibilities of the researcher and client organization members specified explicitly? | Yes | Steering committee as well as the problem solving team were specified. |
| 1e – Were project objectives and evaluation measures specified explicitly? | Yes | Key deliverables and evaluation criteria were communicated to all stakeholders. |
| 1f – Were the data collection and analysis methods specified explicitly? | Yes | Software Inc. approved our data collection methods, including interviews, meeting attendance, data and reports from internal systems, and internal communications. |

The principle of the Cyclical Process Model evaluates the relationship between diagnosing and acting (Davison et al., 2004). It emphasizes the need for modifying processes based on continuing evaluations. We followed McKay and Marshall's (2001) dual-cycle model; therefore, the information gleaned from the problem-solving cycle was

incorporated into the research cycle, and the knowledge from the research cycle was integrated in the problem-solving cycle. We modified our project plans throughout the course of our study in response to challenges encountered and new knowledge gained. Continuous evaluation of our strategy and results were discussed in meetings held between steering committee members. Table 6.2 summarizes the evaluation of the principle of Cyclical Process Model criteria of our study.

**Table 6.2: Criteria for the Cyclical Process Model**

| ***Principle 2– Criteria for the Cyclical Process Model (CPM)*** | ***Applied to Software Inc.*** | |
|---|---|---|
| 2a – Did the project follow the CPM or justify any deviation from it? | Yes | We followed McKay and Marshall's (2001) dual-cycle model, therefore the information from the problem-solving cycle added to the research cycle while the knowledge from the research cycle was employed in the problem-solving cycle. |
| 2b – Did the researcher conduct an independent diagnosis of the organizational situation? | Yes | |
| 2c – Were the planned actions based explicitly on the results of the diagnosis? | Yes | |
| 2d – Were the planned actions implemented and evaluated? | Yes | |
| 2e – Did the researcher reflect on the outcomes of the intervention? | Yes | |

| | | |
|---|---|---|
| 2f – Was this reflection followed by an explicit decision on whether or not to proceed through an additional process cycle? | Yes | Throughout the course of our study we modified our project plans based on challenges encountered and new knowledge gained. Continuous evaluation of our strategy and results were discussed in meetings held between steering committee members. |

The Principle of Theory focuses the research cycle and the project by ensuring that the research is guided by a theoretical framework (Davison et al., 2004). We adopted Pettigrew's contextualist inquiry theory as a framework to analyze how release cycle management can be improved in the context of recurrent development of software (Pettigrew, 1985). Based on insights from our analysis, the study developed recommendations for software providers to manage their software releases and software processes. Our study also adopted the service-dominant logic framework (Vargo & Lusch, 2004) to analyze how the release-management process can be organized to improve Software Inc.'s ongoing value co-creation with its customers. As a result, the study contributed to improving release management at Software Inc. and added to knowledge about the challenges and opportunities for software vendors to manage releases and improve the value delivered to and co-created with their customers. The theoretical frameworks chosen for our study guided our interventions and research activities as well as helped in evaluating the outcomes. Table 6.3 summarizes the evaluation of the Principle of Theory criteria of our study.

**Table 6.3: Criteria for the Principle of Theory**

| ***Principle 3 – Criteria for the Principle of Theory*** | ***Applied to Software Inc.*** | |
|---|---|---|
| 3a – Were the project activities guided by a theory or set of theories? | Yes | We adopted Pettigrew's *contextualist inquiry* theory as a framework to analyze how release cycle management can be improved in the context of recurrent development of software. *Service-dominant logic* framework was adopted to analyze how the release management process can be organized to improve Software Inc.*'s* ongoing value co-creation with its customers. |
| 3b – Was the domain of investigation and the specific problem setting relevant to, and significant for, the interest of the researcher's community of peers as well as the client? | Yes | |
| 3c – Was a theoretically based model used to derive the causes of the observed problem? | Yes | |
| 3d – Did the planned intervention follow from this theoretically based model? | Yes | The theoretical frameworks chosen for our study guided our intervention and research activities at Software Inc. as well as helped in evaluating the outcomes. |

The principle of Change through Action helps researchers and clients isolate and resolve problems (Davison et al., 2004). Research team members and the steering committee agreed to improve both the release process of *Secure-on-Request* and the service quality delivered to their customers. The researchers and steering committee members identified specific areas for improvement after a comprehensive assessment was conducted. The research team ensured that decisions were made with the involvement of all relevant stakeholders at Software Inc. The process and plans for the project were documented and progress was communicated to all stakeholders. Consequently, Software

Inc. was supportive of our efforts throughout the project and was appreciative of the work done to improve their release-management process and service quality. Table 6.4 summarizes the evaluation of the principle of Change through Action criteria.

**Table 6.4: Criteria for the Principle of Change through Action**

| ***Principle 4 – Criteria for the Principle of Change through Action*** | ***Applied to Software Inc.*** | |
|---|---|---|
| 4a – Were both the researcher and client motivated to improve the situation? | Yes | Software Inc. and the research team members agreed on improving the release process of *Secure-on-Request* and improving the service quality delivered to customers. |
| 4b – Were the problem and its hypothesized cause(s) specified as a result of the diagnosis? | Yes | Specific areas for improvement were identified after a comprehensive assessment was conducted at Software Inc. |
| 4c – Were the planned actions designed to address the hypothesized cause(s) | Yes | |
| 4d – Did the client approve the planned actions before they were implemented? | Yes | Decisions were made with the involvement of all relevant stakeholders. Project plans were documented and progress was communicated to all stakeholders. |
| 4e – Was the organization situation assessed comprehensively both before and after the intervention? | Yes | |
| 4f – Were the timing and nature of the actions taken clearly and completely documented? | Yes | |

The principle of Learning through Reflection concerns learning through reflection from practical work as well as research (Davison et al., 2004). The research team

discussed in a meeting with the steering committee members the areas targeted for improvement in the software-release and the service-delivery process. Shortly thereafter, initial recommendations for improvement in these areas were communicated to Software Inc. The research team provided an update on the status of each improvement project in a weekly communication that was sent out to key stakeholders. Several meetings were held with key stakeholders from Software Inc. to assess progress and discuss ways to ensure continuous improvement and rigorous data collection. Table 6.5 summarizes the evaluation of the principle of the Learning through Reflection criteria.

**Table 6.5 Criteria for the Principle of Learning through Reflection**

| ***Principle 5 – Criteria for the Principle of Learning through Reflection*** | ***Applied to Software Inc.*** | |
|---|---|---|
| 5a – Did the researcher provide progress reports to the client and organizational members? | Yes | The research team provided an update on the status of each improvement project, in a weekly communication material that was sent out to Software Inc. key stakeholders. |
| 5b – Did both the researcher and the client reflect upon the outcomes of the project? | Yes | The research team discussed the areas needed for improvement Software Inc. Initial recommendations for improvement were communicated to key stakeholders shortly thereafter. |
| 5c – Were the research activities and outcomes reported clearly and completely? | Yes | |
| 5d – Were the results considered in terms of implications for further action in this situation? | Yes | Several meetings were held with key stakeholders from Software Inc. to assess progress and discuss ways to ensure continuous improvement and rigorous data collection |
| 5e – Were the results considered in terms of implications for actions to be taken in related research domains? | Yes | |
| 5f – Were the results considered in terms of implications for the research community (general knowledge, informing/re-informing theory)? | Yes | |
| 5g – Were the results considered in terms of the general applicability of CAR? | Yes | |

In sum, we applied literature-derived knowledge on, Pettigrew's contextualist inquiry theory and service-dominant logic as theoretical frameworks (Pettigrew, 1985; Vargo & Lusch, 2004, 2008), and action research as a methodology (Davison et al., 2004;

Lewin, 1951; Mathiassen, 2002; McKay & Marshall, 2001; Rapoport, 1970), and engaged in collaborative research and problem-solving at Software Inc. Our research aimed to provide rich data for software-process and service-delivery improvements at Software Inc.

