APPROVAL SHEET

Title of Dissertation: Software Process Improvement through the Removal of Project-level Knowledge Flow Obstacles: The Perceptions of Software Engineers

Name of Candidate: Susan Marie Mitchell
Doctor of Philosophy, 2012

Dissertation and Abstract Approved: Carolyn B. Seaman

Carolyn B. Seaman
Associate Professor
Information Systems Department

Date Approved: 4/26/2012

CURRICULUM VITAE

Name: Susan Marie Mitchell.

Degree and date to be conferred: Ph.D., 2012.

Collegiate institutions attended:

University of Maryland, Baltimore County
September 2004 – May 2012
Ph.D., Information Systems.

University of Maryland, Baltimore County
September 2004 – May 2009
M.S., Information Systems.

The Johns Hopkins University
June 1980 – July 1983
M.S., Computer Science.

University of Maryland, Baltimore County
September 1976 – December 1978
B.A., Psychology.

Montgomery College
September 1974 – May 1976
A.A., General Education: Science and Mathematics.

Professional Publications:

Mitchell, Susan M. and Carolyn B. Seaman, "Software Process Improvement through the Identification and Removal of Project-level Knowledge Flow Obstacles," 34th International Conference on Software Engineering (ICSE 2012), Zurich, Switzerland, June 2012 (publication pending).

Mitchell, Susan M. and Carolyn B. Seaman, "A Knowledge Mapping Technique for Project-level Knowledge Flow Analysis," 5th International Symposium on Empirical Software Engineering and Measurement (ESEM'12), Banff, Alberta, Canada, September 2012.

Mitchell, Susan M. and Carolyn B. Seaman, "A Comparison of Software Cost, Duration, and Quality for Waterfall vs. Iterative and Incremental Development: A

Systematic Review," 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM'09), Lake Buena Vista, USA, October 2009.

Mitchell, Susan M., Sreedevi Sampath and Viviane Malheiros, "Leveraging Knowledge Management During Software Product Development," Consortium on Computing Sciences in Colleges East (CCSC-E'08), Frederick, USA, October 2008.

Mitchell, Susan M. and Wayne G. Lutters, "Assessing the Value of Computer Science Course Material Repositories: A Work in Progress," Consortium on Computing Sciences in Colleges East (CCSC-E'07), Fredericksburg, USA, October 2007.

Mitchell, Susan M. and Wayne G. Lutters, "Assessing the Value of Computer Science Course Material Repositories," Conference on Software Engineering Education and Training (CSEE&T'06), Turtle Bay, USA, March 2006.

Professional positions held:

2000 – present	University of Maryland, Baltimore County Baltimore, Maryland Lecturer, Computer Science.
1989 – 1999	Montgomery College Rockville, Maryland Associate Professor, Computer Science.
1/88 – 12/88	Singer, Link Simulation Systems Division Silver Spring, Maryland Systems Engineer.
11/86 – 12/87	Communications and Systems Specialists, Inc. Burtonsville, Maryland Systems Analyst.
1/86 – 11/86	Independent Software Consultant.
7/79 – 12/85	Technology Service Corporation Silver Spring, Maryland Programmer/Analyst.

ABSTRACT

Title of Document: SOFTWARE PROCESS IMPROVEMENT THROUGH THE REMOVAL OF PROJECT-LEVEL KNOWLEDGE FLOW OBSTACLES: THE PERCEPTIONS OF SOFTWARE ENGINEERS.

Susan Marie Mitchell, Ph.D., 2012

Directed By: Dr. Carolyn B. Seaman, Information Systems Department

Uncontrollable costs, schedule overruns, and poor end product quality continue to plague the software engineering field. Innovations formulated with the expectation to minimize or eliminate cost, schedule, and quality problems have generally fallen into one of three categories: programming paradigms, software tools, and software process improvements. It is this last category of software process improvement (SPI) that is the concern of this research. Specifically, this research investigates SPI through the application of knowledge management (KM) at the software project level. KM is a technique that is used in many business domains as a means to capture, use, and build upon corporate knowledge assets. It can be viewed as complementary to SPI when used in a software development domain.

This research advances the use of KM in SPI by investigating *the use and flow of knowledge from the present in the present*. In particular, it examines the flow of knowledge within an individual software development project as it executes, looking for impediments to that flow. The implicit hypothesis within this approach is that the

removal of obstacles to project-level knowledge flow (K-flow) will enhance SPI. In particular, this research provides support for the

- application of KM at the individual project level for SPI,
- location and mitigation or removal of project-level K-flow obstacles for SPI,
- necessity for management of both explicit and tacit project-level knowledge,
- significance of the role of software engineers in project-level KM efforts, and
- value of a facilitation role in project-level KM.

The methodology used for this research was that of an exploratory case study of an industrial software development project. Data concerning project knowledge sources, sinks, flows, and context was gathered from professional software engineers and managers using semi-structured interviews. Using this data, a diagnostic project knowledge map (K-map) was constructed. The K-map was analyzed within a focus group of project software engineers for K-flow obstacles and potential solutions. Using these obstacles and solutions, questionnaires were formulated to probe software engineers' perceptions of the effect, if any, of the K-flow obstacle solutions on SPI. Findings reveal that software engineers perceive that the removal or mitigation of project-level K-flow obstacles generally reduces the time that it takes them to do their work, helps them to meet their deadlines, and improves their work quality, thus resulting in SPI.

SOFTWARE PROCESS IMPROVEMENT THROUGH THE REMOVAL OF PROJECT-LEVEL KNOWLEDGE FLOW OBSTACLES: THE PERCEPTIONS OF SOFTWARE ENGINEERS

By

Susan Marie Mitchell

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Dedication

For

Dad, Mom, Peggy, and Nancy

Once in a while, when it's good
It'll feel like it should
And they're all still around
And you're still safe and sound
And you don't miss a thing
'Til you cry when you're driving away in the dark

Singing, stop this train
I want to get off and go home again
I can't take this speed it's moving in
I know I can't
'Cause now I see I'll never stop this train

from "Stop This Train"
by John Mayer, 2008

Acknowledgements

There is no “right” time in one’s life to pursue a PhD. It will always be a difficult journey. The best one can do is to have the good fortune to be surrounded by the “right” people, people who understand you and support you. I have been thus blessed throughout my journey. I would first like to acknowledge the support, patience, tolerance, and guidance of my PhD advisor and mentor, Dr. Carolyn Seaman. (And she’s pretty darn smart, too.) I hope that the thin wall that was necessary between us as mentor and mentee will now disappear and that we will grow closer as both colleagues and friends. I thank my committee members, Dr. Wayne Lutters, Dr. Sreedevi Sampath, Dr. Victoria Yoon, and Dr. William Agresti for their support and agreeing to take a chance on me.

How many times did I say I was going to give up? Consult Dr. Wendy Carter-Veale and Dr. Renetta Tull of UMBC’s PROMISE Dissertation House program. (“I’m just saying …”) I also wish to express my appreciation to the two organizations, which unfortunately must remain anonymous, that gave me access to the subject projects for my research. And for your hospitality, wi-fi, and not kicking me out at closing time, thanks to Bun Penny, Panera Bread, Riverside Coffee, and the Clarksville Bagel Bin. The family members and friends who rallied around me, even when they thought I was crazy for doing this, are too numerous to mention, but I love you all. And last, but never least, thank you to my champion, my husband Eddie, and to my patient son, Cooper. I’m back!

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Software process improvement (SPI) is the systematic, continuous modification of a software development process for the purposes of reducing production costs, improving schedule adherence, and increasing software product quality. It is generally prescriptive in nature. That is, it is typically driven by a set of predefined improvement guidelines. These guidelines are used to adjust the process in an iterative manner, with measurements taken during, and analyzed after, each iteration to confirm that improvements in cost, schedule, and/or product quality are indeed occurring.

Knowledge management (KM) is a technique that is used in many business domains as a means to capture, use, and build upon corporate knowledge assets. It can be viewed as complementary to SPI when used in a software development domain. Indeed, knowledge captured from past software development projects for use in subsequent ones is common within SPI efforts. However, a vital, missing, component of KM for SPI is *the effective and efficient use and flow of knowledge from the present in the present, during software product development.*

This empirical study advances the use of KM for SPI by investigating not only the use and flow of knowledge from the past in the present, but also from *the present in the present*. In particular, it examines knowledge flow (K-flow) within an individual software development project *as it is carried out*, looking for impediments to that flow. The results of this study demonstrate that the mitigation or removal of such K-

flow obstacles has the compelling potential to generate SPI. The method used for the location of project K-flow obstacles is that of "knowledge mapping," or K-mapping (section 2.2.6). Study results also support the efficacy of the use of K-mapping, in the context of this study's design, for the location of project K-flow obstacles.

This research is exploratory, rather than confirmatory, in nature. Its major contribution is that it establishes a firm foundation upon which to pursue the actual effect of project K-flow obstacle mitigation and/or removal on SPI. Its exploratory nature is justified, as virtually no empirical research has been conducted in this area. Specifically, it substantiates the

- application of KM at the individual project level for SPI,
- location and mitigation or removal of project-level K-flow obstacles for SPI,
- necessity for management of both explicit and tacit project-level knowledge,
- significance of the role of software engineers in project-level KM, and
- value of a facilitation role in project-level KM.

1.1 Background

This research is undertaken in the context of the uncontrollable costs and schedule overruns that continue to plague the software engineering field. Co-existing and intertwined with these critical concerns is the challenge of producing a quality software product. These problems are common to all areas of engineering, but are

undoubtedly complicated within software engineering by the nebulous nature of software itself. For decades, multitudes of innovations intended to address these difficulties have been introduced and implemented. Unfortunately, evaluation of their effectiveness is sketchy at best, given the scarcity of reported empirical studies in the software engineering field.

Innovations formulated with the expectation to minimize or eliminate cost, schedule, and quality problems have generally fallen into one of three categories: programming paradigms, software tools, and software process improvements. For example, in the 1960s and 1970s, structured programming and top-down program design were introduced as paradigms aimed at reducing program complexity. Reducing complexity was hoped to result in increased coding efficiency and simplification of debugging and maintenance. Later, object-oriented programming was touted as a superior, more natural paradigm for use towards these ends.

Numerous software tools have been integrated into the software development life cycle (SDLC) over the years. These tools have been aimed at easing the tasks performed by roles from project manager, to requirements analyst, to coder and tester. They include computer-aided software engineering (CASE) tools, debuggers, versioning control systems, integrated development environments, test automation software, and so on.

Since the 1970s, when Royce documented what has become known as the "waterfall" approach to software development (Royce 1970), software process improvement (SPI) has received a great deal of attention as a feasible approach to resolving cost and schedule overruns while increasing product quality. The catalysts for SPI initiatives are numerous. For example, there may be a particular process problem that must be solved. Perhaps schedule slippage during coding has been a recurring theme among projects. SPI may also be pursued because a company wishes to optimize its process for competitive reasons, such as being able to advertise their ability to produce a high-quality product at a reasonable cost. Or it is possible that particular organizations will not contract to a company that has not proven that its software development process has reached a specified level of "maturity."

There are several general manners in which SPI may be implemented. The adoption of an entirely new process is one option. However, such a radical approach brings with it much up-front investment in cost, time, and training, at a minimum. It would also be a very risky undertaking. The new process may not ultimately perform better than the old. A less risky, less disruptive SPI approach would be to identify, analyze, and adjust process problem areas. Adjustments could be made in an incremental fashion, working on one problem area at a time. Additionally, each problem area could be modified in an iterative manner. Performance metrics could then be kept for each modification iteration and compared with those of the previous one.

SPI initiatives do overwhelmingly tend to be incremental and iterative in nature. As such, they normally involve the recording of pre-defined quantitative metrics to verify that improvement is indeed taking place. This results in an improvement feedback loop, as shown in Figure 1.

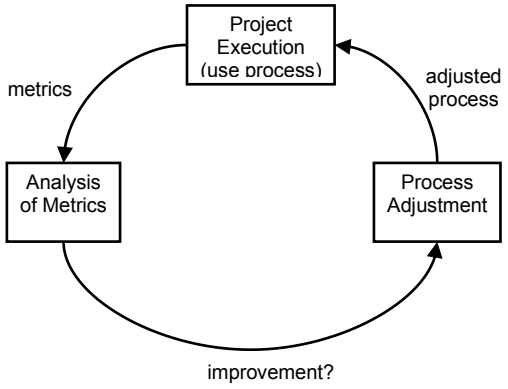


Figure 1. SPI Feedback Loop

Process improvements come in many forms and range from very high-level to very low-level. An example of a high-level improvement would be the gradual movement from a traditional, basically sequential development process model to one that encompasses some form of iteration. A lower-level improvement might be the honing of a code inspection process in order to increase error detection rate. Even lower could be a change in a developer coding habit, such as requiring developers to code in pairs rather than in isolation.

This research involves the application of the field of knowledge management (KM) to software development projects for SPI purposes. In particular, it is concerned with the

detection and removal of impediments to project knowledge flow (K-flow), thus improving project efficiency, schedule adherence, and work quality. Note that this research deals with project *knowledge*, rather than data or information. Data are discrete, objective facts. For example, the number of defects located during a code inspection and the length of time that it took to remove the defects are data. They are data because they do not inform. They just simply exist. Now, if a programmer were told that it was found in previous code inspections on this project that the greater the complexity of the code module inspected, the greater the number of defects found, he or she would be receiving information. The programmer is being enlightened about a particular characteristic of the project he or she is working on. However, he or she still does not have the ability to act upon the situation. *Knowledge is the key to "actionability."* (Nissen 2002) For example, one programmer may share with another programmer his or her experience with his or her own project code that dividing the code into smaller, less complex modules reduces the overall number of defects found in inspection because the individual defects in each module are easier to find and fix before inspection. The former programmer has now shared *knowledge* with the latter programmer, who will be able to take action. Note that this example not only illustrates the use of knowledge as opposed to data or information, but that it also shows the value of knowledge that is created, shared, and used within and by the project's software engineers. Additionally, it demonstrates the complexity of problems that may be solved through the use of project knowledge, as opposed to the types of problems that may be solved using typical project information and data such as schedule information, document templates, and project metrics.

Note also that this research involves the investigation of the flow of both explicit and tacit knowledge. Explicit knowledge is knowledge that is easy to articulate and codify, such as the common causes and fixes for certain code bugs. Tacit knowledge, however, is not easy to formalize. (Davenport and Prusak 1998) For example, one's personal code debugging strategy evolves over time and is strongly tied to one's experiences. Documenting such a complex item is likely to result in an incomplete, possibly inaccurate, picture, compromising the potential of the reuse of the knowledge. A better method for transferring this tacit knowledge might be to have experienced debuggers work one-on-one with less experienced debuggers. The knowledge is likely to be more fully and accurately transferred through this "personalization" process.

KM is a practice that has been formally employed in industry since about the 1980s. It stems from the appreciation that knowledge has become a primary, if not the primary, asset of modern businesses, providing them with their strategic advantage in the marketplace. As such a vital commodity, businesses have integrated procedures for managing their knowledge assets into their formal management routines. KM includes methods to create, acquire, codify, store, distribute, share, and build upon a business' internal knowledge. Benefits include, among other things, the preservation of knowledge associated with employee turnover, the reuse of technologies and processes, the fostering of innovation and creativity, and the increase of employee and group knowledge, skills, and decision making abilities. (Wiig 1997)

Organizations that have integrated KM into their management practices acknowledge that KM is unlikely to succeed without proper leadership. Thus, many companies have created dedicated roles such as Chief Knowledge Officer (CKO) to implement and oversee their strategic knowledge plans. (Liebowitz 1999) Among the major CKO responsibilities are the design, implementation, and oversight of the organization's knowledge infrastructure, fostering organizational knowledge creation and use, and acting as the chief advocate or "evangelist" for knowledge. (Davenport and Prusak 1998) Note that this is a dedicated role, that KM is acknowledged as an undertaking requiring specific skills and expertise.

1.2 Motivation

The purpose of this section is to present the factors that compelled us to undertake this particular dissertation study. The factors motivate the study's emphasis on knowledge flow at the project level for SPI.

1.2.1 Reliance on Knowledge at Rest

KM efforts in software engineering are greatly skewed towards the use of technology, chiefly for the storage and retrieval of knowledge. (Bjornson and Dingsoyr 2008) This is natural in a field that defines itself in terms of technology. But however vital, technology simply facilitates KM. It is not a KM driver; it is a tool. *It does not facilitate knowledge flow; it only accommodates knowledge "at rest."* Expanding on this point, KM in software engineering is also heavily dependent upon the generation

and use of codified, or explicit, knowledge. A strong theme common among the 32 empirical studies in the literature review in section 2.4 is the need to *not focus exclusively on explicit knowledge, but also on tacit knowledge, when implementing KM in a software engineering domain*. This sentiment is consistent with the recent movement towards agile software development methods, which advocate the lessening of codification and the increase of person-to-person knowledge transfer, or "personalization." (Fowler and Highsmith 2001)

1.2.2 Lack of Project-level KM

Knowledge management has always been utilized, to some degree, for SPI purposes. For example, internal and external software documentation is a form of KM that has been encouraged for decades as a minimal necessity for debugging and maintenance purposes. The addition of this practice was a small, yet effective, method of SPI. A more sophisticated example of the use of KM for SPI is that of the "experience factory." (Basili, Caldiera et al. 1994) An experience factory is a logical and physical organizational unit that supports software project efforts by providing relevant experience from past projects. Project experience is continually analyzed, synthesized, packaged, and stored in a repository, and supplied upon demand. Although the experience factory concept has met with success, it is but one example of how KM for SPI relies heavily on collecting and storing knowledge from the present for use in the future, and the use of knowledge from the past in the present (Figure 2, left). As stated earlier, a *vital component of KM for SPI that is missing is the effective and efficient use and flow of knowledge from the present in the present,*

during software product development. The addition of such a practice not only has the potential for "real-time" SPI during product development, but the likelihood of enhancing the pool of knowledge for exploitation by future projects (Figure 2, right).