# APPENDIX A REFERENCES


Avison, D., Baskerville, R., & Myers, M. (2001). Controlling action research projects. Information technology & people, 14(1), 28-45.

Ballintijn, G. (2005). A case study of the release management of a health-care information system. Paper presented at the proceedings of the IEEE International Conference on Software Maintenance, ICSM2005, Industrial Applications track.

Barqawi, N. (2014). Software Service Innovation: An Action Research into Release Cycle Management.

Basili, V., Briand, L., Condon, S., Kim, Y.-M., Melo, W. L., & Valett, J. D. (1996). Understanding and predicting the process of software maintenance release. Paper presented at the Proceedings of the 18th international conference on Software engineering.

Baskerville, R. L., & Wood-Harper, A. T. (1996). A critical perspective on action research as a method for information systems research. Journal of Information Technology, 11(3), 235-246.

Biswas, P. K. (2007). Autonomic Software Release Management for Communications Networks. Paper presented at the Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on.

Bjarnason, E., Wnuk, K., & Regnell, B. (2010). Overscoping: Reasons and consequences—A case study on decision making in software product management. Paper presented at the Software Product Management (IWSPM), 2010 Fourth International Workshop on.

Boote, J. W., Hanemann, A., Kudarimoti, L., Louridas, P., Marta, L., Michael, M., . . . Tsompanidis, I. (2007). Quality assurance in perfSONAR release management. Paper presented at the Quality of Information and Communications Technology, 2007. QUATIC 2007. 6th International Conference on the.

Brown, S. L., & Eisenhardt, K. M. (1995). Product development: Past research, present findings, and future directions. Academy of management review, 343-378.

Coghian, D. (2001). Insider Action Research Projects Implications for Practising Managers. Management Learning, 32(1), 49-60.

Danesh, A. S., Saybani, M. R., & Danesh, S. Y. S. (2011). Software release management challenges in industry: An exploratory study. African Journal of Business Management, 5(20), 8050-8056.

Davison, R., Martinsons, M. G., & Kock, N. (2004). Principles of canonical action research. Information Systems Journal, 14(1), 65-86.

Elephant, P. (2006). ITIL IT Service Management Essentials. Course Workbook. Burlington, Ontario: Pink Elephant Inc.

Erenkrantz, J. R. (2003). Release management within open source projects. Proceedings of the 3rd Open Source Software DevelopmentWorkshop, 51-55.

Jensen, B. B., Lyngshede, S., & Søndergaard, D. Quality Assurance Recommendations for Open Source Developers.

Jensen, C., & Scacchi, W. (2005). Collaboration, leadership, control, and conflict negotiation and the netbeans. org open source software development community. System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on, 196b-196b.

Kajko-Mattsson, M., & Meyer, P. (2005). Evaluating the acceptor side of EM< sup> 3</sup>: release management at SAS. Paper presented at the Empirical Software Engineering, 2005. 2005 International Symposium on.

Kajko-Mattsson, M., & Yulong, F. (2005). Outlining a model of a release management process. Journal of Integrated Design and Process Science, 9(4), 13-25.

Kakola, T., Koivulahti-Ojala, M., & Liimatainen, J. (2009). An Information Systems Design Theory for Integrated Requirements and Release Management Systems. Paper presented at the System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on.

Karpen, I. O., Bove, L. L., & Lukas, B. A. (2012). Linking Service-Dominant Logic and Strategic Business Practice A Conceptual Model of a Service-Dominant Orientation. Journal of Service Research, 15(1), 21-38.

Krishnan, M. S. (1994). Software release management: a business perspective. Paper presented at the Proceedings of the 1994 conference of the Centre for Advanced Studies on Collaborative research.

Lahtela, A., & Jantti, M. (2011). Challenges and problems in release management process: A case study. Paper presented at the Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on.

Lewin, K. (1951). Field theory in social science: selected theoretical papers (Edited by Dorwin Cartwright.).

Mathiassen, L. (2002). Collaborative practice research. Information Technology & People, 15(4), 321-345.

Mathiassen, L., Chiasson, M., & Germonprez, M. (2012). Style composition in action research publication. MIS Quarterly, 36(2), 347-363.

McFeeley, B. (1996). IDEAL: A User's Guide for Software Process Improvement: DTIC Document.

McKay, J., & Marshall, P. (2001). The dual imperatives of action research. Information Technology & People, 14(1), 46-59.

Meyer, M. H., & Seliger, R. (1998). Product platforms in software development. Sloan Management Review, 40(1), 61-74.

Michlmayr, M. (2005). Quality improvement in volunteer free software projects: Exploring the impact of release management. Paper presented at the Proceedings of the First International Conference on Open Source Systems.

Michlmayr, M., Hunt, F., & Probert, D. (2007). Release management in free software projects: Practices and problems. Open Source Development, Adoption and Innovation, 295-300.

Miles, M. B., & Huberman, A. M. (1994). Qualitative data analysis: An expanded sourcebook: Sage Publications, Incorporated.

Napier, N. P., Mathiassen, L., & Johnson, R. D. (2009). Combining perceptions and prescriptions in requirements engineering process assessment: an industrial case study. Software Engineering, IEEE Transactions on, 35(5), 593-606.

Otte, T., Moreton, R., & Knoell, H. D. (2008). Applied quality assurance methods under the open source development model. Paper presented at the Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International.

Pettigrew, A. M. (1985). Contextualist research and the study of organizational change processes. Research methods in information systems, 53-78.

Prasad, R. K. (1994). Towards a zero-defect product-the End-To-End test process. Paper presented at the Software Testing, Reliability and Quality Assurance, 1994. Conference Proceedings., First International Conference on.

Pratim Ghosh, P., & Chandy Varghese, J. (2004). Globally distributed product development using a new project management framework. International Journal of Project Management, 22(8), 669-678.

Ramakrishnan, M. (2004). Software release management. Bell Labs Technical Journal, 9(1), 205-210.

Rana, A. I., & Arfi, M. W. (2005). Software Release Methodology: A Case Study. Paper presented at the Engineering Sciences and Technology, 2005. SCONEST 2005. Student Conference on.

Rapoport, R. N. (1970). Three dilemmas in action research with special reference to the Tavistock experience. Human relations, 23(6), 499-513.

Schneider, B., & Bowen, D. E. (2010). Winning the service game: Springer.

Scott, J. A., & Nisse, D. (2001). Software configuration management. SWEBOK, 103.

Susman, G. I., & Evered, R. D. (1978). An assessment of the scientific merits of action research. Administrative Science Quarterly, 582-603.

Syed, K. (2014). Improving Recurrent Software Development: A Contextualist Inquiry into Release Cycle Management.

Team, C. P. (2006). CMMI for Development, version 1.2.

Van de Ven, A. H. (2007). Engaged Scholarship: A Guide for Organizational and Social Research: A Guide for Organizational and Social Research: Oxford University Press.