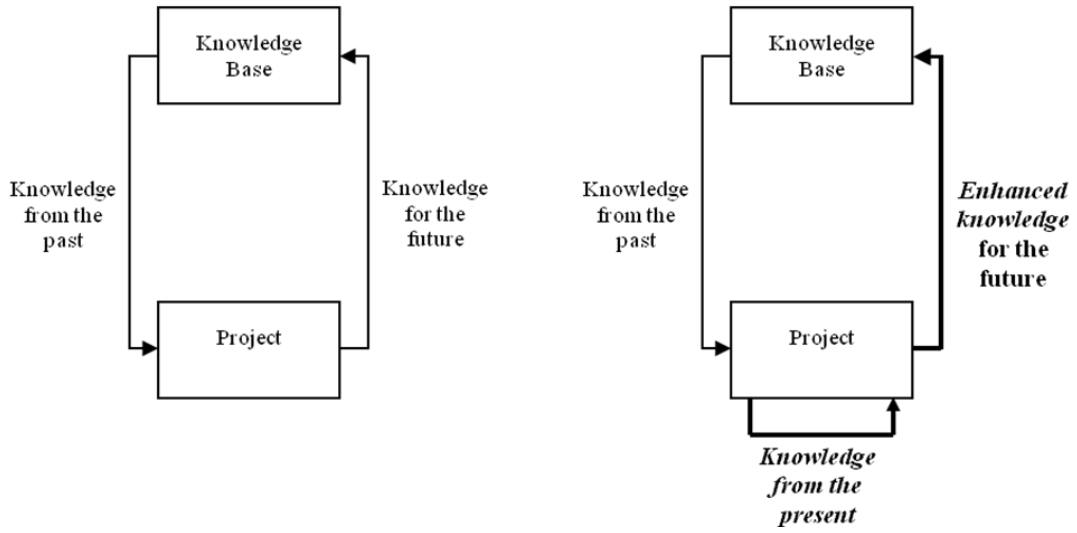


Figure 2. Project-level KM for SPI

The example in section 1.1 regarding the creation of knowledge by one programmer and shared with another is a typical example of knowledge that is created, shared, and used in the present. This effective creation and transfer of knowledge was possible because the source (the first programmer) and the target (the second programmer) of the knowledge were working within the same context. Thus, both were able to relate to the situation at hand, that of finding large numbers of defects in complex code modules. Additionally, the sharing, transfer, and ability to act upon the knowledge within a timely manner were enhanced because of the logical proximity of the knowledge source and target. Even if the programmers were geographically distributed, the path would have been within the same logical proximity and the

length of time for the flow of knowledge would only have been minimally increased by the need to use some electronic means of communication.

As the above example demonstrates, KM efforts in software engineering need to assimilate a holistic KM style that nurtures and supports a KM culture at every organizational level, *including the project level*. The importance of KM at the project level is best expressed by the following excerpt regarding knowledge creation from Nonaka and Takeuchi's book "The Knowledge Creating Company." (Nonaka and Takeuchi 1995)

> As we have pointed out, knowledge is created only by individuals. An organization cannot create knowledge on its own without individuals. It is, therefore, very important for the organization to support and stimulate the knowledge-creating activities of individuals or to provide the appropriate contexts for them. Organizational knowledge creation should be understood as a process that "organizationally" amplifies the knowledge created by individuals and crystallizes it at the group level through dialogue, discussion, experience sharing, or observation.

1.2.3 Lack of a Hands-on Perspective for SPI

Knowledge creation is but one aspect of KM. However, knowledge creation is dependent upon all other aspects, such as knowledge acquisition, sharing, and use. The findings from a multiple case analysis by Arent and Norbjerg (Arent and

Norbjerg 2000) suggest that *the project group is the primary source of knowledge about practice* and, therefore, is the natural stage for trying out and learning about new SPI practices. However, the literature review in section 2.4 uncovered only two empirical studies (Salo 2005; Rodriguez-Elias, Martinez-Garcia et al. 2009) that address KM for SPI by specifically utilizing the perspectives of project team members. It follows that this research adds to the literature by using the perspectives of the people "in the trenches." We believe that it is these people, the software engineers, who are most likely to have the awareness necessary to significantly contribute to SPI.

1.3 Research Questions

We find it sensible to divide the research questions for this study into three categories. The first category's questions strive to establish the resources and K-flows that project software engineers encounter and deal with on a daily basis, in addition to any resource or K-flow difficulties that they must cope with. The question in the second category explores the SPI achieved, if any, using the process of project-level K-flow analysis. The last category investigates the evidence, if any, that exists for establishing a dedicated project-level KM role.

The specific research questions for this dissertation study are as follows.

I. Current State of Project-level KM

1) What knowledge resources are available to software project engineers?

2) What are the knowledge flows within a software project?

3) What is the context within which knowledge flow occurs?

4) What problems are associated with project knowledge flow?

II. SPI Benefits

5) In what ways could project-level knowledge flow analysis benefit SPI?

III. Project KM Oversight and Facilitation

6) What evidence exists to justify the creation of a dedicated project KM role?

Chapter 2: Review of the Literature

This chapter presents a review of the literature that provides the driving factors for the study's overall theme of software process improvement (SPI) through the use of project-level knowledge flow (K-flow) analysis and its corresponding research questions (section 1.3). The literature review was conducted by the dissertation author, with oversight and guidance by her advisor. The author will, hereafter, be referred to as "the researcher." As illustrated in Figure 3, the major literature review topics include software process improvement, knowledge management, the manner in which knowledge management is applied in business settings, and the state of knowledge management as it applies to the field of software engineering. The intersections of these four reviews represent the basis for the study's research questions. Each of the review areas is discussed in detail in the following sections.

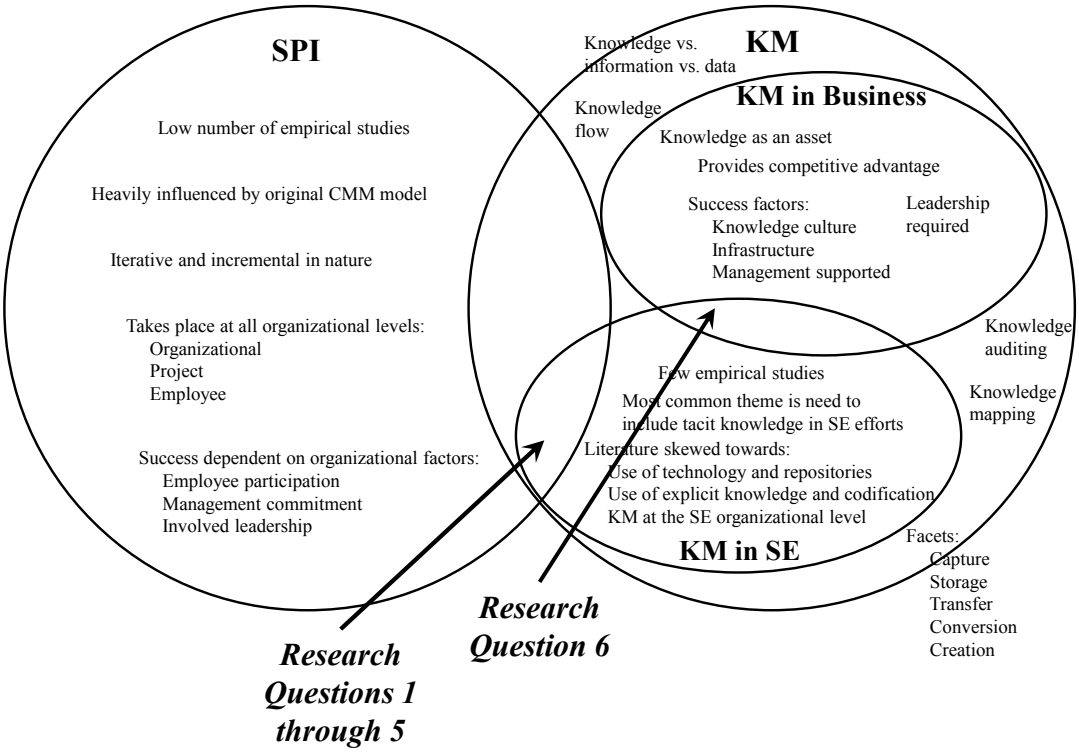


Figure 3. Supporting Literature for Study Theme and Research Questions

2.1 Software Process Improvement

2.1.1 Background

The use of software development processes has existed for decades. Royce (Royce 1970) is generally recognized as the first person to explicate a software process when he documented what has become known as the "waterfall" or traditional approach to software development. This approach relies on the up-front development of the bulk of system requirements and design and is principally sequential in nature, allowing little backtracking to prior software development life cycle (SDLC) phases. Since that time, many processes have incorporated iteration over and within phases and incremental product development and delivery. However, throughout the course of

process change and evolution, software development has always been plagued with cost and schedule overruns and compromised product quality.

Software process improvement (SPI) is the systematic, continuous modification of a software development process for the purposes of reducing production costs, improving schedule adherence, and increasing software product quality. SPI has its roots in the industrial total quality management (TQM) movement. TQM is a business management strategy that strives to make the best use of all available resources and opportunities by constant improvement. (Peratec 2009) The current state-of-the-art in quality management has been shaped by quality gurus such as W. Edwards Deming (Deming 1982), J. M. Juran (Juran and Gryna 1988), and P. B. Crosby (Crosby 1996), among others. (Dyba 2005)

In 1984, the United States Department of Defense established the Software Engineering Institute (SEI) to work with and support government, academic, and industrial software engineering pursuits in the area of SPI. The SEI is a non-profit research and development center whose mission is to help to improve organizations' software engineering capabilities and to develop or acquire the right software, defect free, within budget and on time. (SEI 2009) As such, the SEI is one of the world's leaders in SPI.

As described in Chapter 1, SPI is generally driven by a set of predefined improvement guidelines, is iterative and incremental in nature, and is applicable to all

levels of a software development organization, from the individual level, through the project level, and up to and including the organizational level. The most well-known, and likely the most widely used, SPI approach is the Capability Maturity Model (CMM), recently upgraded to the Capability Maturity Model Integrated (CMMI). The CMMI is a prescriptive process improvement model for systems engineering, software engineering, hardware engineering, and integrated teams. Its goal is to help an organization's processes to "mature," progressing from no defined process (Level 0) to the optimum process level (Level 5), where process performance results from across the organization are studied to locate and resolve common causes of problems in how the process is used. (Kulpa and Johnson 2008) Other prominent SPI approaches include BOOTSTRAP, TAPISTRY, IDEAL, SPICE, and the previously mentioned (Chapter 1) Experience Factory. (Hansen and Kautz 2004)

2.1.2 The SPI Literature

Hansen et al (Hansen, Rose et al. 2004) have conducted a literature review to describe the character of the SPI field. To qualify for inclusion in the review, a document had to include the term "software process improvement" in its title, abstract, or keywords and in addition have relevant contents. The review located and classified 322 documents according to whether the document's goal is prescriptive (telling SPI professionals what to do), descriptive (a report of an actual instance of an SPI program), or reflective (theoretically analytical). The results show that the SPI literature is dominated by prescriptive documents (42%), followed by purely descriptive (23%), descriptive and prescriptive (13%), reflective (6%), and, lastly, descriptive and reflective (2%). As one can see, the literature is highly prescriptive

and non-reflective. The unfortunate implication, as discussed in the literature review's conclusions, is that only a small minority of the SPI literature consists of empirical studies. A second finding is that the SPI literature is heavily influenced by the original CMM model (28% of the documents). Moreover, many of the other documents were inspired by CMM or other SEI contributions. A final finding is that descriptive reports of the application of SPI methods are greatly skewed towards success. As the authors speculate, this makes sense, as organizations are unlikely to make public their failures. In summary, the literature in the area of the available SPI models and methods, how well they work, and why they work or fail, is unfortunately, lacking in evidence-based studies conducted by qualified researchers. It is also biased by what organizations promoting and/or using SPI wish to make public.

Empirical studies of key factors for SPI success do, fortunately, exist. Dyba (Dyba 2005) conducted a survey of 120 software and quality managers representing whole software intensive organizations and independent business units. He investigated the dependence of SPI success on six organizational factors: business orientation, involved leadership, employee participation, measurement, exploitation, and exploration. He found that all six factors are positively correlated with SPI success. Synopses of the factors are given below, in no particular order.

- Business orientation – SPI goals and actions should be aligned with explicit and implicit business goals and strategies.

- Involved leadership – Leaders at all organizational levels should be genuinely committed to and actively participate in SPI.
- Employee participation – Employees should use their knowledge and experience to decide, act, and take responsibility for SPI.
- Measurement – Organizations should collect and utilize quality data to guide and assess the effects of SPI activities.
- Exploitation of existing knowledge – Organizations should incorporate a learning strategy that involves the exploitation of existing knowledge.
- Exploration of new knowledge – Organizations should incorporate a learning strategy that involves the exploration of new knowledge.

Employee participation was identified as the factor with the strongest influence on SPI success, followed by business orientation. Overall, the importance of Dyba's study is that it empirically verifies the importance of organizational factors for SPI success. This is significant, given that the SPI literature focuses almost entirely on prescriptive software engineering tools and techniques (see above). Dyba suggests that rather than try to imitate technical procedures, organizations should focus SPI efforts on creating an organizational culture within which these procedures can thrive.

Niazi et al (Niazi, Wilson et al. 2006) performed a study to identify SPI critical success factors (CSFs) by 1) performing a survey of the SPI literature looking for CSFs, 2) conducting interviews within software producing companies to identify

perceived CSFs, and then 3) comparing the CSFs resulting from both techniques. From the SPI literature, they found that the most frequently cited SPI CSF is senior management commitment (66%), followed by staff involvement (51%), and training and mentoring (49%). More than 25% of the literature cited staff time and resources, creating process action teams, reviews, experienced staff, clear and relevant SPI goals, and the assigning of responsibilities as CSFs. Interviews identified senior management commitment and training (68% each) to be the most frequently cited CSFs. Other frequently cited factors were resources, staff involvement, experienced staff, and facilitation (all over 27%). Two new CSFs rated highly, those of promoting awareness of the long-term benefits of SPI among the higher management and the staff members of the organization (59%) and a defined SPI implementation methodology (35%).

Using the two data sets to arrive at a final list of SPI CSFs involved not only identifying CSFs that appeared in both sets, but also deciding if any of the CSFs unique to a single data set stand on their own. This was done by using the following criteria.

- If a factor is cited in the literature with a frequency of >= 30%, then it was treated as a CSF.
- If a factor was cited by the interview respondents with a frequency of >= 30%, then it was treated as a CSF.

The authors explain that a similar approach has been used by other researchers (Rainer and Hall 2002) with a 50% limit as the criterion for a critical factor. However, they reduced the limit to 30% in order to have a sufficient number of SPI implementation factors. They explain that the focus of their research is on the criticality of specific implementation factors, not simply on the comparisons between the two data sets.

The nine factors from the two data sets that were considered CSFs are creating process action teams, experienced staff, support from higher management, staff involvement, training, reviews, SPI benefit awareness, allocation of resources, and having a defined SPI implementation methodology. The first six of these factors are common to both data sets and above the 30% limit. The authors, therefore, have more confidence that these CSFs are vital to the successful implementation of SPI programs.

Comparing the results of the Dyba and Niazi et al studies is not straightforward. The Dyba study involved the investigation of organizational SPI factors only. Additionally, the six organizational factors studied were pre-defined. The Niazi study did not begin with any predefined factors, organizational or otherwise. However, note that all of the Niazi CSFs are in fact organizational factors, with the possible exception of having a defined SPI implementation. This further reinforces Dyba's conclusion regarding the importance of organizational factors for SPI success.

2.1.3 SPI Application Levels

SPI is often thought of by practitioners and others as a high-level, organizational activity. For example, the CMMI, along with other popular SPI models, forms the frame of reference for SPI. These models generally connote improvement at the organizational level. But SPI, even within the CMMI model, must and does take place at all organizational levels, from the individual software engineer, through the team-level, and up to and including the organizational level. For example, SPI might include the improvement of an individual software engineer's work habits, the removal of obstacles to team productivity, or the establishment of an organization-wide software engineering process group (SEPG).

As another example, Watts Humphrey's Team Software Process (TSP), a team-level development process, advocates the maintenance of a process improvement proposal (PIP) process, which defines how to capture and document the engineers' process improvement ideas (Humphrey 2000) as they work. The oversight of the PIP is one of the activities within the Quality/Process Manager role. PIPs are also used in Humphrey's Personal Software Process (PSP) (Humphrey 2005), which is a process intended for use at the individual software engineer level. In this case, a PIP is maintained by the individual engineer.

The point of the existence of SPI efforts at all levels is being made for the clarification of this dissertation research. This research will focus specifically on SPI efforts at the project level.

2.2 Knowledge Management

There is no lack of definitions for knowledge management (KM) in the literature. So, perhaps it is better to introduce KM by stating the commonalities among the definitions. KM involves, among other processes, the creation, capture, storage, transfer, and use of knowledge within an organization. The purpose of these efforts is to provide the employees of the organization with the knowledge that they need to maximize their effectiveness, thereby maximizing the organization's effectiveness. Simply put,

> Knowledge management is the practice of transforming the intellectual assets of an organization into business value. (Platt 1996)

Or, more pragmatically,

> KM is getting the right knowledge to the right people at the right time so they can make the best decision. (Petrash 1996)

KM is a relatively new concept, as the world has just in the past two to three decades entered the "knowledge revolution," where businesses are principally service and customer-oriented rather than strictly production and product-oriented. (Wiig 1997) KM became a noticeable topic of discussion as recently as the late 1980s to early 1990s and its concepts, implementation strategies, and tools are still changing. Even

as it evolves, KM is under adoption by businesses world-wide as they have realized the competitive advantage that knowledge provides. As of 2001, over 80% of global corporations had KM projects in progress. (Lawton 2001)

This section begins with a general discussion of knowledge, followed by the concepts of knowledge conversion and creation, knowledge capture, storage, and transfer, knowledge flow, and knowledge auditing. An understanding of these concepts is necessary for the ensuing discussion of knowledge mapping. Knowledge maps and knowledge mapping will be used heavily in the methodology for this research (Chapter 3).

2.2.1 Knowledge

An appropriate beginning to the discussion of knowledge management (KM) is to present what is meant by knowledge. It is helpful to start by distinguishing between the concepts of data, information, and knowledge. The descriptions in this document are based on the meanings given to these terms by Davenport and Prusak. (Davenport and Prusak 1998) Data is a set of discrete, objective facts about events. For example, a trip to the grocery store can be described by data. The number of items that were purchased, the cost of each item, and the total bill are all data. But this data by itself tells nothing about why the particular items were purchased or whether the buyer was satisfied with his purchases. The data are simply a set of facts. They do not inform. The ability to inform is the major distinction between data and information. Information has relevance to its receiver and changes the way that he perceives something. Telling the grocery store manager that the prices of the items that were

purchased were well above those in a neighboring store is information, as it has meaning and relevance for the manager. However, this information does not provide the manager with what he/she needs to act on the situation. The enabling of "actionability" is the major difference between information and knowledge.

Knowledge is a more nebulous concept than either data or information. As such, this paper will rely on Davenport and Prusak's exact definition:

> Knowledge is a fluid mix of framed experience, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of knowers. In organizations, it often becomes embedded not only in documents or repositories but also in organizational routines, processes, practices, and norms.

For example, when an apprentice works under the tutelage of a master craftsman, the craftsman gradually transfers his knowledge to the apprentice. It may not be precisely clear why we know that such a situation is effective for training the apprentice, but we intuit that simply codifying the craftsman's skills and experiences, as in a document, or having the apprentice only observe the craftsman would not endow him with the full know-how of his mentor. What the craftsman needs to impart to his apprentice is too rich. It is not data, nor information, but knowledge. It is both explicit and tacit knowledge. Explicit knowledge can be easily communicated, codified, and

stored for future use. For example, the craftsman may record the materials that he uses for the construction of a particular item, or some procedures that he uses may be simple and concrete enough to set down in writing. However, much of the knowledge that he needs to transfer to his apprentice is tacit; that is, knowledge that is not easily articulated or formalized. Over the years, this knowledge has become a part of the craftsman. It can only be conveyed via full interaction with his apprentice.

Figure 4 summarizes the dependencies between data, information, and knowledge. Notice that while data is in abundance compared with knowledge, it is knowledge that is the superior source for "actionability." That is, knowledge provides the ability for an individual or organization to convert what it knows into action. It is by this action that the individual and the organization profit.

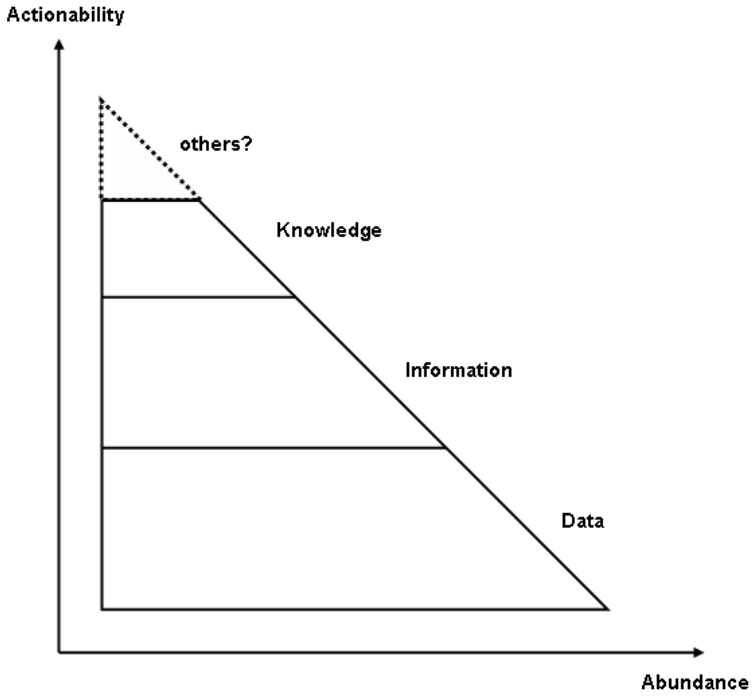


Figure 4. Knowledge Hierarchy
(Nissen 2002)

2.2.2 Knowledge Capture, Storage, and Transfer

As explicit and tacit knowledge are of vastly different compositions, the methods used to capture, store, and transfer them vary accordingly. For example, it is common to capture explicit knowledge in formats such as text and diagrams and store them in databases to which common access can be given. However, the capture and storage of tacit knowledge is not so straightforward. The codification of tacit knowledge comes at great expense. Much of the knowledge will be lost or minimally muddled through the codification process. This is not to say that an attempt should not be made to codify tacit knowledge. In fact, it is imperative in many situations, such as when trying to retain the internal knowledge of a key employee just before he or she leaves a company.

However, in the day-to-day operations of an organization, a more effective way to manage tacit knowledge may be through personalization. (Hansen, Nohria et al. 1999) Personalization is the sharing of knowledge through direct person-to-person contact. Examples include workshops, tutorials, one-on-one discussions, and informal "water cooler" conversations. Here, tacit knowledge is captured and stored in the minds of the receiving individuals. Although there is still the possibility of knowledge loss or misunderstanding, personal interaction allows questioning, verbal and pictorial clarification, and is accompanied by informative cues such as tone of voice and body language.

Both explicit and tacit knowledge constantly move throughout organizations, even without any formalized KM strategy in place. However, it is important to make the distinction between knowledge that is transmitted from one person or place to another and that which is "transferred." Transmitted knowledge simply moves, whereas transferred knowledge not only moves, but is absorbed on the receiving end. (Davenport and Prusak 1998) The importance of this distinction is that knowledge absorption (transference) is a prerequisite for knowledge use, which is the ultimate KM goal.

2.2.3 Knowledge Conversion and Creation

For KM to be effective, knowledge needs not only to be transferred, but new knowledge needs to be continually created. Knowledge conversion is the process of creating new knowledge from existing knowledge. It is accomplished through the social interaction between tacit knowledge and explicit knowledge. (Nonaka and Takeuchi 1995) Table 1 illustrates the four modes of knowledge conversion as defined by Nonaka. (Nonaka 1994) An example of externalization, or the conversion of tacit knowledge to explicit knowledge, is when a person commits a personal experience to paper, creating new codified knowledge. Not as intuitive is socialization, or the "conversion" of tacit knowledge to tacit knowledge. For example, teacher-to-student knowledge transfer creates new tacit knowledge in the mind of the student. A common form of internalization is the tacit knowledge creation that takes place when one reads a book. Lastly, combination, or explicit to explicit knowledge conversion, involves combining and reworking different bodies of codified knowledge, creating new codified, or explicit, knowledge.

		To	
		Tacit Knowledge	**Explicit Knowledge**
From	**Tacit Knowledge**	Socialization	Externalization
	Explicit Knowledge	Internalization	Combination

Table 1. Four Modes of Knowledge Conversion

It is worth repeating that knowledge conversion does not simply mean converting from tacit to explicit knowledge or vice versa, although such a transformation may take place during conversion. Conversion is the creation of new knowledge by combining, modifying, and reworking existing knowledge.

2.2.4 Knowledge Flow

The term "knowledge flow" is used generously throughout the KM literature. However, it is common for it to be used without explicitly being defined. Additionally confusing is that the word "flow" is sometimes used as a verb and other times as a noun.

Flow is typically used as a verb whose implicit definition is the movement of knowledge from a person(s) or place to another person(s) or place. It is not always made clear whether the movement of the knowledge involves actual knowledge transfer (absorption). That is, if one person gives a document (explicit knowledge) to another person, is that considered a flow of knowledge, even though no transfer of knowledge has taken place? Or would the receiver need to read and absorb the

contents of the document in order for a flow to have taken place? Flow is generally used as a noun when an author is attempting to summarize the various movements of knowledge that take place within an organization, regardless of whether the movement is a transfer or simply a transmission. Each movement is a flow and is described or sometimes illustrated as a vector with its origin at the sending end of the knowledge.

A precise and unambiguous definition of knowledge flow (K-flow) is needed for the characterization of the methodology for this study (Chapter 3). Nissen's definition has been chosen for this purpose. (Nissen 2006) Nissen defines a K-flow as "a dynamic movement of knowledge between coordinates (e.g. between individuals or organizations, or points in space or time)." Note that he does not specify that a flow must result in a transfer (absorption) of knowledge at the receiving end.

2.2.5 Knowledge Auditing

Before one introduces a new organizational practice or modifies an existing one, it is wise to conduct and document a broad overview of the state-of-the-business in the particular area of concern. This overview provides a baseline from which one may acquire a thorough understanding of the current business practice. A common procedure used to obtain such a baseline when introducing or modifying a KM initiative is the "knowledge audit." According to Debenham and Clark (Debenham and Clark 1994), a knowledge audit is a procedure used to produce a report detailing the structural overview of a designated area of an organization's knowledge as well as

the characteristics of that knowledge. The report consists of a two-page executive summary highlighting the major findings of the audit, and a "block map" with corresponding "block proformas." A block map is a diagram showing all knowledge blocks (chunks of related knowledge), their interrelationships, and the physical knowledge repositories in which they reside. A block proforma is a schema describing qualitative (e.g. name, auditor, repository) and quantitative (e.g. the quality and redundancy of the block knowledge) information about a knowledge block. Each knowledge block has a corresponding block proforma.

Knowledge audits as defined by Debenham and Clark are a starting point, but they are descriptions of explicit knowledge stores and their associations only. Tacit knowledge stores (i.e. people) are omitted. Additionally, such audits are static in nature. That is, they do not take into account the movement, or flow, of knowledge from people to non-human entities (e.g. document repositories), from non-human entities to people, or between people. Given that K-flow is omitted, so is the context within which flow takes place.

Liebowitz et al (Liebowitz, Rubenstein-Montano et al. 2000) extend Debenham and Clark's knowledge audit model by offering the following questions to provide direction for a knowledge audit (adapted from Dataware Technologies, 1998).

- What knowledge do we have?
- What knowledge is missing?

- Who needs this knowledge?
- How will we use it?

Note that Debenham and Clark's model addresses only the first question regarding existing knowledge. Liebowitz's remaining three questions extend their model by adding the "people factor." That is, in order to answer these questions, one needs to address the knowledge needs of the business and, therefore, its people. Liebowitz identifies the following steps as a guide to locating the answers to the above questions.

1) Identify what knowledge currently exists in the targeted area
    a. Determine existing and potential sinks, sources, flows, and constraints in the targeted areas, including environmental factors that could influence the targeted area
    b. Identify and locate explicit and tacit knowledge in the targeted area
    c. Build a knowledge map of the taxonomy and flow of knowledge in the organization in the targeted area. The knowledge map relates topics, people, documents, ideas, and links to external resources, in respective densities, in ways that allow individuals to find the knowledge they need quickly.
2) Identify what knowledge is missing in the targeted area
    a. Perform a gap analysis to determine what knowledge is missing to achieve business goals

b. Determine who needs the missing knowledge

3) Provide recommendations from the knowledge audit to management regarding the status quo and possible improvements to the knowledge management activities in the targeted area

Notice that this K-audit model is dynamic in nature. For example, Step 1 mentions K-flows, environmental factors, tacit knowledge, and the creation of a knowledge map (discussed further in section 2.2.6). Steps 2 and 3 take Debenham and Clark's model even further by identifying problems with and providing possible improvements to existing organizational knowledge management practices. This dissertation research will incorporate all three of these steps within one portion of its overall methodology.

2.2.6 Knowledge Mapping

As with many KM concepts, the concept of a "knowledge map," or K-map, has many different meanings. However, all K-maps characterize in some fashion the knowledge within an organization or other entity. There is no specific or proper representation for a K-map. Its format and content are dependent upon its intended use. For example, if a company wishes to provide its employees with the means to locate other employees who may be able to answer their questions on particular topics, the company may create an online employee "yellow pages" that lists each employee's areas of expertise. A simple organizational chart is another form of K-map that may, to a degree, serve the same purpose. In section 2.2.5, Debenham and Clark's (Debenham and Clark 1994) "block map," a diagram of knowledge collections and

how they are related, is another example of a K-map. They suggest the use of this type of K-map when assessing (auditing) the current state of an organization's knowledge.

As would be expected, the term "knowledge mapping" means the construction of a K-map. As mentioned, the format and contents of a K-map are dependent upon its intended use. The purpose of this dissertation research is to explore knowledge flow within a software development project. The research methodology relies heavily on the construction and use of K-maps. An appropriate type of map to use is a "diagnostic map." Diagnostic maps are used "… to relate project situations specifically perceived as problems (anomalies, failures) to sources and to more general organizational or behavioral features". (Lanzara and Mathiassen 1985) The "project situations" for this research are K-flows. Examples of K-flow problems are the presence of resources, such as documents, that are never used, lessons learned repositories that are never accessed, flow bottlenecks, and missing flows. In order to identify potential K-flow problems, a rich diagnostic map should be drawn. The diagnostic K-maps used in this research will be modeled after the map constructed by Hansen and Kautz (Hansen and Kautz 2004) in their SPI research. Their rendition of a K-map was arrived at by combining the strengths of Rich Picture drawing (Monk and Howard 1998) and diagnostic mapping. (Lanzara and Mathiassen 1985) It contains the following three major components.

- Structural elements – The static components, or nodes, of the map. Typical structural elements are actors (people) and knowledge sources and sinks. They are visually represented by any symbol that is indicative of what they are in reality. For example, an actor is represented by a stick figure.
- K-flows – The dynamic components of the map that illustrate the interaction between structural elements. They are visually represented by unidirectional or bi-directional lines.
- Climate – The context within which knowledge flow takes place. Climate is visually represented by text within a "thought cloud" or other annotation.

The Hansen and Kautz K-map representation is very open-ended, allowing the use of symbols, text, and color as the mapmaker sees fit. The K-maps created during this research followed their lead.

2. 3 KM in Business

A major aim of this research is to explore the integration of knowledge management into the software development life cycle (SDLC) at the project level for the purpose of software process improvement (SPI). Knowledge management (KM) is a practice that has been utilized, to varying degrees, by businesses for as long as businesses have existed in order to improve their overall operations. This section explains the manners in which KM has been utilized in businesses, its benefits to businesses (e.g. maintaining intellectual capital, fostering innovation and creativity), the critical factors for its success, and the importance of a leadership role in a business's KM

endeavors. This information will help to establish the foundation for the inclusion of KM in software product development efforts.

So, why is it vital for an organization to manage its knowledge? According to Wiig, knowledge has been implicitly managed to some degree throughout time, and economies have been the driving factor in KM evolution. (Wiig 1997) Minimally, it has always been necessary to manage knowledge by passing certain skills down from generation to generation. This was true even when economies were largely agrarian. As the world moved into the industrial revolution, knowledge began to be recognized as an important asset, but largely among guilds and specialists. Economic emphasis was chiefly on the efficient production of manufactured goods and KM was still largely limited to the sharing and passing on of skills. Even with the introduction of IT in the second half of the 20th century, when countless people migrated from physical work to desk work, the value of mental labor was still not fully understood or acknowledged. It has only been since the beginning of the "knowledge revolution" in the early 1990's that organizations have begun to see their employees "… instead of being a replaceable commodity, as the fundamental capability behind their whole existence and success--the source of profitability and the driver for sustained viability." (Wiig 1997) We have become a "knowledge society," and it has become imperative for organizations to manage knowledge if they wish to maintain a competitive market advantage.

To illustrate the current wealth of KM products, consulting services, and even college degree programs, one need only search the Internet using the keywords "knowledge management" followed by one of these terms. KM industry and academic literature, magazines, organizations, and web sites are also abundant. Estimating a dollar amount for the value of the current KM services market is difficult, but Ovum, a global telecom, IT, and software consulting firm, reported in 1998 that the market was already $2.6 billion and Ovum was projecting a climb to $8.8 billion by 2004. (Woods 1998) Additionally, as of 2002, 33% of Fortune 1000 companies reported that KM activities were under way. (Bontis 2002) KM has become an accepted business practice.

2.3.1 Benefits

Organizational knowledge management facilitates competitive advantage in numerous ways. (Wiig 1997) Minimally, it

- maintains the organization's intellectual capital,
- preserves the knowledge loss associated with employee turnover,
- improves the reuse of technology and processes,
- facilitates the adoption of lessons learned,
- facilitates knowledge sharing among employees, increasing individual and group knowledge, skills, and decision making abilities,
- fosters innovation and creativity,

- motivates employees and improves morale, bringing about greater productivity and quality,
- adds competitive value to products and services by the application of direct or embedded human expertise, and
- enhances the ability to understand customer problems and needs.

These KM benefits in turn have contributed to the production of higher quality products, the more rapid delivery of products and services, improvement in the response to customer needs, and increased customer satisfaction. These outcomes are all undeniably major contributors to competitive advantage.