Van Der Hoek, A., Hall, R., Heimbigner, D., & Wolf, A. (1997). Software release management. Software Engineering—ESEC/FSE'97, 159-175.

Van der Hoek, A., & Wolf, A. L. (2002). Software release management for component-based software. Software: Practice and Experience, 33(1), 77-98.

Vargo, S. L., & Lusch, R. F. (2004). Evolving to a new dominant logic for marketing. Journal of marketing, 1-17.

Vargo, S. L., & Lusch, R. F. (2008). Service-dominant logic: continuing the evolution. Journal of the Academy of Marketing Science, 36(1), 1-10.

Wright, H. K. (2009). Release engineering processes, models, and metrics. Paper presented at the Proceedings of the doctoral symposium for ESEC/FSE on Doctoral symposium.

# APPENDIX B: *SECURE-ON-REQUEST* RELEASE MANAGEMENT IMPROVEMENT PROJECTS – STATUS REPORT

## Secure-on-Request Release Management Improvement Projects

| Project Name | Leadership Team | Research Team | Project Roles | Project Deliverables |
|---|---|---|---|---|
| Improve Customer Relationship | • Business Owner English<br>• PM 1<br>• Dev Manager | • GSU Research Team | • Project Manager: Release Manager<br>• Project Contributors: Business Owner , PM 1, TAMs Manager, Select TAMs<br>• Project Consultants: GSU Research Team<br>• Project Sponsor: Business Owner | • Enhanced Service Usability<br>• Value Added Services<br>• Capturing The Voice of The Customer<br>• Operational Preparedness<br>• Implementation Plan<br>• Leadership Team Commitment |
| Improve Requirements And Quality | | | • Project Manager: Release Manager<br>• Project Contributors: Dev Manager, PM 1, PM 2, QA Manager, QA Engineer<br>• Project Consultants: GSU Research Team<br>• Project Sponsor: Business Owner | • Requirement Management Process<br>• Requirement Specification Formats<br>• Development–Test Exchange Process<br>• Development–Test–Documentation Management<br>• Operational Preparedness<br>• Implementation Plan<br>• Leadership Team Commitment |
| Improve Release Cycle | | | • Project Manager: Release Manager<br>• Project Contributors: Business Owner, PM 1, Dev Manager, QA Manager<br>• Project Consultants: GSU Research Team<br>• Project Sponsor: Business Owner | • Revised Release Model<br>• Customer Communication Strategy<br>• Operational Preparedness<br>• Implementation Plan<br>• Leadership Team Commitment |

## Improve Secure-on-Request Customer Relationship

| Research Team | Project Roles | Project Deliverables |
|---|---|---|
| • GSU Research Team | • Project Manager: Release Manager<br>• Project Contributors: Business Owner, PM 1, TAMs Manager, Select TAMs<br>• Project Consultants: GSU Research Team<br>• Project Sponsor: Business Owner | • Enhanced Service Usability<br>• Value Added Services<br>• Capturing The Voice of The Customer<br>• Operational Preparedness<br>• Implementation Plan<br>• Leadership Team Commitment |

### Project Schedule

| Project Milestones | Target Dates |
|---|---|
| Project Start Date | Week of 7/1/2013 |
| Implementation Decision | By 8/15/2013 |
| Implementation Complete | By 9/30/2013 |
| Lessons Learned | By 10/15/2013 |

# Improve Secure-on-Request Requirements And Quality

| Project Roles | Project Deliverables |
|---|---|
| • Project Manager: Release Manager<br>• Project Contributors: Dev Manager, PM 1, PM 2, QA Manager, QA Engineer<br>• Project Consultants: GSU Research Team<br>• Project Sponsor: Business Owner | • Requirement Management Process<br>• Requirement Specification Formats<br>• Development–Test Exchange Process<br>• Documentation Management<br>• Operational Preparedness<br>• Implementation Plan<br>• Leadership Team Commitment |

## Project Schedule

| Project Milestones | Target Dates |
|---|---|
| Project Start Date | Week of 7/1/2013 |
| Implementation Decision | By 8/15/2013 |
| Implementation Complete | By 9/30/2013 |
| Lessons Learned | By 10/15/2013 |

# Improve Secure-on-Request Release Cycle Project

| Research Team | Project Roles | Project Deliverables |
|---|---|---|
| • GSU Research Team | • Project Manager: Release Manager<br>• Project Contributors: PM 1, Dev Manager, QA Manager<br>• Project Consultants: GSU Research Team<br>• Project Sponsor: Business Owner | • Revised Release Model<br>• Customer Communication Strategy<br>• Operational Preparedness<br>• Implementation Plan<br>• Leadership Team Commitment |

## Suggested Project Schedule

| Project Milestones | Target Dates |
|---|---|
| Project Start Date | Week of 7/1/2013 |
| Implementation Decision | By 8/15/2013 |
| Implementation Complete | By 10/19/2013 |
| Lessons Learned | By 10/31/2013 |

# APPENDIX C: CUSTOMER ADVISORY BOARD - MEETING ITEMS

## Logistics

- An annual user group meeting is an ideal forum for holding smaller advisory councils since the customers are already there
- Three or four vendor employees, led by product management, facilitate the meeting. Development manager also should be involved; Sales people usually want to be on-hand if their customers are invited
- It is not recommended to make commitments in a customer advisory meeting. This is an input session, not a decision-making meeting
- Invite six to eight customer representatives. Many will want to send two people, often a technology advisor as well as a business representative of the customer.

## Sample Agenda

9:00 – Introductions

- Introductions of company personnel and customers.

9:10 a.m. — CAB Mission & Purpose

- State how the Customer Advisory Board meeting operates, how members can contribute and what both parties receive from the experience.

9:30 a.m. — Customer presentation

- Ask the customer to present three or four slides about how they are using your product and challenges they are facing when dealing with your product and company

## Sample Agenda- Cont.

10:00 a.m. – Roadmap of product plans

- Overview of Secure-on- Request product roadmap . Give a brief overview of planned features and their benefits to the customer

10:30 a.m. – Refreshment and Discussion

- Discuss features ideas, new products suggestions. Ask about professional services and web-based support. Ask how your customers get information on new products and new releases.

11:15 a.m. – Summary

- Acknowledge what the Customer Advisory Board has achieved and what actions will be taken

11:30 a.m. - Adjournment

## Sample Invitation

Dear Customer,

This is a formal invitation to participate in the Secure-on- Request Customer Advisory Board.

We're hosting a Customer Advisory Board to connect industry leaders like you, learn more about your unmet needs and innovate together

We selected you from among many of our customers because your views are important to us and we value your commitment to a long term relationship with our company

Customer Advisory Board meetings will provide a forum and sounding board to share your ideas and help achieve your business goals. We require members to attend two professionally facilitated working sessions annually.

We welcome you to join us at the first meeting in September 2013 during Software Inc. conference.

Please let us know your decision or if you have any questions?