2.3.2 Implementation

The implementation of organizational KM is a major undertaking involving large investments of time and resources. And there is always the chance that the effort may result in only partial success or even fully fail. In order to maximize the chances for the success of a KM effort, critical factors must be addressed and suitable leadership assigned before implementation.

2.3.2.1 Critical Success Factors

A 2003 study by Alazmi and Zairi summarized the critical success factors (CSFs) found in the KM literature. (Alazmi and Zairi 2003) They define CSFs as "… points, areas, or goals that have to be given extensive attention and support by the management to achieve the mission, quality and high performance" of an endeavor.

CSFs were gathered from fifteen literary sources and grouped into nine categories. The categories are listed below from highest to lowest frequency of occurrence. Note that four of the categories have been combined, specifically technological infrastructure and knowledge infrastructure into infrastructure, and sharing and culture into culture, as they are complementary concepts for the purposes of this research. The categories, with brief descriptions, are as follows.

- Culture – the willingness, motivation, and ability for all employees to share knowledge and the evolution of an organizational culture for knowledge sharing, innovation, and overall KM acceptance
- Infrastructure – the technology, environment, and personnel necessary to support the overall KM strategy
- Top-management supported – buy-in and continued visible support by top management in the form of finances, resources, and conviction
- Knowledge strategy – the up-front planning of a clear approach to the implementation and continued support of the KM effort
- Training – the support and resources for employee and KM worker training and continued learning
- Creating – the integration of methods, motivation, and resources for creating new knowledge and building upon existing knowledge
- Transferring – the integration of methods, motivation, and resources for the successful transfer of knowledge

2.3.2.2 Leadership

Additionally, these CSFs require an individual or group to champion, plan, and oversee them to assure that they come to fruition and are integrated into the organizational culture. Examples of roles and organizational groups used to support KM are Chief Knowledge Officers (CKOs), knowledge project managers, knowledge reporters and editors, knowledge network facilitators, "coaches" to facilitate the effective use of technology, and committees to prioritize and set knowledge strategy. (Davenport, Long et al. 1997) As many KM experts indicate that leadership in the form of a CKO is vital to KM success (Liebowitz 1999), and familiarity with the role of CKO will be a concept needed for a later discussion, time will be taken to elaborate on this role.

Lawton describes a CKO as a senior-level executive who creates an infrastructure and cultural environment for knowledge sharing. (Lawton 2001) Basically, he or she is the overseer of the KM effort. Occasionally the role is dual in nature, with the additional responsibility of overseeing employee professional development. In this case, the role is usually referred to as a Chief Learning Officer (CLO). One additional distinction to be made is that a CKO is not the same as a Chief Information Officer (CIO), who is responsible for the organization's IT matters, not for KM. This discussion of KM leadership will be restricted to the role of CKO.

According to Davenport and Prusak (Davenport and Prusak 1998), the chief responsibilities of a CKO are to

- lead the development of knowledge strategy,
- design, implement, and oversee the organization's knowledge infrastructure,
- design and implement the organization's knowledge codification approaches,
- provide critical input to the process of knowledge creation and use around the organization,
- measure and manage the value of knowledge,
- manage relationships with external providers of information and knowledge,
- manage the organization's professional knowledge managers, and
- advocate or "evangelize" for knowledge and learning from it.

Moreover, the CKO is obligated to make the KM effort pay off economically.

The role of the CKO is dynamic and multi-faceted, requiring a go-getter approach. Successful CKOs are entrepreneurs, technologists, environmentalists, and consultants. (Earl and Scott 1999) As entrepreneurs, they are highly motivated and excited by business development and by growing something. They are energetic KM advocates. As technologists, they are able to understand when it is fitting to adopt a technology to support KM, evaluate suitable technologies, cope with implementation issues, and assess the success of the technology. As environmentalists, they facilitate conversations and interactions that encourage knowledge creation and exchange. This includes the physical design of space for both planned and chance meetings, arranging for experience-sharing events, and actively bringing together people with

common interests who rarely interact. Lastly, as consultants, CKOs introduce ideas and listen to other's ideas, assessing whether they are consistent with the organization's business model and goals.

As of 2002, 25% of Fortune 500 companies had CKOs, 80% had KM staff, and 42% anticipated appointing a CKO within the next three years. (Bontis 2002) The position has understandably become a part of corporate management culture.

2.4 KM in Software Engineering

Bjornson and Dingsoyr have conducted a systematic literature review on the topic of knowledge management in software engineering. (Bjornson and Dingsoyr 2008) A systematic literature review (or simply systematic review) is a procedure used to summarize and evaluate all accessible research literature pertaining to a particular research question, topic area, or phenomenon of interest. (Kitchenham 2004) Bjornson and Dingsoyr's systematic review enumerates and summarizes the knowledge management literature related to software engineering through the first quarter of 2006. Twenty-nine empirical studies and 39 reports of lessons learned were located. Empirical studies were restricted to those whose described tools or theories were tested in industry (e.g. studies using student subjects were excluded). This dissertation research literature review will rely only upon the 29 empirical studies, not the lessons learned reports.

To complete and update the literature review process, the researcher performed her own literature search, attempting to locate empirical studies relevant to the dissertation research themes of K-flow, K-audits, and K-mapping in software engineering. She conducted the search using the same databases used in Bjornson and Dingsoyr's systematic review. The results of the search include all relevant empirical studies through the first quarter of 2011. Only four such studies were located (Agostini, Albolino et al. 2003; Pandey 2003; Hansen and Kautz 2004; Rodriguez-Elias, Martinez-Garcia et al. 2009), one of which was a duplicate from the Bjornson and Dinsoyr review. Table 2 combines and summarizes the concepts and major findings of both Bjornson and Dinsoyr's systematic review and the researcher's literature search (32 unique documents), resulting in 30 unique table entries. The italicized studies in Table 2 are those that, to varying degrees, supplied the researcher with background upon which to build her research. References for these studies only are provided.

Concepts	**Major Findings**
Knowledge repositories	K-repositories are an approach to supporting risk in project management Users should be involved in K-repository development K-repositories are an approach to support design activities Benefits can be realized quickly, the repositories remain useful over time, and more benefits accrue over time K-repositories can be used for different kinds of knowledge than originally intended

Cartographic tools	A cartographic tool for allocating resources, searching for competence, identifying projects opportunities, and upgrading skills enabled learning practice at both the individual and company level
Automated KM tools	There are some main knowledge needs to be taken into account in the design of an automated KM system.
Managing software development process knowledge	KM use is feasible as underlying theory to supplement CMM *Regardless of KM approach used for SPI, both tacit (to change practice) and explicit knowledge (to create an organizational memory) must be created* (Arent, Norbjerg et al. 2000) *A techno-centric approach to SPI may impose unnatural work practices and fails to take into account SPI that might occur spontaneously within a community of practice* (Segal 2001) UP ensures large effects in learning, and improves communication and work distribution in companies Software process may be implemented in a beneficial and cost-effective way in small SW organizations
Managing knowledge through formal routines	Formal routines must be supplemented by collaborative, social processes

K-flow mapping	*A K-mapping technique can be used for SPI purposes by providing information about possible organizational K-flow problems* (Hansen and Kautz 2004) Causal maps for risk modeling contributes to organizational learning *The Tightly Coupled Engineering Team (TCET) process increases intra-team training and knowledge flow* (Pandey 2003) *The Software Process Engineering Metamodel (SPEM) can be adapted to be used as a process modeling language for analyzing knowledge flows in software processes* (Rodriguez-Elias, Martinez-Garcia et al. 2009)
Extracting knowledge via project reviews	Postmortems must be conducted in an environment for reflection, dialogue, criticism, and interaction *The knowledge and reasoning behind process improvements must be in an explicit format for organizational learning* (Salo 2005)
Implications of social interaction on knowledge sharing	*Tayloristic teams fail to effectively share knowledge among stakeholders mainly because of a purely codified approach* (Melnik and Maurer 2004) Increasing reflection level in mentor programmers can result in more double-looped learning
Use of networks	Networks should be used in addition to other activities when introducing new software engineering methods The role of networks in software engineering Networks built on existing information networks are more likely to be successful

KM strategy	*KM success factors include technological, organization, and human resource factors* (Feher and Gabor 2006) Ongoing interaction between different learning processes are important to the improvement of learning process practice *Both codification and personalization knowledge management strategies exist in software companies* (Trittmann 2001)
KM in general	*Knowledge pull leads to more effective KM than push* (Agila and Sun 2004) *Knowledge needs to be internalized to improve processes* (Ravichandran and Rai 2003) *Leadership is the most important enabler for KM* (Ward and Aurum 2004) *Perceived complexity, advantage, and risk contribute to the use of KM artifacts* (Desouza, Awazu et al. 2006)

Table 2. Major Findings from KM in SE Literature Review

General observations regarding the gaps in the body of work described by Table 2 are as follows.

- Empirical studies of KM in software engineering are few, which is consistent with the lack of empirical studies in the field of software engineering in general. (Kitchenham, Dyba et al. 2004)
- About 31% (ten) of the 32 total studies address the issue of KM in SPI in some manner. Eight of the ten studies look at KM in SPI from the organizational frame of reference. Only two studies investigate SPI in terms of

how knowledge is managed at the software development team or process level. (Salo 2005; Rodriguez-Elias, Martinez-Garcia et al. 2009)

- Only three of the ten studies addressing KM in SPI mention K-flow as it relates to SPI. One discusses a problem with the transfer of knowledge from the project team level to the organizational level. (Salo 2005) A second is fully devoted to the use of K-flow mapping and analysis for SPI at the organizational level. (Hansen and Kautz 2004) The third discusses the adaption of an existing software process engineering model (SPEM) to include knowledge flow analysis. (Rodriguez-Elias, Martinez-Garcia et al. 2009)
- Bjornson and Dingsoyr note that the concepts within the 29 studies in Table 2 that they located have very little overlap. Thus, no real body of evidence has been built to support any particular concept regarding KM in software engineering. They report that the only finding that is repeated over several studies is the need to not focus exclusively on explicit knowledge, but also on tacit knowledge when implementing KM.
- A noteworthy number of studies (25%) deal exclusively with the use of technology to manage knowledge. This is consistent with the view that KM in general was, until the past few years, still mainly focused on first generation KM. (Buono and Poulfelt 2005) That is, that KM is largely a technical issue that can be effectively addressed by capturing, storing, and sharing knowledge via technical means such as online repositories.

By far, the most valuable (to this researcher) document located was that by Hansen and Kautz. (Hansen and Kautz 2004) It describes an action research study whose main objective was to aid a software development company with their SPI efforts by identifying potentially problematic areas related to organizational K-flow. This study is valuable to this dissertation research in two ways. First, it demonstrates the successful use of K-flow mapping and analysis as a viable approach to SPI. Second, it introduces a straightforward and effective K-mapping technique. This dissertation research relies heavily on the SPI approach used in the Hansen and Kautz study as follows.

- As with Hansen and Kautz, it relies on employee interviews and focus groups as the major resources for data collection and validation in the construction of K-maps.
- It uses Hansen and Kautz's K-mapping technique for the drawing of its K-maps.
- It uses Hansen and Kautz's categories of characteristic K-flow situations (hubs, black holes, springs, and missing links) as a baseline for the identification of potential project K-flow problem areas.

A second study that is fundamentally relevant to this dissertation research presents an adaptation of the Software Process Engineering Model (SPEM) process modeling language that enables the analysis of knowledge flows in specific software processes. (Rodriguez-Elias, Martinez-Garcia et al. 2009) The authors have extended SPEM

using the Unified Modeling Language (UML) (Fowler 2003) to represent knowledge, knowledge sources, and knowledge flows within a software development process. They present a case study where the extended SPEM was applied to an ongoing software maintenance process, resulting in the identification of and proposed solutions to problems affecting process knowledge flow. These results provide a formidable basis for the examination of the application of K-flow analysis to specific software projects rather than processes.

There are many gaps in the literature regarding KM in software engineering and, more specifically, KM applied at the software project level. The major gaps are as follows.

- There is very little overlap in the topics covered and findings in the KM in SE literature. Studies do not build on each other.
- The theme most common within the literature is the need to include tacit knowledge in KM in SE efforts.
- The literature is skewed towards technology solutions to KM in SE, especially the use of knowledge repositories.
- The literature is skewed towards the use of explicit knowledge and codification techniques. Personalization is largely ignored.
- There are only two studies devoted to the application of K-flow mapping and analysis for SPI purposes. (Hansen and Kautz 2004; Rodriguez-Elias, Martinez-Garcia et al. 2009)

- The literature concentrates on the application of KM at the SE organizational level. There are only two studies of the application of KM at the project or process levels. (Salo 2005; Rodriguez-Elias, Martinez-Garcia et al. 2009)

2.5 Summary

The overarching goal of this research is to further software process improvement through the use of project-level knowledge management. Specifically, relative to the existing literature, it will add to the empirical research regarding the

- utilization of project-level KM for SPI,
- state of knowledge sources, K-flows, and K-flow context in industrial software development projects,
- identification and solution of problems associated with project-level K-flow,
- significance of tacit knowledge in software projects,
- value of exploiting the perspectives of project-level personnel (i.e. software engineers) for SPI,
- use of K-mapping and K-flow analysis as techniques for SPI, and
- need for a project KM leadership role.

Chapter 3: Methodology

3.1 Overview

The research strategy used for this study was that of an exploratory case study (Yin 2008), with the case being an industrial software development project team. A case study was chosen 1) to investigate the phenomenon of software development project-level knowledge flow, 2) to observe this phenomenon in an authentic context, 3) to build a theory regarding the effect of project-level K-flow obstacle removal on software process improvement (SPI), and 4) to develop propositions for further inquiry into the area of project-level knowledge management (KM). Mostly qualitative, with one quantitative, data gathering techniques were used. Qualitative methods consisted of document inspections, one-on-one semi-structured participant interviews, and a focus group. The quantitative method employed was that of a participant questionnaire.

The procedure followed for the conduct of this study was derived from the literature review (Chapter 2) and other published sources, guidance from research colleagues, and the feedback from the pilot study described below. This chapter explains the overall case study methodology and is organized as follows. To begin, the context of the pilot study is given, and the data collection methods and procedures followed described. Next, the pilot study results are presented and discussed, followed by study limitations. The results are then summarized and conclusions presented. Methodology

lessons learned from the pilot study to be applied to the industrial case study are given.

The chapter then proceeds on to the context and data collection methods and procedures followed for the industrial case study. The chapter concludes with a discussion of the case study validity concerns.

3.2 Pilot Study

The purpose of the pilot study was to assess the proposed research methodology in an authentic setting before applying it to the actual case study. The study was also utilized to answer the following open methodological questions.

- Should the concepts and examples of knowledge and knowledge management be included in the initial phone contact with participants, or should they be part of the initial interviews?
- Is the validation by the participants of their individual knowledge maps necessary?

3.2.1 Setting and Background

The pilot study took place within a medium-sized (approximately 12,000 students) U.S. public university. The subject software development team was culled from the business division of the university's information technology department. This

division routinely develops and maintains much of the university's internal business software systems.

The team was tasked with the installation, customization, deployment, and maintenance of one of three parts of an Enterprise Resource Planning (ERP) system intended to manage all university business operations. The particular third of the system assigned to the team deals with student operations and services such as student recordkeeping, student financials, and registration. This portion, or "sub-project," will hereafter be referred to as the ERP3 Project (not the actual project designation) and the team as the ERP3 Project Team. The ERP3 Project Team was originally composed of a combination of approximately 30 university technical employees, university staff, and industry consultants. The ERP3 Project ran from September 2007 through December 2009.

The technical manager of the ERP3 Project Team served as the gatekeeper for project background information and participant recruitment. Participant role diversity was a goal in order to approach the study's research questions from different perspectives, a methodology known as "triangulation." (Denzin 1978) Therefore, the commonly used software team roles of manager, analyst, developer, and tester were chosen. These designations were used in a very broad context. For example, an analyst could be a domain specialist, a requirements elicitation expert, or a business process analyst. However, it would serve no purpose for this research to take the roles to this level of granularity. We were able to recruit three analysts, four developers, two testers, and

two managers. One of the managers was the technical manager himself. (Analysts, developers, and testers will be collectively referred to as "software engineers" from this point on.) The technical manager also provided relevant organizational charts, as well as the project charter. The charter was very informative, giving a detailed description of the ERP3 Project's purpose and goals, timeline, project staffing, and project strategies.

3.2.2 Data Collection and Analysis Procedure

An overview of the data collection and analysis procedure followed for the pilot study is shown in Figure 5. The individual steps for data collection are described in detail below. All steps were conducted by the researcher (i.e. the dissertation author), with oversight and guidance by her advisor.

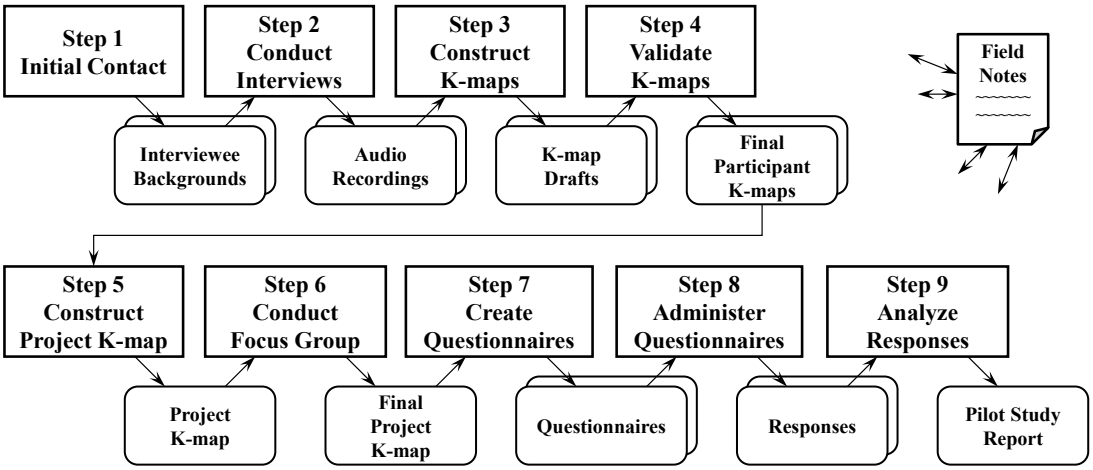


Figure 5. Pilot Study Data Collection and Analysis Procedure

3.2.2.1 Step 1: Initial Participant Contact

Initial contact with participants was via e-mail to schedule a brief introductory phone conversation. Each phone conversation consisted of

- giving a brief explanation of the dissertation research and the participant's anticipated activities in the research,
- gathering participant educational and professional background data,
- confirming the participant's role (e.g. analyst, developer) in the ERP3 Project, and
- arranging for an interview date and time.

Of the eleven participants, five were given only the brief dissertation research background mentioned above, while the remaining six were also given specific explanations and examples of knowledge and knowledge management. The purpose of this division was to determine if participants would be better able to understand and answer questions regarding knowledge and knowledge management during their interviews if they had been given some prior background.

3.2.2.2 Step 2: Participant Interviews

Twelve participant interviews were conducted from August 18, 2009 through October 1, 2009. Two managers, three analysts, four developers, two testers, and one database administrator (DBA) were interviewed. The DBA interview was not part of the original interview plan. However, the university's information technology

department's DBAs and their interactions with the ERP3 Team were mentioned so often during other participant interviews that the researcher thought it wise to conduct one DBA interview to enrich the pool of triangulation data.

The twelve semi-structured interviews took place in each participant's office and were audio recorded. The target length for an interview was 1 hour, 30 minutes. The actual lengths ranged from 1 hour, 15 minutes to 1 hour, 50 minutes. Each interview began with a brief introduction to the concepts of knowledge and KM, regardless of whether or not the participant had received the phone conversation explanations of these concepts. The researcher used a set of guideline questions during the interviews for consistency of question wording and to assure that all topics were covered. The guideline questions for software engineers are given in Figure 6. Those for managers are given in Figure 7. Notice that the manager questions are identical to those for the engineers, with the exception that the managers are not asked the questions relative to themselves, but relative to the engineers.

1. Could you briefly fill me in about the software development process that is used on the ERP3 project?
2. Tell me about how communication works within the project team.
3. Could you tell me about the tasks that you perform for the ERP3 project?
4. Tell me about your knowledge needs on a typical day for you on the project.
5. Can you think of a time recently when you needed some particular knowledge while working, but had trouble locating it?
6. Are there people on the project team that you consider "knowledge gurus?" That is, they seem to know something about everything.
7. Are there people on the project team that you consider "knowledge facilitators?" That is, they know how to get the knowledge people need or direct them to it.
8. I've been thinking about whether having a designated "knowledge facilitator" role would help a project team to be more efficient. I was wondering about your thoughts on this idea.
9. Is there anything else that you would like to share with me about the use or flow of knowledge within the project?
10. Is there anything else about the project in general that you would like to share with me?

Figure 6. Pilot Study Software Engineer Interview Guideline Questions

1. Could you briefly fill me in about the software development process the team follows to implement the ERP3 system?
2. Tell me about how communication works within the project team.
3. Could you tell me about the tasks that you perform for the ERP3 project?
4. Tell me about the knowledge needs of some of the team members on a typical day on the project.
5. Can you think of a time recently when a team member needed some particular knowledge while working, but had trouble locating it?
6. Are there people on the project team that you consider "knowledge gurus?" That is, they seem to know something about everything.
7. Are there people on the project team that you consider "knowledge facilitators?" That is, they know how to get the knowledge people need or direct them to it.
8. I've been thinking about whether having a designated "knowledge facilitator" role would help a project team to be more efficient. I was wondering about your thoughts on this idea.
9. Is there anything else that you would like to share with me about the use or flow of knowledge within the project?
10. Is there anything else about the project in general that you would like to share with me?

Figure 7. Pilot Study Manager Interview Guideline Questions

3.2.2.3 Step 3: Audio Transcription and Participant K-map Construction

The purpose of audio transcription of participant interviews was to extract all data associated with knowledge flows involving the ERP3 Project software engineers. Transcription began in September of 2009. For each interview, the researcher listened to the audio and simultaneously drew and annotated each relevant K-flow discussed. The result was a complete participant K-map. Due to unforeseen circumstances unrelated to the pilot study itself, only four of the twelve interviews were able to be converted to K-maps. These four maps, therefore, form the basis of the rest of the

analysis. Fortunately, diversity of participant roles was maintained, as the maps were generated from interviews with a manager, an analyst, a developer, and a tester.

3.2.2.4 Step 4: Participant K-map Validation

The next step was to validate the participant K-maps to ensure that the researcher had accurately captured all K-flows and associated data from the interview sessions. (Lincoln and Guba 1985) The necessity and accuracy of this step was questionable at this point. The researcher, therefore, planned to validate only half of the maps and observe the factors, if any, that might impact the quality of the study. Validation sessions took place in each participant's office and were audio recorded.

The first participant K-map generated, which was from the first interview (a developer), was validated on July 21, 2010. The researcher arrived with her hand-drawn interviewee K-map and a corresponding "skeleton" K-map (knowledge sources and sinks only) drawn on a 2' by 1.5' portable whiteboard (Figure 8). The overall layout of the skeleton K-map delineates the boundaries between the ERP3 project team (the innermost outlined area), the ERP3 project itself (the area directly surrounding the project team), the university's IT department, the university itself (name masked in the figure), and the "outside world." Humans are drawn as stick figures. Knowledge sources and sinks are drawn using symbols such as disk drives and documents. Missing resources that the participant had mentioned during his/her interview are included in the lower right-hand corner of the board.

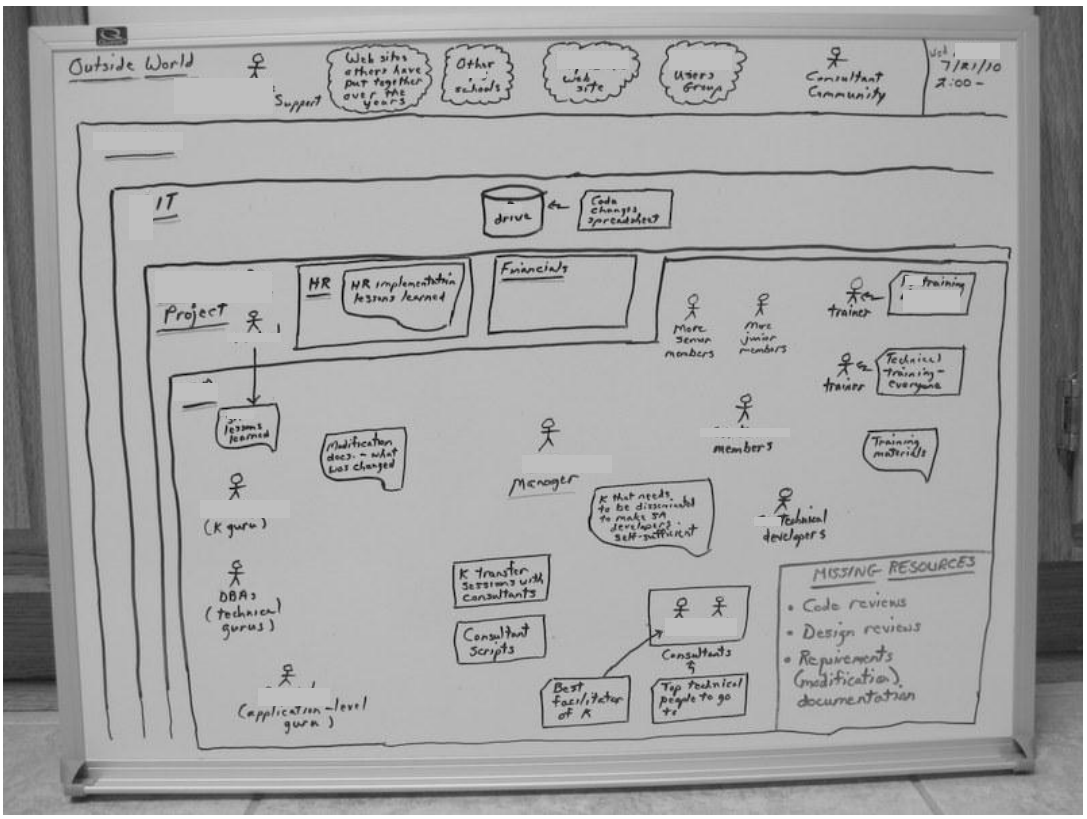


Figure 8. Participant K-map at Start of Validation Session (Sanitized)

The researcher first explained the structure and notation of the K-map to the participant. Together, they went over each K-flow, one by one, that the researcher had on her hand-drawn K-map, adding these K-flows, if appropriate, to the larger whiteboard version, and fleshing them out (e.g. adding K-flow context). As the process continued, it became obvious that the participant became increasingly more aware of the distinction between information and knowledge, as several flows were discarded as being information rather than knowledge flows after the participant had a chance to think things over. The validated K-map corresponding to the skeleton map in Figure 8 above is shown in Figure 9.

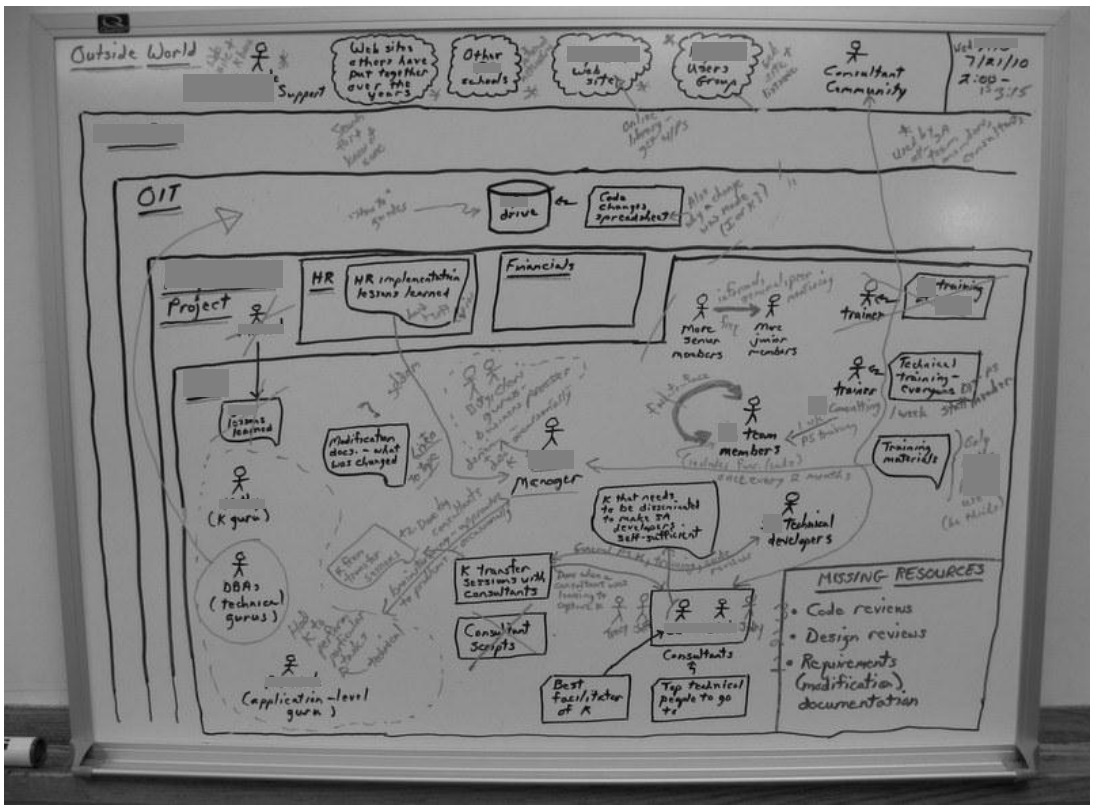


Figure 9. Participant K-map at End of Validation Session (Sanitized)

At the end of this first validation session, it was abundantly clear that the accuracy of the resulting K-map was greatly increased. Validation 1) facilitated the removal of resources and flows that were actually associated with information, rather than knowledge, 2) filled in missing knowledge sources, sinks, K-flows, and context information, 3) made the meaning of knowledge clearer to the participant, and 4) helped the researcher to further understand the project, the participant's role on the project, and the K-flows as perceived by the participant. She, therefore, performed this validation step for all four participant K-flow maps.

3.2.2.5 Step 5: Project K-map Construction

The consolidation of individual participant K-maps into a single project K-map was a tedious but straightforward process of the researcher sketching the project map by hand, checking off the items from the individual maps as she went along. Recall that four participant K-maps were generated, one for a manager, one for an analyst, one for a developer, and a last for a tester. As the manager's map was to be used for triangulation purposes only, its data was not included in the project K-map. After the final project K-map was created, it was then blown up to a 4' by 3' poster to be used in the focus group session.

3.2.2.6 Step 6: Project K-map Focus Group

A focus group was conducted on August 4, 2010 in order to analyze the project K-map. It took place in a conference room in the same location where the ERP3 Project Team members were housed. The purpose of the focus group was to identify bottlenecks in and obstacles to K-flow, the theory being that the removal of such bottlenecks and obstacles would improve the ERP3 Project software process in terms of time, schedule, and product quality (i.e. result in SPI). Possible solutions to K-flow bottlenecks and obstacles were also discussed.

The intended ERP3 Project Team attendees were the four team members for whom participant K-maps had been created (a manager, an analyst, a developer, and a tester). However, the analyst could not attend. Therefore, another analyst who had been interviewed and worked closely with the absentee analyst was asked to take his

place. Additionally, the manager was called away just before the focus group began. No substitute sat in for the manager due to the late notice. A computer science PhD student, not associated with the research, acted as a note taker (additional triangulation), and the researcher as the focus group facilitator.

To begin the focus group, the researcher briefly recapped the procedure that had been followed so far regarding the ERP3 Project pilot study (e.g. interviews, individual K-maps). She felt this necessary, as there had been such a long gap since the original interviews were conducted in the fall of 2009. She also explained that the project K-map was the result of merging data from only three of the twelve participant K-maps. (These three maps were those of the developer and tester in attendance, and the analyst who could not attend.) Again, the manager K-map was used for triangulation purposes and, therefore, not used as input to the project K-map. However, the data from the manager K-map was helpful in stimulating focus group discussion. Lastly, she explained that the focus group would be conducted as if analyzing the project K-map in terms of SPI for projects in the future that might have a similar structure to the ERP3 Project. The focus group was originally intended to be conducted in terms of improvements that could be made to the ERP3 Project as it was ongoing. However, the ERP3 Project was at this point soon coming to an end.

The researcher then asked the three participants to take a few minutes to look at the enlarged project K-map, which was taped to a whiteboard at the front of the room (Figure 10). She briefly explained its notation (e.g. knowledge sources and sinks,

explicit-to-tacit, tacit-to-explicit, and tacit-to-tacit knowledge flows) and asked that the group point out any obvious errors or inaccuracies that they might see. No problems were identified.

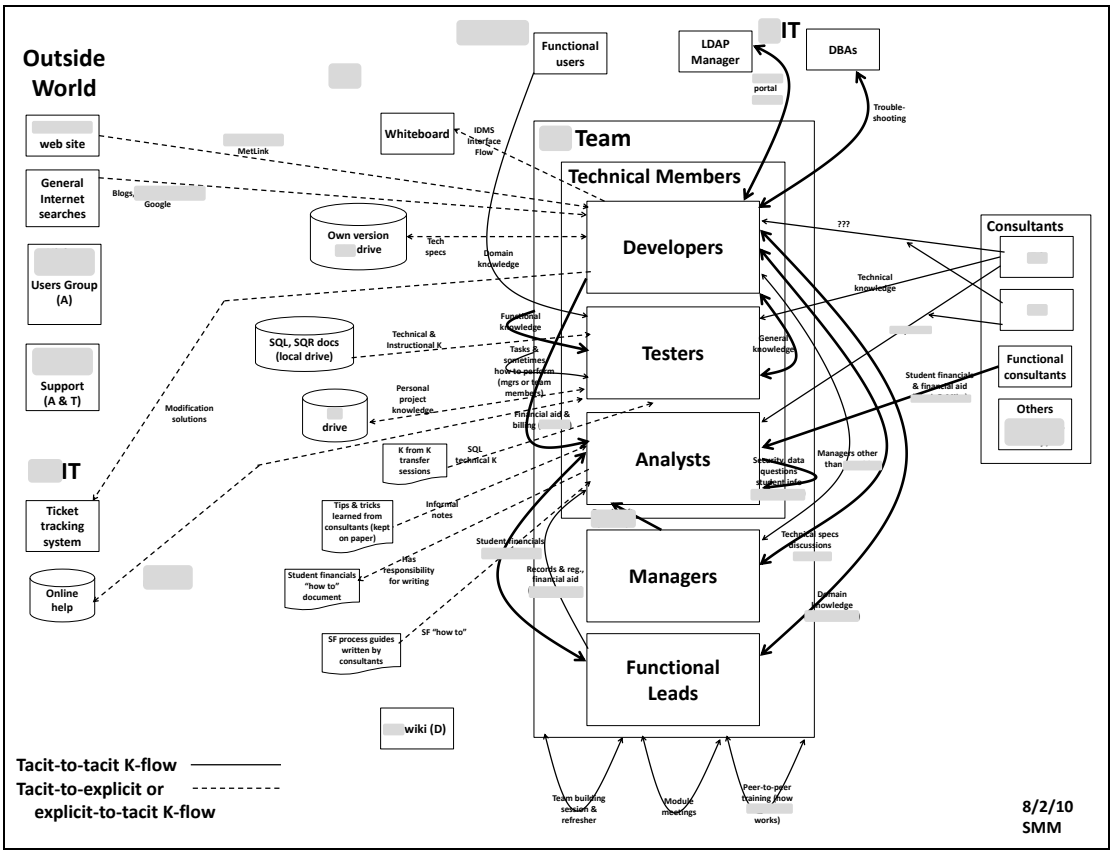


Figure 10. Project K-map at Start of Focus Group Session (Sanitized)

The process of locating K-flows that were candidates for SPI followed. Conversation was guided, yet informal and free-flowing. The researcher used colored markers to annotate the K-flows being discussed. As she added comments to the K-map, she confirmed with the group that the comments were consistent with their thoughts. The note taker was also actively capturing the group discourse, which would later be

cross-checked with the final project K-map. A photograph of the final annotated K-map is shown in Figure 11.