# APPENDIX D: OCTOBER 2013 RELEASE *SECURE-ON-REQUEST* RELEASE CHECKLIST

**From:** Syed, Kamran
**To:**
**Subject:** FoD October Release Activities - FINAL UPDATE
**Date:** Sunday, October 20, 2013 7:05:00 PM

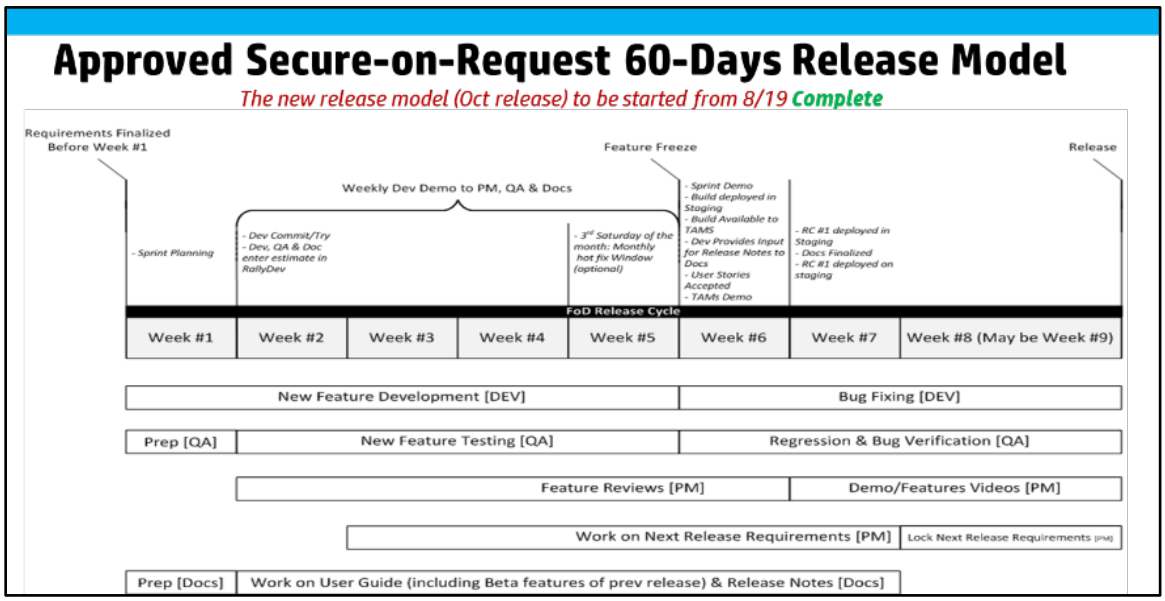
| | | | Event Update Email | | Status |
|---|---|---|---|---|---|
| | | | From | To | |
| TWO WEEKS AGO | Tuesday, October 01, 2013 | Dynamic security team start security assessment | Kamran | Dennis Antunes, Andrew Ringlein | Complete |
| | Thursday, October 03, 2013 | Regression Testing Started | Dwayne | Kamran | Complete |
| | Thursday, October 03, 2013 | Static Testing | Gil | Knight, Kamran | Complete |
| | Friday, October 04, 2013 | PM Sign Off in UAT complete | PM | FOD_PMENGOPS | Complete (Conditional approval given) |
| | Friday, October 04, 2013 | Business Sign Off in UAT complete | Ryan/Ben | FOD_PMENGOPS | Complete (Conditional approval given) |
| LAST WEEK | Tuesday, October 08, 2013 | Dynamic Security Assessment Final Report | Dennis Antunes, Andrew Ringlein | FOD_PMENGOPS | Complete |
| | Thursday, October 10, 2013 | Doc (User Guide & Release Notes) checked in build | Heather | FOD_PMENGOPS | Complete |
| | Friday, October 11, 2013 | Dev hands off RC#1 build to Ops to deploy on staging | Knight | FOD_PMENGOPS | Complete |
| | Friday, October 11, 2013 | Go/No-Go Meeting | Kamran | FOD_PMENGOPS | Complete – Next Go/No-Go Meeting on 10/17 |
| THIS WEEK | Monday, October 14, 2013 | Ops confirms RC#1 deployed on staging | Ops | FOD_PMENGOPS | Complete |
| | Monday, October 14, 2013 | QA test in staging | Dwayne | FOD_PMENGOPS | Complete |
| | Thursday, October 17, 2013 | Go/No-Go Meeting | Kamran | FOD_PMENGOPS | Complete – It's a GO |
| | Thursday, October 17, 2013 | Dev hands off final RC build to Ops to deploy on staging | Knight | FOD_PMENGOPS | Complete |
| | Friday, October 18, 2013 | Ops confirms final RC deployed on staging | Ops | FOD_PMENGOPS | Complete |
| | Friday, October 18, 2013 | QA finished testing of final RC in staging | Dwayne | FOD_PMENGOPS | Complete |
| | Saturday , 10/19 | Internal communication sent confirming production update is starting and Production is down. | Ops | FOD_PMENGOPS | Complete |
| | Saturday , 10/19 | Ops deploy final RC to production | Ops | FOD_PMENGOPS | Complete |
| | Saturday , 10/19 | Ops confirms via email final RC deployed on production | Ops | FOD_PMENGOPS | Complete |
| | Saturday , 10/19 | QA starts validation (Checks Basic Functionality (Dashboards, LMS, Dynamic start/finish, Static start/finish, report generation) | Dwayne | FOD_PMENGOPS | Complete |
| | Saturday , 10/19 | QA completes validation and sends confirmation email | Dwayne | FOD_PMENGOPS | Complete |
| | Saturday , 10/19 | Final communication sent confirming release is live or postponed | Ops | FOD_DEPLOY, FOD_PMENGOPS | Complete |

# APPENDIX E: DATA ANALYSIS SCHEMA (AS SEEN IN NVIVO)

**Nodes**

| Name | Sources | References | Created On |
|---|---|---|---|
| Inner Context | 1 | 1 | 12/25/2013 1:34 PM |
| People | 21 | 68 | 12/25/2013 1:34 PM |
| Culture | 20 | 67 | 12/25/2013 1:35 PM |
| Politics | 9 | 20 | 12/25/2013 1:36 PM |
| Hot-Fixes | 11 | 17 | 1/2/2014 7:53 AM |
| Managment | 9 | 14 | 12/25/2013 1:35 PM |
| Technology | 4 | 6 | 12/25/2013 1:35 PM |
| Structure | 4 | 6 | 12/25/2013 1:35 PM |
| Outer Context | 0 | 0 | 12/25/2013 1:32 PM |
| Customers | 16 | 67 | 12/25/2013 1:33 PM |
| Competetors | 7 | 13 | 12/25/2013 1:32 PM |
| Market | 8 | 13 | 12/25/2013 1:33 PM |
| HP at Large | 4 | 5 | 12/25/2013 1:33 PM |
| Content | 0 | 0 | 12/25/2013 1:36 PM |
| Release Cycle Process | 16 | 44 | 12/25/2013 1:37 PM |
| Release Cycle Management | 12 | 23 | 12/25/2013 1:37 PM |
| Recurrent Product Development | 13 | 22 | 12/25/2013 1:37 PM |
| Release Cycle Organization | 11 | 18 | 12/25/2013 1:38 PM |
| Process | 0 | 0 | 12/25/2013 1:39 PM |
| Improvement Goals | 17 | 36 | 12/25/2013 1:39 PM |
| Improvement Projects | 8 | 10 | 12/25/2013 1:45 PM |
| Improvement Organization | 6 | 8 | 12/25/2013 1:45 PM |
| Improvement Process | 1 | 1 | 12/25/2013 1:42 PM |
| Miscellaneous | 0 | 0 | 1/3/2014 12:34 PM |
| Interesting | 10 | 38 | 1/2/2014 8:36 AM |
| Lars | 1 | 2 | 1/3/2014 12:41 PM |
| Background Information | 1 | 1 | 1/3/2014 12:53 PM |
| SPI | 0 | 0 | 1/3/2014 12:54 PM |
| SPI Inhibitors | 16 | 67 | 12/25/2013 1:48 PM |
| SPI Commitment | 9 | 13 | 12/25/2013 1:48 PM |
| SPI Enablers | 9 | 9 | 12/25/2013 1:48 PM |

# APPENDIX F: *SECURE-ON-REQUEST* NEW RELEASE CYCLE MODEL



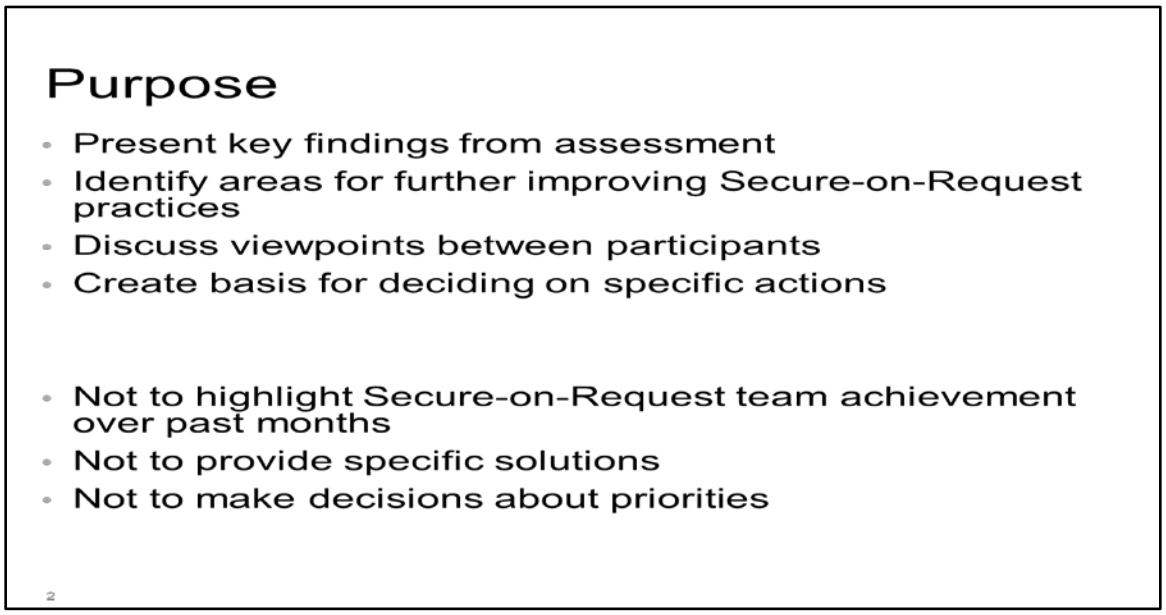

## Reoccurring Meetings in 60-Day Release Cycle

| Meeting | Purpose | Facilitator | Invitees | Schedule |
|---|---|---|---|---|
| Stakeholders Meeting | Gather & discuss input from key stakeholders for the requirements of the next release | Release Manager | Business Owner, PM 1, PM 2, TAMs Manager, Dev Manager, Service OPs | Thursday of week #7 of previous release |
| Sprint Planning | Sprint team to agree to complete the set of ready top-ordered backlog items | Release Manager | PM, Dev, QA, Docs, PMO teams | Monday of the Week #1 |
| PM & Docs Review | PM walks Doc team through requirements to start working on the documentation impact for the release | Release Manager | PM 2 & Docs | Tuesday of week #1 |
| Weekly Dev Demo | Dev demos to PM, QA & Docs weekly completed new features. Week #2 thru Week #5 only. | Release Manager | PM, Dev, QA, Docs, PMO teams | Wednesdays of week #2 thru week #5 |
| Sprint Demo | Dev team to show the work they have accomplished | Release Manager | PM, Dev, QA, Docs, PMO teams & TAMs Manager | Monday of week #6 |
| TAMs Demo | PM to show to the TAMs new features included in the release | Release Manager | Business Owner, PM, TAMs, Dev, Docs, PMO teams | Tuesday of week #6 |
| Sprint Retrospective | For sprint team to learn what works and what does not work and make adjustments for the next sprint | Release Manager | PM, Dev, QA, Docs, PMO teams | First Thursday after the release |
| Retrospective with Ops | For Ops team to learn what works and what does not work and make adjustments for the next sprint | Release Manager | Business Owner, Service OPs, Dev Manager, QA | First Friday after the release |

Release Manager to schedule a retrospective meeting to discuss the release cycle – a one hour session is scheduled ***Complete***

# APPENDIX G: *SECURE-ON-REQUEST* RELEASE MANAGEMENT ASSESSMENT AND IMPROVEMENT OPTIONS

Meeting with Steering Committee - June 20th 2013



## Agenda

- Background
  - Project Goals
  - Overall Approach
  - Assessment Method
- Assessments
  - Performance Data Assessment
  - Practice Assessment
  - Interview Assessment
- Improvement Options
  - Area #1: Release Frequency
  - Area #2: Service Requirements
  - Area #3: Software Quality
  - Area #4: Customer Relationships
- Discussion

3

## Project Goals

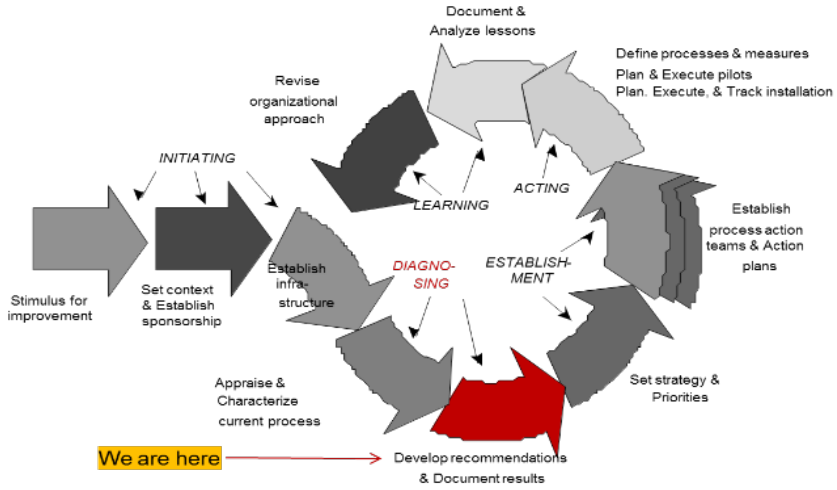
1. Improve Secure-on-Request's engineering team ability to effectively manage releases for its products and to quickly respond to customer needs
2. Develop contributions to new knowledge about software release management

The research focuses on two different perspectives:

- An *internal engineering and management perspective*
- An *external and customer focused perspective*

4

# Assessment Context




*This research is based on the IDEAL model, a software process improvement framework by the Software Engineering Institute (SEI)*