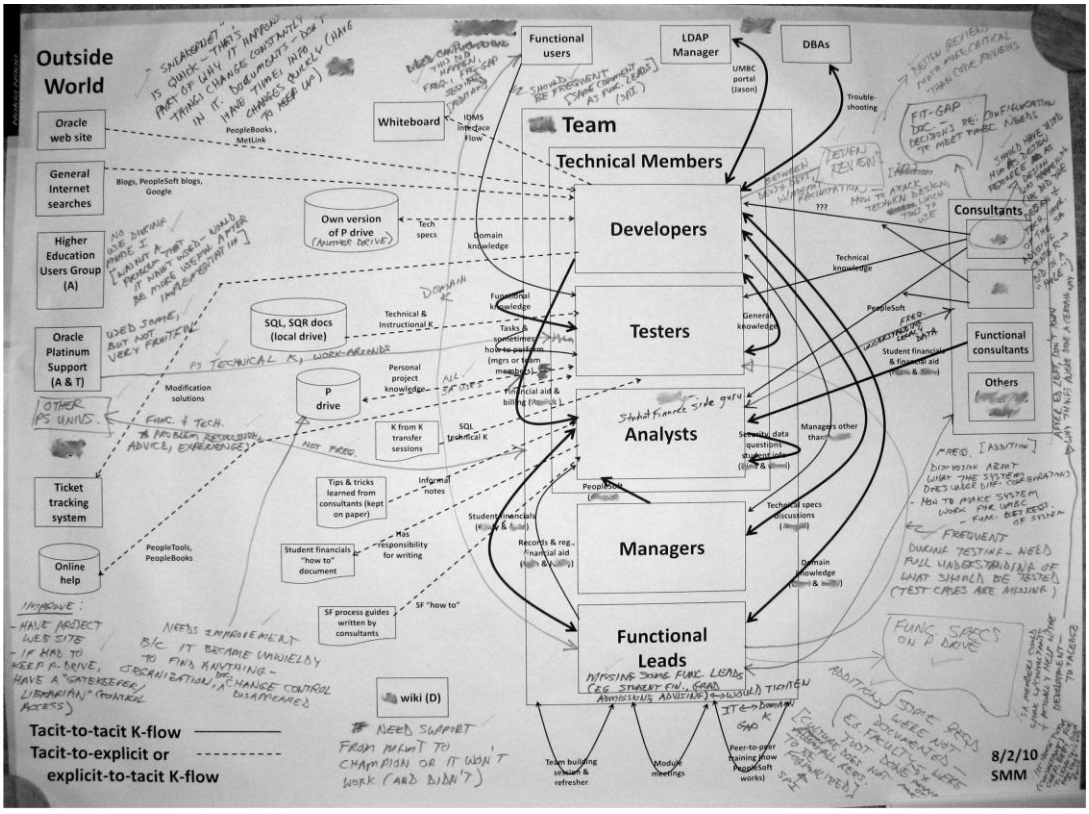


Figure 11. Project K-map at Conclusion of Focus Group Session (Sanitized)

3.2.2.7 Steps 7 and 8: Participant Questionnaires

The last stage of the study methodology was to confirm that the study procedures had actually captured K-flows that the ERP3 Project Team *as a whole* (not just the focus group attendees) considered to be compelling areas for SPI. This was accomplished through the construction of a questionnaire based on the K-flows identified by the focus group for improvement, the focus group notes, and the researcher's field notes.

After the focus group, the researcher compiled a list of the K-flows identified as candidates for SPI in the focus group, in no particular order. She then asked the focus group participants individually, by e-mail, to prioritize the K-flows from most to least important to SPI for projects that might be similarly structured to the ERP3 Project.

Based on the participants' prioritizations, the highest-priority K-flows were chosen to form the basis of a questionnaire design. Focus group notes were consulted to find specific potential solutions, in the form of K-flow modifications, to the highest-priority problematic K-flows as identified by the group. The questionnaire asked the respondent to rate how likely particular modifications (described in section 3.2.3) to the K-flows would have been to 1) improve the time that it took him/her to complete his/her work (a proxy for project cost), 2) improve his/her ability to meet deadlines (a proxy for project schedule), and 3) improve the quality of his/her work (a proxy for end-product quality). In reality, three slightly different versions of the questionnaire were developed, as there were three different types of participants: software engineers (analysts, developers, and testers), managers, and an "observer" (the DBA). The questionnaire for the DBA (referred to as the "Observer Questionnaire") was constructed by "reversing" the Software Engineer Questionnaire's questions. That is, rather than ask the questions in terms of how the DBA thought a particular K-flow change would have affected him/her, it was asked in terms of how he/she thought it would have affected the software engineers. This wording adjustment was necessary to use the Observer Questionnaire for triangulation purposes. Finally, a Manager Questionnaire was developed. With the exception of one additional question, it was

identical to the Observer Questionnaire. The additional question concerned the resources that the manager would be willing to allocate for a Knowledge Flow Facilitator (KFF) role. This question was added to further address research question six ("What evidence exists to justify the creation of a dedicated project KM role?"). Piloting of the Software Engineer Questionnaire was performed in early February of 2011 using the computer science PhD student focus group note taker. Only minor wording adjustments were needed. The appropriate questionnaires were administered online using the Survey Monkey survey software application (SurveyMonkey 2009-2011) to the nine ERP3 Project Team software engineers, the two managers, and the DBA (the "observer") on February 11, 2011.

3.2.3 Results and Discussion

The results of the pilot study are given in this section. Two observations are presented first, followed by a description and discussion of the questionnaire results.

One observation from the four participant K-map validation sessions indicates that knowledge and knowledge management are difficult concepts for study participants to become comfortable with. Recall that the validation sessions were one-on-one meetings used to confirm that the researcher had correctly captured the K-flows and associated data that a participant had identified in his/her initial interview. As she and the participant walked through his/her K-map, many K-flows, such as user help requests and deliverables lists, were identified *by the participant* as actually being information flows, and were eliminated from the map. This was true for all four

validation sessions. Thus, it took both an interview and a K-map validation session for each participant to internalize the distinction between knowledge and information and K-flows and information flows. This observation points out the necessity for the understanding by all parties concerned with the K-mapping process of a thorough, consistent working definition of knowledge. It also lends some support to the need for project-level personnel with KM expertise.

Moving to the ERP3 Project K-map focus group, the observation was made that the attendees (an analyst, a developer, and a tester) appeared to be more relaxed and forthcoming than if their manager had actually been able to attend (recall that the manager had been called away at the last minute). No negative statements regarding the manager were made. However, some good-hearted joking took place, with remarks such as, "We won't tell John about that!" (Note that "John" is a pseudonym.) Reflecting upon this, we believe that manager interviews, K-map drawing and validation, and manager responses to the final questionnaire give enough triangulation from the management perspective. As the research is concerned with K-flow improvement from the software engineers' point of view, it appears advantageous to exclude managers from the project K-map focus group.

Looking in further detail at the outcomes of the ERP3 Project K-map focus group, thirteen K-flows were identified from the project K-map as potential areas for SPI. They are listed, in no particular order, in Table 3. They are numbered only for the purpose of further discussion.

#	K-flow Description	Why the K-flow Should be Improved
1	End Users --> Testers (domain knowledge)	Testers need to know the full expectations of the system's functionality from a user perspective. K-flow should be more frequent.
2	Domain Leads --> Testers (domain knowledge)	Testers need more than test scripts. They need to be sure that no test cases are overlooked. K-flow should be frequent.
3	Modification requestor --> Developers (system modification knowledge)	Not all specifications (modifications) are formalized, but should be. (The project culture allows undocumented modifications if the person requesting them "has power.")
4	Shared drive <--> ERP3 Team (all project information and knowledge)	It is difficult to find what you need. Even if found, there is no versioning control of documents, so you don't know what stage the document is in and what has been done to it.
5	ERP3 wiki <--> ERP3 Team (project information and knowledge)	A wiki exists, but is not used because no manager champions its use. Could be used in place of the shared drive.
6	ERP support organization --> ERP3 Team (technical troubleshooting knowledge)	Not very fruitful. The organization cannot always answer questions or give quality answers.
7	ERP system documentation --> ERP3 Team (technical knowledge)	ERP system documentation is not very good.
8	ERP3 Team --> documentation in general (technical and domain knowledge)	"Things" constantly change and there is no time to document or constantly make changes to the documents.
9	Domain Leads <--> ERP3 Technical Team members (domain and technical knowledge)	Need more Domain Leads. Some domain knowledge is missing. There is a knowledge gap between technical and domain people. Both need more of the other's knowledge.
10	Developers <--> Developers (how to attack technical design and which tools to use)	Design reviews are needed. This knowledge-flow/resource is totally missing.
11	Lessons learned from the installation of the previous two pieces of the ERP system --> ERP3 team (knowledge on what went wrong/right with these efforts and why)	The Technical Manager uses this document to make sure that the team doesn't go down the same wrong paths as the previous two efforts. It seems that the rest of the ERP3 Team either doesn't have access to or know that this knowledge exists.
12	Outside consultant <--> ERP3 Team (knowledge about portions of ERP3 implementation, especially student advising)	This flow did not exist. ERP3 Team doesn't know why the consultant implemented parts of the system the way that he did.
13	Tutor/trainer --> ERP3 Team members without ERP software system experience (technical ERP software system knowledge)	Some ERP3 Team members had never worked with the ERP software before, yet they were expected to keep up their usual pace of work.

Table 3. K-flows Identified by Focus Group for Potential SPI

Each focus group participant's top five K-flows, from most to least important to SPI for projects that might be similarly structured to the ERP3 Project, are given in Table 4. The K-flows are designated by their numbers from Table 3 above.

	K-flows		
Priority	**Analyst**	**Developer**	**Tester**
1	13	4	2
2	2	5	1
3	9	3	10
4	3	1	4
5	4 and 5	9	8

Table 4. Top Five K-flows as Prioritized by Focus Group Members

As can be seen, only K-flow #4, (Shared drive to/from ERP3 Team) appeared in all three top five lists, so this flow became "1" on the overall priority list. K-flows 1, 2, 3, 4, 5, and 9 each appeared twice. To determine their overall priorities, the researcher looked at how highly rated each K-flow was in each of the group member's lists. For example, K-flow #2 (Domain Leads to Testers) was very high on two of the three lists. Therefore, she gave this K-flow an overall priority of "2." And so on. The resulting overall top five K-flow prioritization is given in Table 5. Notice that these K-flows fall into one of two categories: explicit project document storage and retrieval (priorities 1 and 5) and tacit internal team communication (priorities 2, 3, and 4). Not only did the most highly prioritized K-flows from the focus group converge into two categories, but the importance of both explicit and tacit K-flow to SPI is revealed.

Priority	K-flow for SPI
1	Shared drive <--> ERP3 Team (all project information and knowledge)
2	Domain Leads --> Testers (domain knowledge)
3	End Users --> Testers (domain knowledge)
4	Domain Leads <--> ERP3 Technical Team members (domain and technical knowledge)
5	ERP3 wiki <--> ERP3 Team (project information and knowledge)

Table 5. Overall Top Five Prioritized K-flows for SPI

As mentioned previously (section 3.2.2.7), the five top problematic K-flows in Table 5 were used to design the final questionnaires (Software Engineer, Manager, and Observer). The suggested K-flow improvements from the focus group notes were used as the basis for the questions. To keep the questionnaire to a reasonable length, only four of the five K-flows in Table 5 (K-flows 1, 2, 4, and 5) were included. K-flow 3 was chosen for exclusion as it to some degree overlaps with K-flow 2. K-flows 2 and 4 were also combined into a single set of questions, as K-flow 2 is a "sub-K-flow" of K-flow 4. Questions are in the format of a balanced five-point bipolar Likert scale, ranging from the most positive answer to the most negative. The option of "Don't know" is also included. Each question concludes with an open-ended area for any additional associated comments.

All questionnaires (Software Engineer, Manager, and Observer) were divided into the same three sections. Section 1 addresses explicit K-flow in the form of document storage and retrieval (Table 5 K-flows 1 and 5). Section 2 addresses tacit K-flow among ERP3 Project Team members (the combined Table 5 K-flows 2 and 4). And

Section 3 addresses the potential benefit of a project-level KM role, providing data with which to address research question six ("What evidence exists to justify the creation of a dedicated project KM role?"). All questionnaire questions and their corresponding results are given in figures later in sections 3.2.3.1 through 3.2.3.3.

Questionnaire replies were received from all study participants with the exception of one software engineer and one manager. A second attempt was made to obtain their responses, but was unsuccessful. Replies were received from all three focus group attendees, allowing for a reasonable comparison between their responses and those of team members who did not attend the focus group. Looking at the questionnaire results, there does not appear to be any pattern of more positive or negative responses by the focus group attendees versus the non-focus group respondents, thus reducing any concern about potential bias that the focus group attendees might have in favor of the ideas generated at the focus group.

Tables giving the questionnaire questions, along with detailed discussions of corresponding responses, are presented below. To simplify the presentation of the responses, the tables are displayed with the questions in the form presented in the Software Engineer Questionnaire. However, responses from all three questionnaire types (Software Engineer, Manager, and Observer) are shown.

3.2.3.1 Document Storage

Section 1 of the questionnaire dealt with the storage and retrieval of explicit project knowledge (i.e. documentation). ERP3 project documentation was stored on a disk drive shared by all team members. Section 1 consists of three "question sets," as shown in Figure 12. Each question set asks for the respondent's perspective on the effect of one of the proposed modifications on potential improvement in time savings, meeting deadlines, and work quality. There is also an open-ended comment area for each of the three question sets. Figure 12 shows the Section 1 question sets and corresponding results. No open-ended question set responses were received from software engineers, the manager, or the observer that add substance to the results discussed below.

Section 1: Document Storage

Key

NFG = Non-focus group
FG = Focus group
M = Manager
O = Observer
A = Analyst
D = Developer
T = Tester

	Question Set 1: If a mechanism had existed to help me to locate files on the shared drive, it would have								Question Set 2: If there had been a standardized mechanism to keep track of the different versions of the shared drive materials, it would have								Question Set 3: If an alternative storage and retrieval mechanism for project materials had been used in place of the shared drive, it would have							
	Sum	NFG	FG	A	D	T	M	O	Sum	NFG	FG	A	D	T	M	O	Sum	NFG	FG	A	D	T	M	O
saved me a large amount of time when doing my work.	1	0	1			1			0	0	0						1	0	1		1			
saved me some time when doing my work.	6	4	2	2	3	1		X	7	5	2	2	3	2	X	X	5	3	2	3	1	1		X
had no effect on the time to do my work.	1	1	0	1					1	0	1	1					1	1	0			1	X	
increased the time to do my work by some amount.	0	0	0						0	0	0						1	1	0		1			
increased the time to do my work by a large amount.	0	0	0						0	0	0						0	0	0					
Don't know	0	0	0				X		0	0	0						0	0	0					
greatly increased my ability to meet deadlines.	0	0	0						0	0	0						0	0	0					
somewhat increased my ability to meet deadlines.	4	1	3	1	1	2			5	3	2	1	2	2			5	2	3	2	2	1		
had no effect on my ability to meet deadlines.	4	4	0	2	2		X	X	3	2	1	2	1			X	3	3	0	1	1	1	X	X
somewhat decreased my ability to meet deadlines.	0	0	0						0	0	0						0	0	0					
greatly decreased my ability to meet deadlines.	0	0	0						0	0	0						0	0	0					
Don't know	0	0	0						0	0	0				X		0	0	0					
greatly increased the quality of my work.	0	0	0						0	0	0						0	0	0					
somewhat increased the quality of my work.	1	0	1		1				6	4	2	1	3	2			2	1	1	1	1			
had no effect on the quality of my work.	7	5	2	3	2	2		X	2	1	1	2				X	6	4	2	2	2	2	X	X
somewhat decreased the quality of my work.	0	0	0						0	0	0						0	0	0					
greatly decreased the quality of my work.	0	0	0						0	0	0						0	0	0					
Don't know	0	0	0				X		0	0	0				X		0	0	0					

Figure 12. Section 1, Questions Concerning Document Storage

Question Set 1- Help Locating Files. The question in this set asks respondents to speculate on the effect that a mechanism to help them to locate files on the shared drive would have on the time it takes them to do their work, their ability to meet deadlines, and the quality of their work. The majority of the eight software engineers responded in the positive range for time savings and meeting deadlines, with time savings receiving the highest ratings. The observer also responded in the positive range for time savings, lending some support to the software engineers' consensus regarding the same. The manager responded "Don't know" to time savings and work quality, and neutrally to meeting deadlines. Recall that the manager, as well as the observer, was responding from an "outside-in" perspective. That is, his/her responses concerned what he/she perceived about the effect of a file location help mechanism on the software engineers, not on themselves. The manager's responses may indicate that consulting only managers about project K-flow may not always be informative, and that the software engineers "in the trenches" may have a clearer perspective.