5

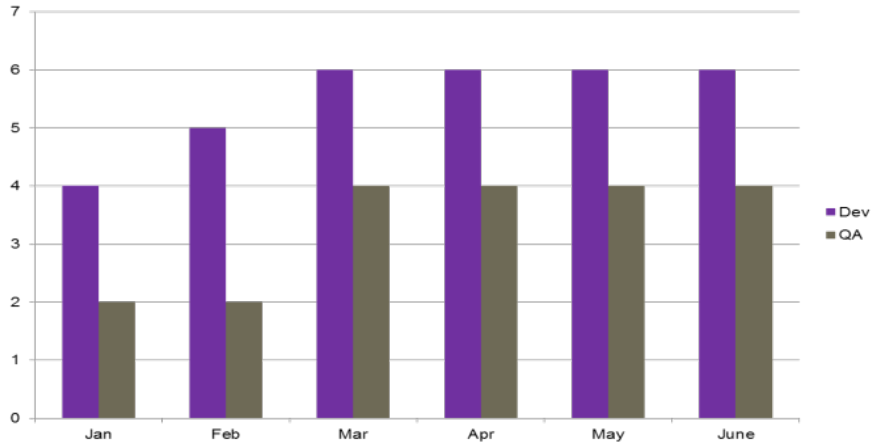
# Assessment Method

- Interviews
- Analyzed data
  - Defect System
  - Requirement System
  - Release Timeline
- Reviewed literature
  - Release management
  - Software service delivery
- Observations

| | Interviewee |
|---|---|
| 1 | (Professional Services/Pre-Sales) |
| 2 | (Sales) |
| 3 | (QA) |
| 4 | (Product Mgmt. 1) |
| 5 | (Ops) |
| 6 | (Dev) |
| 7 | (QA Manager) |
| 8 | (Business Owner) |
| 9 | (Dev Manager) |
| 10 | (Product Mgmt. 2) |
| 11 | (TAMs) |
| 12 | (PMO) |
| 13 | (PMO) |

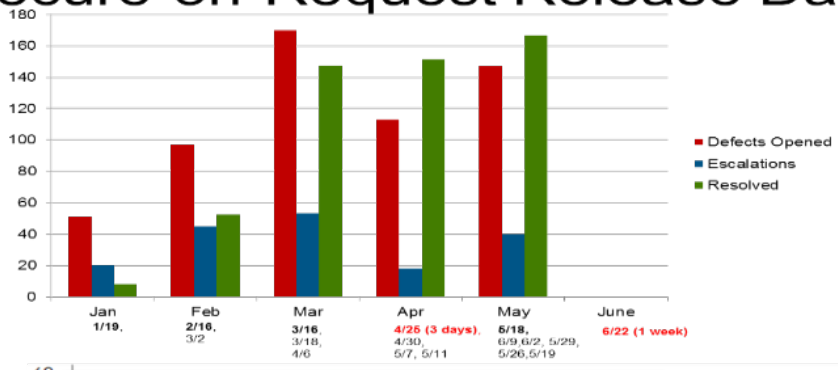
6

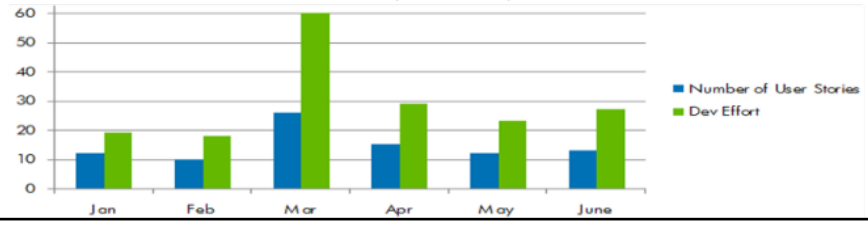
# Secure-on-Request Engineering Team Resources




7

# Secure-on-Request Release Data




8

## Interview Assessment

- Specifying and Stabilizing Requirements
- Prioritizing Requirements Across Channels
- Managing Technical Debt
- Testing Releases
- Managing Release Cycles
- Maintaining Complete Service Information
- Communicating Releases Across Customers
- Giving Customers A Voice

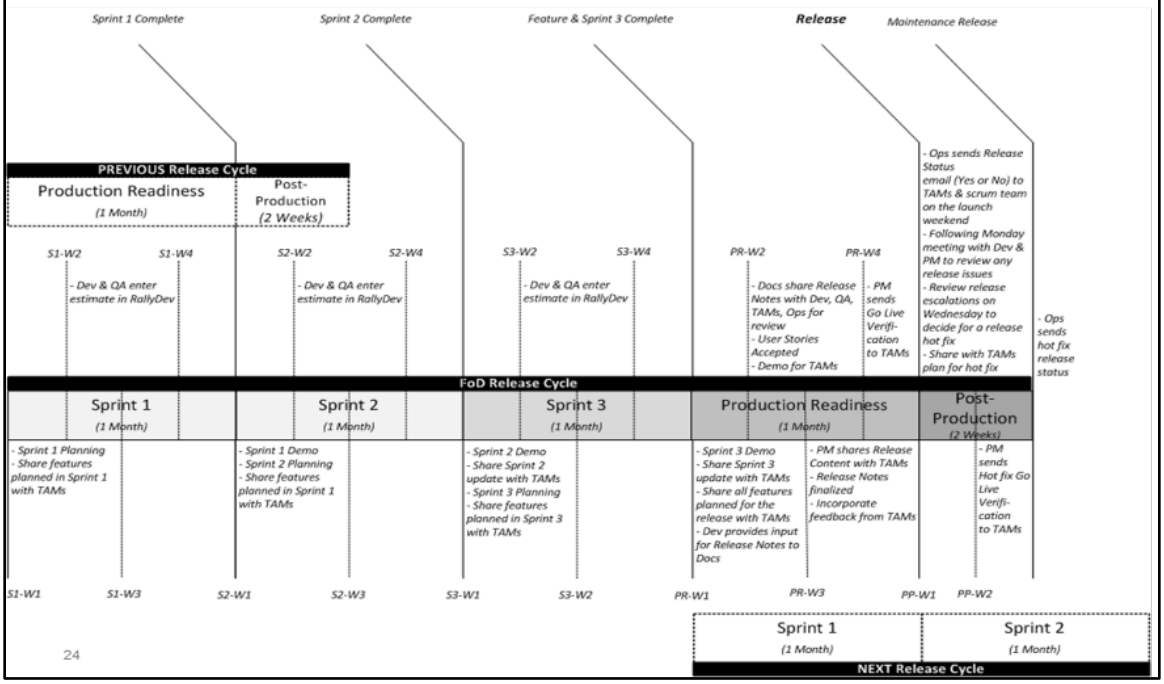
14

## Improvement Options

- Area #1: Release Frequency
- Area #2: Service Requirements
- Area #3: Software Quality
- Area #4: Customer Relationships

23

# Area #1: Release Frequency (90 day model)




# Area #2: Service Requirements

- Allow more time for requirements analysis
- Ensure key stakeholders agree on requirements and how they are prioritized
- Ensure requirements are explicated and effectively shared across developers, QA and documentation
- Ensure requirements changes are managed explicitly and shared effectively
- Use Wireframes to ensure effective communication between technical and business people
- Early demo of feature for key stakeholders

25

## Area #3: Software Quality

- Allow time for testing by reducing release frequency
- Involve QA early in the process to support development of test cases based on requirements
- Strengthen collaboration between development and QA about requirements, test cases, test results, and defect fixing
- Introduce automatic testing to free resources from mundane testing, provide quick feedback to developers, and focus on high-priority issues

26

## Area #4: Customer Relationships

- Help customers build knowledge and competence by maintaining complete service information and scheduling monthly customer webinars
- Gain better insight into customer needs and expectations by integrating support capability directly in the portal and scheduling quarterly on site reviews with customers
- Improve communication of releases across TAMs and customers by providing updates and notifications in the system on new features upon application access
- Continue assessments with key people, TAM's and customers to create stronger basis for improving customer relationships

27

# REFERENCES


Aaen, I. (2002, January). Challenging Software Process Improvement By Design. In ECIS (pp. 379-390).

Abrahamsson, P. (2000). Is management commitment a necessity after all in software process improvement?. In Euromicro Conference, 2000. Proceedings of the 26th (Vol. 2, pp. 246-253). IEEE.

Abrahamsson, P. (2001). Rethinking the concept of commitment in software process improvement. Scandinavian Journal of Information Systems, 13, 69-98.

Allison, I., & Merali, Y. (2007). Software process improvement as emergent change: A structurational analysis. Information and software technology, 49(6), 668-681.

Avison, D., Baskerville, R., & Myers, M. (2001). Controlling action research projects. Information technology & people, 14(1), 28-45.

Bach, J. (1995). Enough about process: what we need are heroes. Software, IEEE, 12(2), 96-98.

Ballintijn, G. (2005, September). A Case Study of the Release Management of a Health-care Information System. In ICSM (Industrial and Tool Volume) (pp. 34-43).

Banker, R. D., & Slaughter, S. A. (2000). The moderating effects of structure on volatility and complexity in software enhancement. Information Systems Research, 11(3), 219-240.

Banker, R. D., Datar, S. M., Kemerer, C. F., & Zweig, D. (1993). Software complexity and maintenance costs. Communications of the ACM, 36(11), 81-94.

Banker, R. D., Davis, G. B., & Slaughter, S. A. (1998). Software development practices, software complexity, and software maintenance performance: A field study. Management Science, 44(4), 433-450.

Barqawi, N., & Syed, K. (2014). Improving Processes and Services in a Software Unit: An Action Research Study into Release Management.

Barqawi, N. (2014). Software Service Innovation: An Action Research into Release Cycle Management.

Baskerville, R. L., & Wood-Harper, A. T. (1996). A critical perspective on action research as a method for information systems research. Journal of Information Technology, 11(3), 235-246.

Baskerville, R., & Pries-Heje, J. (1999). Grounded action research: a method for understanding IT in practice. Accounting, Management and Information Technologies, 9(1), 1-23.

Baskerville, R., & Pries-Heje, J. (1999). Knowledge capability and maturity in software management. ACM SIGMIS Database, 30(2), 26-43.

Baskerville, R., & Wood-Harper, A. T. (1998). Diversity in information systems action research methods. European Journal of Information Systems, 7(2), 90-107.

Benlian, A., Koufaris, M., & Hess, T. (2011). Service quality in software-as-a-service: developing the SaaS-Qual measure and examining its role in usage continuance. Journal of Management Information Systems, 28(3), 85-126.

Bitner, M. J., Ostrom, A. L., & Morgan, F. N. (2008). Service blueprinting: a practical technique for service innovation. California Management Review, 50(3), 66.

Bollinger, T. B., & McGowan, C. (1991). A critical look at software capability evaluations. Software, IEEE, 8(4), 25-41.

Borjesson, A., & Mathiassen, L. (2004). Successful process implementation. Software, IEEE, 21(4), 36-44.

Brodman, J. G., & Johnson, D. L. (1995). Return on investment (ROI) from software process improvement as measured by US industry. Software Process: Improvement and Practice, 1(1), 35-47.

Carlshamre, P. (2002). Release planning in market-driven software product development: Provoking an understanding. Requirements Engineering, 7(3), 139-151.

Carmel, E., & Becker, S. (1995). A process model for packaged software development. Engineering Management, IEEE Transactions on, 42(1), 50-61.

Coghian, D. (2001). Insider Action Research Projects Implications for Practising Managers. Management Learning, 32(1), 49-60.

Colomo-Palacios, R., Soto-Acosta, P., García-Peñalvo, F. J., & García-Crespo, Á. (2012). A study of the impact of global software development in packaged software release planning. Journal of Universal Computer Science, 18(19), 2646-2668.

Crosby, P. B. (1979). Quality is free: The art of making quality certain (Vol. 94). New York: McGraw-Hill.

Curtis, B. (1994). A Mature View of the CMM. American Programmer, 7, 19-19.

Davison, R., Martinsons, M. G., & Kock, N. (2004). Principles of canonical action research. Information systems journal, 14(1), 65-86.

Deming, W. E. (1992). Out of Crisis, Massachusetts Institute of Technology, Centre for Advanced Engineering Study.

Diaz, M., & Sligo, J. (1997). How software process improvement helped Motorola. Software, IEEE, 14(5), 75-81.

Dorenbos, D., & Combelles, A. (2004). Introduction: Lessons Learned around the World: Key Success Factors to Enable Process Change. Software, IEEE, 21(4), 20-21.

Dyba, T. (2005). An empirical investigation of the key factors for success in software process improvement. Software Engineering, IEEE Transactions on, 31(5), 410-424.

Elephant, Pink. (2006). ITIL IT Service Management Essentials. Course Workbook. Burlington, Ontario: Pink Elephant Inc.

Fayad, M. E., & Laitnen, M. (1997). Process assessment considered wasteful. Communications of the ACM, 40(11), 125-128.

Fichman, R. G., & Kemerer, C. F. (1997). The assimilation of software process innovations: an organizational learning perspective. Management Science, 43(10), 1345-1363.

Frederiksen, H. D., & Mathiassen, L. (2008). A contextual approach to improving software metrics practices. Engineering Management, IEEE Transactions on, 55(4), 602-616.

Gaur, P., & Oberoi, A. (2012). A Resource Oriented Intelligent Scheduling Scheme to Estimate Software Release. International Journal, 2(9).

Grady, R. B. (1997). Successful software process improvement (p. 253). Prentice Hall PTR.

Guerrero, F., & Eterovic, Y. (2004). Adopting the SW-CMM in a Small IT Organization. Software, IEEE, 21(4), 29-35.

Haley, T. J. (1996). Software process improvement at Raytheon. Software, IEEE, 13(6), 33-41.

Harter, D. E., Kemerer, C. F., & Slaughter, S. A. (2012). Does software process improvement reduce the severity of defects? A longitudinal field study. Software Engineering, IEEE Transactions on, 38(4), 810-827.

Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., & Paulk, M. (1997). Software quality and the capability maturity model. Communications of the ACM, 40(6), 30-40.

Humble, J., & Farley, D. (2010). Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education.

Humphrey, W. S. (1989). Managing the Software Process (Hardcover). Addison-Wesley Professional.

Humphrey, W. S., & Curtis, B. (1991). Comments ona critical look'[software capability evaluations]. Software, IEEE, 8(4), 42-46.

Humphrey, W. S., Snyder, T. R., & Willis, R. R. (1991). Software process improvement at Hughes Aircraft. Software, IEEE, 8(4), 11-23.

Juran, J. M. (1959). A Note on Economics of Quality. Industrial Quality Control, pp. 20-23, 1959.

Juran, J. M. (1992). Juran on quality by design: the new steps for planning quality into goods and services. SimonandSchuster.com.

Kemerer, C. F. (1995). Software complexity and software maintenance: A survey of empirical research. Annals of Software Engineering, 1(1), 1-22.