Question Set 2 – Shared Drive Standardized Versioning Control. The question in this set asks respondents to speculate on the effect of a standardized versioning control mechanism. The responses for all three categories (time, deadlines, and work quality) were strongly in the positive range. The manager's and observer's responses were also positive for time savings. The manager once again answered "Don't know" for two of the categories (deadlines and work quality), further indicating a possible disconnect between observing and being "in the trenches."

Question Set 3 – Alternative Storage and Retrieval Mechanism. This question asks how having an alternative mechanism to the shared drive would affect software engineers' performance of their jobs. Software engineers' responses were in the positive range for time and deadlines, but not so for quality. This corresponds to the responses received for Question Set 1. On reflection, questions 1 and 3 are indeed slanted towards document storage and retrieval speed, which may not necessarily be perceived as a factor in quality of work.

3.2.3.2 Communication

Section 2 of the questionnaire also consists of three question sets. The first two are similar in format to the question sets of Section 1, but deal with intra-team communication K-flow modifications, rather than document storage and retrieval K-flow modifications. The third question set was based on an existing K-flow (team member co-location) that, according to the software engineer interviews and the researcher's observations, worked well. It was included as a partial test of questionnaire validity. Figure 13 shows the Section 2 question sets and corresponding results. No open-ended question set responses were received from software engineers, the manager, or the observer that add substance to the results discussed below.

Section 2: Communication

Key

NFG = Non-focus group
FG = Focus group
M = Manager
O = Observer
A = Analyst
D = Developer
T = Tester

	Question Set 1: If there had been more frequent communication between the Domain Team leads and the Technical Team members, it would have								Question Set 2: If more Domain Team leads had been included on the ERP3 Team, it would have								Question Set 3: Having the majority of ERP3 Team members co-located on the same floor of the same building							
	Sum	NFG	FG	A	D	T	M	O	Sum	NFG	FG	A	D	T	M	O	Sum	NFG	FG	A	D	T	M	O
saved me a large amount of time when doing my work.	2	0	2	1	1				0	0	0	0					6	4	2	2	3	1	X	
saved me some time when doing my work.	6	5	1	2	2	2	X		6	5	1	3	2	1	X	X	2	1	1	1		1		X
had no effect on the time to do my work.	0	0	0					X	2	0	2		1	1			0	0	0					
increased the time to do my work by some amount.	0	0	0						0	0	0						0	0	0					
increased the time to do my work by a large amount.	0	0	0						0	0	0						0	0	0					
Don't know	0	0	0						0	0	0						0	0	0					
greatly increased my ability to meet deadlines.	0	0	0						2	1	1	2					5	3	2	2	3		X	
somewhat increased my ability to meet deadlines.	6	4	2	2	2	2	X		4	4	0	1	2	1	X	X	2	2	0	1		1		X
had no effect on my ability to meet deadlines.	2	1	1	1	1			X	2	0	2		1	1			1	0	1			1		
somewhat decreased my ability to meet deadlines.	0	0	0						0	0	0						0	0	0					
greatly decreased my ability to meet deadlines.	0	0	0						0	0	0						0	0	0					
Don't know	0	0	0						0	0	0						0	0	0					
greatly increased the quality of my work.	2	1	1	2					0	0	0						3	2	1	1	2		X	
somewhat increased the quality of my work.	5	3	2	1	2	2	X		6	5	1	3	2	1		X	5	3	2	2	1	2		X
had no effect on the quality of my work.	1	1	0		1			X	2	0	2		1	1	X		0	0	0					
somewhat decreased the quality of my work.	0	0	0						0	0	0						0	0	0					
greatly decreased the quality of my work.	0	0	0						0	0	0						0	0	0					
Don't know	0	0	0						0	0	0						0	0	0					

Figure 13. Section 2, Questions Concerning Communication

Question Set 1 – More Frequent Communication. Question set 1 asks respondents to assess the effect on their time, deadlines, and work quality if there were more frequent communication between Domain Team Leads and Technical Team members. The software engineers' responses for all three categories were almost exclusively positive, especially for the time aspect. Interestingly, the manager responded positively for all three categories (in contrast to the responses for Section 1), possibly indicating that managers may have more insight into tacit K-flow than explicit K-flow. The observer's responses to all categories were neutral.

Question Set 2 – More Domain Team Leads. Question Set 2 asks another question regarding intra-team communication, that being the effect on time, deadlines, and work quality if more Domain Leads had been part of the ERP3 team. Again, software

engineers' responses were quite positive for all three categories. Additionally, both manager and observer responses were also mainly positive. Looking at the responses to Question Sets 1 and 2 combined, it seems plausible that communication between Domain Leads and the Technical Team members may have improved if more Domain Leads had been included on the ERP3 team. Additionally, the positive responses to both question sets points out the importance of tacit K-flows, which commonly place second in the literature to explicit K-flows when it comes to SPI efforts.

Question Set 3 – ERP3 Team Member Co-location. As stated earlier, this question set was included as a partial test of questionnaire validity. The co-location of team members was expected to receive high ratings across the board, as it did, as co-location had already been identified by many software engineers in their interviews as being a very positive aspect of intra-team K-flow. Although this question set was used as a validity check, it illuminates the point that there could be utility in the location of K-flows that are currently working well for the team, "positive" K-flows. These flows could be exploited or built upon for SPI purposes during the current project or future ones.

3.2.3.3 Knowledge Flow Facilitator Role

The two question sets in questionnaire Section 3 are concerned with the potential benefit of the inclusion of a project-level Knowledge Flow Facilitator (KFF) role. Figures 14 and 15 show the question sets and corresponding results. No open-ended question set responses were received from software engineers, the manager, or the observer that add substance to the results discussed below.

Section 3: Knowledge Flow Facilitator Role

Key

NFG = Non-focus group
FG = Focus group
M = Manager
O = Observer
A = Analyst
D = Developer
T = Tester

Question Set 1

Having a Knowledge Flow Facilitator as a member of the ERP3 Team during Phase I would have

	Sum	NFG	FG	A	D	T	M	O
saved me a large amount of time when doing my work.	2	1	1	1	1			
saved me some time when doing my work.	6	4	2	2	2	2		X
had no effect on the time to do my work.	0	0	0				X	
increased the time to do my work by some amount.	0	0	0					
increased the time to do my work by a large amount.	0	0	0					
Don't know	0	0	0					
greatly increased my ability to meet deadlines.	2	1	1	1	1			
somewhat increased my ability to meet deadlines.	5	3	2	2	1	2		X
had no effect on my ability to meet deadlines.	0	0	0				X	
somewhat decreased my ability to meet deadlines.	0	0	0					
greatly decreased my ability to meet deadlines.	0	0	0					
Don't know	1	1	0		1			
greatly increased the quality of my work.	2	1	1	1	1			
somewhat increased the quality of my work.	3	2	1	1	1	1		X
had no effect on the quality of my work.	2	1	1	1		1	X	
somewhat decreased the quality of my work.	0	0	0					
greatly decreased the quality of my work.	0	0	0					
Don't know	1	1	0		1			

Figure 14. Section 3, Questions Concerning an ERP3 Team KFF Role

Section 3: Knowledge Flow Facilitator Role (continued)

Question Set 2

Key

NFG = Non-focus group
FG = Focus group
M = Manager
O = Observer
A = Analyst
D = Developer
T = Tester

Having a Knowledge Flow Facilitator as a member of *software development teams in general* would

	Sum	NFG	FG	A	D	T	M	O
save team members a large amount of time when doing their work.	1	0	1		1			
save team members some time when doing their work.	5	3	2	3	1	1		X
have no effect on the time for team members to do their work.	0	0	0				X	
increase the time for team members to do their work by some amount.	0	0	0					
increase the time to for team members to do their work by a large amount.	0	0	0					
Don't know	2	2	0		1	1		
greatly increase team members' ability to meet deadlines.	1	0	1		1			
somewhat increase team members' ability to meet deadlines.	4	2	2	2	1	1	X	X
have no effect on team members' ability to meet deadlines.	1	1	0	1				
somewhat decrease team members' ability to meet deadlines.	0	0	0					
greatly decrease team members' ability to meet deadlines.	0	0	0					
Don't know	2	2	0		1	1		
greatly increase the quality of team members' work.	1	0	1		1			
somewhat increase the quality of team members' work.	4	3	1	2	1	1	X	X
have no effect on the quality of team members' work.	1	0	1		1			
somewhat decrease the quality of team members' work.	0	0	0					
greatly decrease the quality of team members' work.	0	0	0					
Don't know	2	2	0	1		1		

Figure 15. Section 3, Questions Concerning a Team KFF Role in General

Question Set 1 – KFF as a Member of the ERP3 Team. This question asks respondents to speculate as to the effect of having had a person in a KFF role during the ERP3 project would have had on the time that it takes them to do their work, their ability to meet deadlines, and the quality of their work. There was positive support from all software engineers regarding time savings and schedule adherence, but less so for increasing the quality of their work. The manager responded that there would have been no effect in all three cases, while the observer responded in the positive for all three. This may lend some support to a possible detachment by management as to the knowledge needs of their team members.

Question Set 2 – KFF as a Member of Software Teams in General. The question in this set asks respondents to speculate on the effect of a KFF role on software development teams in general (i.e. not the ERP3 team). Software engineers were, in general, less certain as to the effect when this question was generalized, as demonstrated by the appearance of some "Don't know" responses. The observer, however, still perceived a positive effect of such a role. It is interesting that the manager expressed a positive effect of a KFF role for schedule adherence and work quality when it came to other software projects, but not for the ERP3 project, which he managed.

Recall that the Manager Questionnaire contained one additional question not included on either the Software Engineer or Observer Questionnaires. This question was, "How much overhead would you be willing to allocate towards a Knowledge Flow Facilitator for a project that you manage?" The purpose of the question was to further probe a manager's perception as to the importance of a KFF role when resources were at stake. Possible responses were as follows.

- None
- Less than one-half a person
- One-half a person
- One person
- More than one person
- Don't know

The manager responded, “Less than one-half a person.” This answer is consistent with his/her reply to Question Set 1, where he/she felt that a KFF role on the ERP3 project would have had no effect on time, schedule adherence, or work quality.

There is one last especially intriguing outcome regarding the overall questionnaire results. That is that, on the whole, the results were principally in the positive range. This indicates that 1) two K-flow areas (the shared drive and intra-team communication) perceived as problematic by the ERP3 team *in general* were located, and 2) sensible solutions as perceived by the team *in general* were produced. This is intriguing because only three of nine possible software engineer K-maps were used to create the ERP3 project K-map. We had been concerned that the inability to create and integrate all nine software engineer K-maps would result in a lack of richness of the project map contents to the point where the focus group outcomes would not generalize to the entire ERP3 software engineer population. However, this does not appear to be the case. There are a few possibilities for this not necessarily anticipated result. One is that it was just luck that sufficiently problematic K-flows, as agreed upon by the entire group of engineers, appeared on the project K-map. Another is that, although problematic K-flows were located, flows that were perceived as more problematic may have been present on a richer project map. Last, perhaps in reality it does not take input from a large number of team members to generate a project map that is sufficiently representative of the K-flow experiences, both good and bad, of the entire project team. (Lutters, Ackerman et al. 2000) Perhaps there is a plateau, or

saturation point, where the contents of individual team member K-maps overlap to the point that minimal new project map data is gained.

3.2.4 Limitations

As this was a pilot study, it is accepted that there were limitations that may threaten the validity of the results. Part of the purpose of a pilot study is to test and hone the proposed design methodology. Indeed, refinements were made to the methodology the author presented at her dissertation proposal defense as the pilot study proceeded. They were recorded and were incorporated in the subsequent industrial case study. They are also discussed later in detail in section 3.2.6, Lessons Learned.

The most significant limitation to the pilot study was likely the time gap that occurred between the participant interviews and the project K-map focus group. Steps were taken to renew focus group members' understanding of the purpose of the research and of their interview sessions. For example, two of the three focus group members had recently participated in the validation of their personal K-maps in one-on-one sessions. The researcher also provided a summary of the study purpose and the procedures that had been followed up to that point at the start of the focus group session.

Another limitation was the reduction in the richness of the project K-map used in the focus group session. As discussed, only three of the nine candidate participant K-maps were incorporated into the project K-map due to unforeseen circumstances not

associated with the pilot study. The incorporation of more or all of the individual K-maps may have resulted in a project K-map with more K-flows cited for improvement. However, questionnaire results demonstrate that ERP3 Project Team members whose maps were not incorporated into the project K-map showed no pattern in their responses to distinguish them from those whose maps were.

The pilot study relied on data gathering by a single researcher, opening the possibility for bias. We countered this by including the perspectives of non-software engineers (two managers) and an observer (the DBA). We also employed a PhD student who was not part of the research to take copious notes during the focus group session, providing an unbiased source for the recording of the session proceedings and cross-checking with the final project K-map.

Lastly, the questionnaires from which we drew our conclusions measured respondent perception, which is not necessarily the same as reality. Triangulation among the three questionnaire types (Software Engineer, Manager, and Observer) assisted to some degree with this common dilemma.

3.2.5 Summary and Conclusions

This section summarizes the pilot study findings and explains how the findings answer each of the dissertation research questions. To review, the research questions are as follows.

I. Current State of Project-level KM

1) What knowledge resources are available to software project engineers?
2) What are the knowledge flows within a software project?
3) What is the context within which knowledge flow occurs?
4) What problems are associated with project knowledge flow?

II. SPI Benefits

5) In what ways could project-level knowledge flow analysis benefit SPI?

III. Project KM Oversight and Facilitation

6) What evidence exists to justify the creation of a dedicated project KM role?

Concerning the first research question, ERP3 Project software engineers (analysts, developers, and testers) conveyed the knowledge resources that they use during their interviews and one-on-one K-map validation sessions. ERP3 software engineers relied heavily on both explicit and tacit knowledge resources. This is supported by the finding that the most highly prioritized K-flows identified for improvement by the software engineers themselves fell into two categories. The first category, dealing with document storage and retrieval, reveals a strong dependence on an explicit knowledge resource (the shared disk drive). The second category, concerning intra-team communication, demonstrates the importance of tacit knowledge resources (Domain leads, Technical Team members, and all team members in general). Other instances of explicit resources used by the software engineers are shown in the ERP3 Project K-

map in Figure 10. Examples include websites (e.g. other universities' sites, the ERP vendor product site), guides written by the ERP3 Project consultants, and documents and blogs located by general web searches. Tacit resources outside of the team include the ERP3 end users, university information technology (IT) department personnel, project consultants, and ERP product support personnel.

Consistent with the use of numerous explicit and tacit knowledge resources is the existence of K-flows that transmit the resources' knowledge (second research question). As illustrated in Figure 10, many of the K-flows were between intra-team sources and sinks. However, there is plentiful reliance on K-flows between the software engineers and management, Domain Team leads, and various sources in the "outside world" (e.g. the university's IT personnel).

K-mapping revealed that the major catalysts (or context) for K-flow both within the team and between the software engineers and the outside world was the need for domain or technical knowledge (third research question). Domain knowledge was concerned with the business processes associated with university student recordkeeping and services. Technical knowledge was generally related to the understanding and implementation of the ERP vendor product code.

Concerning problems associated with project K-flow (fourth research question), the focus group identified thirteen problematic ERP3 Project K-flows. The top five were selected and prioritized, with suggestions given for K-flow modifications in order to

support SPI. These top five problematic K-flows fell into two categories. The first category concerns difficulties with the storage and retrieval of documents to and from the project's shared disk drive, and the second intra-team communication. Additionally, the first category reveals explicit K-flow problems, while the second reveals tacit K-flow problems.

K-flow analysis culminated in the collection of questionnaire data concerning the perceptions of the software engineers regarding suggested modifications to the above mentioned K-flow problem areas. Data was gathered regarding perceptions in time savings (a proxy for cost savings), meeting deadlines (a proxy for schedule adherence), and personal work quality (a proxy for end-product quality). Overall, the responses to the Software Engineer questionnaire revealed that the software engineers perceive

- time savings to be the most likely SPI resulting from any of the three suggested modifications to the shared drive K-flow difficulty,
- the implementation of a standardized versioning mechanism the most likely document storage K-flow modification to result in SPI in all areas (time savings, meeting deadlines, and work quality), and
- time savings, improvement in meeting deadlines, and an increase in quality of work are all likely SPIs resulting from the two suggested intra-team communication K-flow modifications.

We believe that these three findings from the people "in the trenches" provide relatively firm positions from which the ERP3 Project process could have been adjusted, or future projects with the same structure can be adjusted, in order to achieve some degree of overall SPI (fifth research question). The generation of these three findings also lends support for the efficacy of knowledge mapping, in the context of this study design, for identifying viable SPI opportunities.

Last, the questionnaire also provided insight into the perceptions of software engineers regarding the effectiveness of a project-level KM facilitator role (sixth research question). The software engineers show strong support for a facilitator role on the ERP3 Project and software projects in general. It is not clear that managers necessarily have the same view or could be convinced to support such a role. The questionnaire results do, however, substantiate the further investigation into this potential solution to the effective implementation of KM at the project-level.