Kohoutek, H. J. (1996). Reflections on the capability and maturity models of engineering processes. Quality and reliability engineering international, 12(3), 147-155.

Krishnan, M. S., & Kellner, M. I. (1999). Measuring process consistency: Implications for reducing software defects. Software Engineering, IEEE Transactions on, 25(6), 800-815.

Kuvaja, P., & Bicego, A. (1994). BOOTSTRAP—a European assessment methodology. Software Quality Journal, 3(3), 117-127.

Lahtela, A., & Jantti, M. (2011, July). Challenges and problems in release management process: A case study. In Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on (pp. 10-13). IEEE.

Larsen, E. A., & Kautz, K. (1996, December). Quality assurance and software process improvement in Norway. In Software Process, 1996. Proceedings., Fourth International Conference on the (pp. 131-148). IEEE.

Lee, Y. W. (2003). Crafting rules: context-reflective data quality problem solving. Journal of Management Information Systems, 20(3), 93-119.

Lewin, K. (1952). Field theory in social science: Selected theoretical papers. D. Cartwright (Ed.). London: Tavistock.

Li, G., & Rajagopalan, S. (1998). Process improvement, quality, and learning effects. Management Science, 44(11-Part-1), 1517-1532.

Mashiko, Y., & Basili, V. R. (1997). Using the GQM paradigm to investigate influential factors for software process improvement. Journal of Systems and Software, 36(1), 17-32.

Mathiassen, L. (1998). Reflective Systems Development. Scandinavian Journal of Information Systems, 10(1&2), 67-118.

Mathiassen, L. (2002). Collaborative practice research. Information Technology & People, 15(4), 321-345.

Mathiassen, L., & Pedersen, K. (2008). Managing uncertainty in organic development projects. Communications of the Association for Information Systems, 23(1), 27.

Mathiassen, L., & Sørensen, C. (1996). The capability maturity model and CASE. Information Systems Journal, 6(3), 195-208.

Mathiassen, L., Chiasson, M., & Germonprez, M. (2012). Style Composition in Action Research Publication. MIS Quarterly, 36(2), 347-363.

Mathiassen, L., Nielsen, P. A., & Pries-Heje, J. (2002). Learning SPI in practice. Improving Software Organizations: From Principles to Practice.

Mazlan, M. A., Sefat, M. H., Selan, N. E., & Lukose, D. (2013). Holistic approach to software build and release process with pre-emptive measures for sustainable quality control. In 22nd Australasian Software Engineering Conference: ASWEC 2013 (p. 12). Engineers Australia.

McFeeley, B. (1996). IDEAL: A User's Guide for Software Process Improvement (No. CMU/SEI-96-HB-001). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

Miles, M. B., & Huberman, A. M. (1994). Qualitative data analysis: An expanded sourcebook. Sage.

Müller, S. D., Mathiassen, L., & Balshøj, H. H. (2010). Software Process Improvement as organizational change: A metaphorical analysis of the literature. Journal of Systems and Software, 83(11), 2128-2146.

Mumford, E. (2001). Helping Organizations to Change. Qualitative Research in International Settings: Issues and Trends, 46.

Myers, M. D. (2008). Qualitative research in business and management. Sage.

Napier, N. P., Mathiassen, L., & Johnson, R. D. (2009). Combining perceptions and prescriptions in requirements engineering process assessment: an industrial case study. Software Engineering, IEEE Transactions on, 35(5), 593-606.

Napier, N. P., Mathiassen, L., & Robey, D. (2011). Building contextual ambidexterity in a software company to improve firm-level coordination. European Journal of Information Systems, 20(6), 674-690.

Ncube, C., Oberndorf, P., & Kark, A. W. (2008). Opportunistic software systems development: making systems from what's available. IEEE Software, 25(6), 38-41.

Ngwenyama, O., & Nielsen, P. A. (2003). Competing values in software process improvement: an assumption analysis of CMM from an organizational culture perspective. Engineering Management, IEEE Transactions on, 50(1), 100-112.

Niazi, M., Wilson, D., & Zowghi, D. (2006). Critical success factors for software process improvement implementation: an empirical study. Software Process: Improvement and Practice, 11(2), 193-211.

Nielsen, P. A., & Nørbjerg, J. (2001). Software process maturity and organizational politics. Working Conference on Realigning Research and Practice in Information Systems Development: The Social and Organizational Perspective, 2001, pp. 221–240.

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). Capability maturity model, version 1.1. Software, IEEE, 10(4), 18-27.

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.

Pettigrew, A., McKee, L., & Ferlie, E. (1988). Understanding change in the NHS. Public Administration, 66(3), 297-317.

Qian, L., Yao, Q., & Khoshgoftaar, T. M. (2010). Dynamic Two-phase Truncated Rayleigh Model for Release Date Prediction of software. Journal of Software Engineering & Applications, 3(6).

Ramasubbu, N., Mithas, S., Krishnan, M. S., & Kemerer, C. F. (2008). Work dispersion, process-based learning, and offshore software development performance. MIS quarterly, 32(2), 437-458.

Rapoport, R. N. (1970). Three dilemmas in action research with special reference to the Tavistock experience. Human relations, 23(6), 499-513.

Ravichandran, T., & Rai, A. (2000). Quality management in systems development: an organizational system perspective. Mis Quarterly, 381-415.

Regnell, B., & Kuchcinski, K. (2011, August). Exploring Software Product Management decision problems with constraint solving-opportunities for prioritization and release planning. In Software Product Management (IWSPM), 2011 Fifth International Workshop on (pp. 47-56). IEEE.

Ruhe, G., & Saliu, M. O. (2005). The art and science of software release planning. Software, IEEE, 22(6), 47-53.

Sawyer, S. (2000). Packaged software: implications of the differences from custom approaches to software development. European Journal of Information Systems, 9(1), 47-58.

Scott, J. A., & Nisse, D. (2001). Software configuration management. SWEBOK, 103.

Sommerville, I. (1995) Software Engineering, Fifth Edition, Addison-Wesley, Reading, Massachusetts.

Stelzer, D., & Mellis, W. (1998). Success factors of organizational change in software process improvement. Software Process: Improvement and Practice, 4(4), 227-250.

Susman, G. I., & Evered, R. D. (1978). An assessment of the scientific merits of action research. Administrative science quarterly, 582-603.

Svahnberg, M., Gorschek, T., Feldt, R., Torkar, R., Saleem, S. B., & Shafique, M. U. (2010). A systematic review on strategic release planning models. Information and software technology, 52(3), 237-248.

Taborda, L. J. (2011). Enterprise Release Management: Agile Delivery of a Strategic Change Portfolio. Artech House.

Team, CMMI Product. (2006). CMMI for Development, version 1.2.

Truex, D. P., Baskerville, R., & Klein, H. (1999). Growing systems in emergent organizations. Communications of the ACM, 42(8), 117-123.

Van Der Hoek, A., Hall, R. S., Heimbigner, D., & Wolf, A. L. (1997). Software release management (Vol. 22, No. 6, pp. 159-175). ACM.

Van der Velden, M. J., Vreke, J., Van Der Wal, B., & Symons, A. (1996). Experiences with the Capability Maturity Model in a research environment. Software Quality Journal, 5(2), 87-95.

Xu, L., & Brinkkemper, S. (2007). Concepts of product software. European Journal of Information Systems, 16(5), 531-541.

Yin, R. K. (2008). Case study research: Design and methods (Vol. 5). Sage