3.2.6 Lessons Learned

The main purpose of the pilot study was to assess and improve, if necessary, the proposed dissertation design methodology. The pilot study was quite successful in doing so. This section discusses the solutions to the researcher's open methodological questions (section 3.2), unanticipated situations encountered during the execution of the pilot study, the lessons learned from them, and how the industrial case study methodology addressed each. They will be presented in the order in which they were encountered during the pilot study.

Initial verbal contact with the pilot study participants was via telephone. The contact was generally to obtain participant background information and set a date and time for our first interview. The researcher divided the eleven participants into two groups, five who were given only a brief description of the dissertation research's purpose and six who were in addition given specific explanations and examples of knowledge and KM. The purpose was to determine if participants would be better prepared to answer questions regarding the likely novel topics of knowledge and KM during their interviews if they had been given some prior background. No difference in question comprehension or ease of reply between the two groups was observed during the interview sessions. The telephone explanations of knowledge and KM were, therefore, omitted from the industrial case study design.

Conducting the pilot study interviews provided the researcher the opportunity to assess her interviewing skills. Although she used a set of guideline questions during participant interviews (Figures 6 and 7), she found that she was not always able to ask all of the questions of all of the participants. In her efforts to allow spontaneity on the participants' part, she found that she had to skip some questions to keep within the allotted interview timeframe. The pilot study interviews gave the researcher the experience necessary to eliminate this problem in preparation for the industrial case study interviews.

One further improvement to the participant interview sessions, as a result of the pilot study experience, was the clarification of the Knowledge Flow Facilitator (KFF) role. One of the interview guideline questions (Figures 6 and 7) asked the interviewee for his/her thoughts on the inclusion of a project-level KFF role. Upon review of the recorded responses, we found that interviewee replies were mainly centered on further discussion of ERP3 K-flow problems, rather than the utility of a KFF role. As a result, pilot study interview data on this topic did not help to address the corresponding research question (question six). A clearly communicated and consistent verbal definition of the KFF concept was thus developed and used in the industrial case study to address this difficulty.

Participant K-map validation sessions were found to be absolutely essential and thus were added to the design for the industrial case study. Validation 1) facilitated the removal of resources and flows that were actually associated with information, rather than knowledge, 2) filled in missing knowledge sources, sinks, K-flows, or context information, 3) made the meaning of knowledge clearer to the participant, and 4) helped the researcher to further understand the project, the participant's role on the project, and the K-flows as perceived by the participant.

As noted in section 3.2.3, Results and Discussion, focus group participants were much more at ease without managers present. As manager interviews, manager K-map drawing and validation, and manager questionnaire responses provided ample

triangulation from the management perspective, only software engineers were invited to attend the case study focus group session in the industrial case study.

For the questionnaire creation process, the researcher relied on the focus group notes to gather suggested K-flow modifications for the problematic K-flows that had been identified. We now realize that it was fortunate that the notes did indeed contain suggested modifications. For the industrial case study, the researcher specifically asked for suggested modifications during the focus group and recorded them immediately on the whiteboard.

Although both the pilot and the industrial case studies are exploratory rather than explanatory or confirmatory, there were areas in the pilot study where short follow-up interviews with some of the participants would have provided more insight into the questionnaire responses. For example, the manager's perception of having a Knowledge Flow Facilitator (KFF) on the ERP3 Project was that it would not have had any effect on efficiency, schedule adherence, or quality of work. Yet he/she sensed that such a role would have a positive effect on schedule adherence and quality for software teams in general (i.e. other than the ERP3 team). A short session with the manager may have uncovered the reason for this distinction. Consequently, questionnaire follow-up interviews were added as the last data gathering mechanism in the industrial case study.

3.3 Industrial Case Study

3.3.1 Study Setting and Background

The case study took place within a division of a sizeable U.S. Department of Defense contracting company (hereafter referred to as the "development company"). The division has several locations with the largest, where this research was conducted, employing approximately 7,500 people. The subject project involves the development of embedded software for an aircraft radar sensor system. The project will hereafter be referred to as the Radar Project (not the actual project designation). The Radar Project effort was subcontracted to the development company by another major U.S. Department of Defense contractor, which produces the actual aircraft that utilizes the radar sensor. The Radar Project has been ongoing since approximately 2003. At its peak development effort, the project consisted of approximately 70 full-time-equivalent members. It is a U.S. Government-classified project, located in a security-controlled facility.

The particular division of the development company that houses the Radar Project operates under a matrix management structure. (Sy and Cote 2004) That is, employees have a functional manager who acts as a traditional "boss," overseeing the employee's administrative concerns. However, the functionally-grouped employees may have one or more project managers (one who could possibly be their functional manager) who oversee their work on the projects to which they are assigned. For example, in a construction company, all electricians may be grouped under the same functional manager. Yet individual electricians may be assigned to work on projects

supervised by those projects' managers. The effect of a matrix management structure relevant to the Radar Project is that the work status in terms of person-hours of the project's individual technical personnel is in a constant state of change dependent upon the current project needs. For example, earlier during the time of heavy development, the majority of the 70 full-time equivalent workforce was working heavily on the project, full-time or to a lesser extent. In the project's current state, where only maintenance activities are taking place and at a very slow pace, there are only approximately thirteen full-time equivalent technical workers on the project, with just a couple working full-time. Matrix management allows the Radar Project to efficiently manage the time and cost of its people resources. However, when specific expertise is needed and no one with the knowledge is currently assigned to the project, a person with the knowledge must be located and the permission granted by management for him or her to charge time to the Radar Project.

Basic demographics of the eleven participants are as follows. Two of the participants are in strictly management positions, one is in a combination management/analyst position, two are analysts (i.e. they are domain and technical specialists and act frequently as internal consultants to other project members), and the remaining six can be described as software people (i.e. they perform software implementation, testing, and maintenance). All participants, except for one software person, have a minimum of a bachelor's degree in a technical field (computer science, electrical engineering, or mathematics). Year of bachelor's degree conferment ranges from 1976 to 2002 and the average length of employment at the subject company is 19

years. One manager and one tester are female. (Note that, for anonymity purposes, all participants will be referred to in the masculine from this point on.)

Initial contact with the development company was made in the summer of 2009 via one of the researcher's industrial contacts. The contact's inquiry to the development company division's management as to whether or not they would be interested in participating as the subject study company was positively received. Soon after, a meeting was held at the development company for the researcher to describe her dissertation research objectives, explain the procedure that she would follow, and define the company resources that she would need. Verbal approval was given by the management present. As the researcher was still in the midst of the pilot study, the development company assigned a liaison to whom she could periodically report her progress and give a heads-up to when she became available to begin the case study.

The researcher contacted her development company liaison in April of 2011 to put into motion the paperwork and/or other procedures that would need to be dealt with to conduct the case study. Regrettably, the paperwork and unanticipated additional development company approvals, mainly due to proprietary materials and security concerns, delayed the study start. In the meantime, the development liaison chose a project for the case study (the Radar Project) and gave the researcher the project lead's contact. The researcher was able to meet with the Radar Project lead in late June 2011, and given final approval to begin participant interviews on August 1, 2011.

The Radar Project lead served as the gatekeeper for initial project background and participant recruitment. He and the researcher met for the first time on June 23, 2011 at the development company's facility to get acquainted and exchange information regarding the Radar Project and its role in the researcher's case study. The researcher explained the study's objectives, the procedure that would be followed, and the resources that she would require (number of participants, parameters for participant selection, approximate person-hours, and physical facilities). She gave the project lead a flowchart of the procedure that was followed for the pilot study (Figure 5) and explained the minor adjustments that would be made for the case study.

The project lead explained the Radar Project's goals, history, and team organization. He explained that the Radar Project had been in a "hold" state since November of 2011, as it is ahead of schedule, waiting for its customer to catch up in its testing efforts of the already-delivered portions of the radar sensor system software. During this hold state, the large majority of the project members have been dispersed throughout the division, working on other projects. They are called back to work on Radar Project maintenance problems on an as-needed basis. These may be problems reported by the customer as they test previously-delivered portions of the system, or problems discovered by the Radar Project members themselves. Thus, work is still being performed, albeit at a much slower, less-critical rate. No new development is taking place, only maintenance activities associated with previously delivered software. The project's current population consists of approximately thirteen full-

time-equivalent members. Presented with this situation, we decided that the case study should focus on Radar Project maintenance activities only, given that maintenance was the most recent effort, and, therefore, the most likely to be accurately recalled and described by participants. The Radar Project project lead suggested that the participants be team members who have been working on the project for several years, as debugging (maintenance) is not a single, isolated phase in the Radar Project life cycle, but an ongoing activity. Therefore, long-term team members would have experienced several software delivery cycles and had several chances to work on maintenance activities. On August 9, 2011, the Radar Project lead e-mailed the researcher the names and e-mail addresses of two managers, three analysts, three developers, and three testers whom he had recruited to participate in the case study.

A brief discussion regarding the assigned participants is appropriate before continuing. As with the pilot study, the researcher requested managers and a diversity of software engineers (analysts, developers, and testers) for triangulation purposes. The researcher knew from the beginning of the study that the Radar Project lead would be one of the two participating managers. However, as the initial participant interviews progressed, it became apparent that one of the analysts fell more into the manager than analyst category. He was, therefore, reclassified as a manager, resulting in three managers, two analysts, three developers, and three testers. It also became clear through the interviews that the three managers were positioned at three different levels in the Radar Project organizational chart, with the Radar Project lead at the top,

another manager (the Radar Project assistant lead) just below, and the last manager below both, directly supervising software engineers working on a specific portion of the Radar Project. As will be seen in Chapter 4, Results and Discussion, position in the personnel hierarchy lead to some interesting differences in the managers' questionnaire responses and comments.

Regarding the analysts, developers, and testers, it became clear through the initial interviews and the participant K-map validation sessions that the difference in responsibilities of the developers and testers was minimal at best. The main tasks for both groups are to implement, test, and maintain code. Upon consulting the Radar Project lead regarding this apparent lack of diversity, he responded that he had placed particular people into one or the other group based on his own observations as to the "natural inclination" of each individual towards coding or testing. Last, although the two analysts did perform some coding and testing, their major tasks leaned much more towards supplying the developers and testers with technical assistance, such as reviewing and making determinations regarding test results.

3.3.2 Data Collection and Analysis Procedure

Figure 16 illustrates the case study data collection and analysis procedure. Each of the steps in the procedure is detailed below.

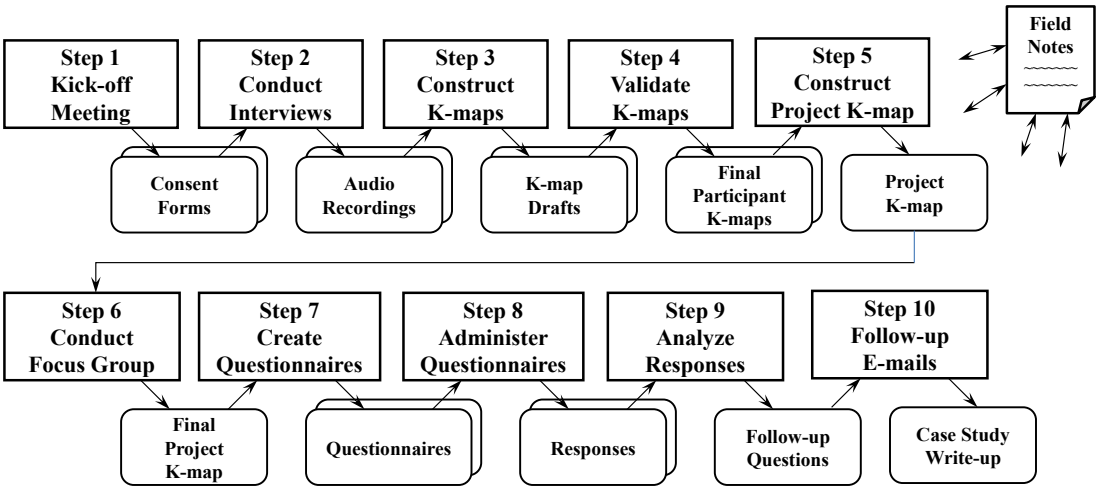


Figure 16. Case Study Data Collection and Analysis Procedure

3.3.2.1 Step 1: Kick-off Meeting

The project lead and the researcher met one more time on August 12, 2011 to discuss the plans for a kick-off meeting. This meeting also gave the researcher the chance to bring and introduce a fellow Information Systems PhD student. The student would be observing the researcher and helping her with the interviews and focus group as part of his study of qualitative research methods. This was fortunate, as it was an excellent opportunity for triangulation. The Radar Project assistant lead also attended the meeting, as the project lead had assigned him the oversight of the logistics of the research study. All reviewed the researcher's agenda for the kick-off meeting. No changes were made. Additionally, the project lead provided the researcher with a sanitized Radar Project organizational chart. Due to the classified nature of the Radar Project, no further project documentation was shared. However, unclassified overviews of the general, high-level history of the project were available to the researcher and her assistant via the Internet.

The kick-off meeting was held on August 16, 2011 in a small conference room at the development company. All participants had been mandated to attend, but three were absent. Additionally, the project lead unexpectedly left to catch a plane once she determined that the remaining three people were most likely not going to show up and that the meeting should begin. She turned the proceedings over to the assistant project lead. The following is a summary of the topics covered at the meeting.

- Introductions and thanks
- Research purpose and its importance to the empirical software engineering community
- What we are asking from the participants (e.g. interviews, focus group) and the amount of time that each task will take
- How the company and participants benefit from the research
- Anonymity and protection of research data
- Consent form explanation and signatures
- Questions and thanks

Although we did obtain consent form signatures from the seven attendees, we did not attempt to schedule initial interviews, as people had not brought their personal calendars. All interviews were scheduled at a later date by the researcher via individual e-mails to each of the eleven participants.

Study data was collected during the period from August, 2011 through January, 2012. The individual steps for data collection are described in detail below.

3.3.2.2 Step 2: Participant Interviews

Eleven semi-structured, audio-recorded participant interviews were conducted between August 17, 2011 and September 8, 2011. Two managers, three analysts, three developers, and three testers were interviewed. Both the researcher and the PhD student assistant attended all interviews, with one conducting the interview and the other taking notes by hand in the event that the data recorder malfunctioned. Although the person conducting the interview asked the questions on the interview guide, the note taker was permitted the freedom to interject when he or she felt that clarification was necessary or that an opportunity for eliciting relevant data might be missed. As previously stated, the PhD student assistant was in attendance to both observe the interviews and to practice conducting interviews himself. Therefore, interviewing and note taking were shared, with the researcher conducting seven interviews and the assistant four. Although no data recorder malfunctions occurred, the person who took notes shared them with the other.

Upon arriving for each interview, both the researcher and her assistant signed in at the guards' desk in the main lobby of the development company facility. Visitor badges were issued, and the interviewee, who would meet us (the researcher and the PhD student assistant) at the desk, escorted us to a small conference room just off the lobby. In situ interviews were not possible due to the classified nature of the facility.

However, audio recording had been approved as long as it took place in view of the guards' desk. The dimensions of the room were approximately 10 X 7 feet, with one of the seven foot sides made of glass. The occupants were, therefore, viewable by people passing through the adjacent hallway. To help the interviewee feel comfortable given this situation, the data recorder was always placed so that it was hidden from anyone who might glance in. We also never allowed an interviewee to sit facing the glass wall. There was never the sense that an interviewee was in any way disturbed or reluctant to fully participate due to the physical environment.

The following procedure was used for each of the eleven semi-structured interviews.

- We thanked the interviewee for participating in the study.
- We then provided a brief refresher on the research study purpose.
- If the participant had not attended the kick-off meeting and, therefore, not signed a consent form, we explained the contents and purpose of the form and obtained the participant's signature.
- The participant was then asked to verify that he/she felt comfortable interviewing from the perspective of the role that we had been told he/she plays on the Radar Project (i.e. analyst, developer, tester, or manager). All answered in the affirmative.
- We reminded the participant to try to answer the interview questions as they pertain to the Radar Project's maintenance activities only.

- The interview was conducted. Participant demographics (i.e. educational and professional backgrounds) were gathered as the last stage of the formal interview process.
- To wrap up, a brief reminder of the remaining study steps was given. Participants were informed that the study results would be provided to the development company either in the form of a report or the researcher's dissertation.

The interviews used the same guideline questions (Figures 6 and 7) as used in the pilot study, with one exception. The exception was the addition of the definition of Knowledge Flow Facilitator (KFF) to question 8, which is concerned with the interviewee's thoughts on the inclusion of a project-level KFF role. One of the lessons learned from the pilot study was that, without a clear, consistent definition of a KFF, interviewees tended to center their answers on further discussion of project problems, rather than focus on the utility of a KFF role. The modified question is given in Figure 17. When arriving at question 8, we would explain to the interviewee the need for us to read them the KFF definition verbatim for clarity and consistency among interviews. This modification worked very well to focus the interviewee on the topic at hand.

The researcher and the PhD student assistant exchanged thoughts on what had taken place immediately following each interview. This greatly contributed to triangulation regarding the content of the interviews. Verbal critiques regarding each other's

interviewing style were also exchanged, enabling the honing of the interviews in their ability to keep the participants engaged and focused.

One additional interview was conducted on February 14, 2012. The person interviewed was a former Radar Project software engineer. The researcher was compelled to interview this individual as he and his contributions to the Radar Project in the area of knowledge facilitation were remarked upon in a favorable light in two participant interviews and again in the focus group. The researcher wished to probe the individual's viewpoint of this perception of him by his former coworkers. A welcome side benefit was that his perspective enriched the pool of triangulation data.

8) I've been thinking about whether having a designated "knowledge facilitator" role would help a project team to be more efficient. I was wondering about your thoughts on this idea.

<u>Knowledge Facilitator</u>: A role to proactively oversee the knowledge needs of the development team and to get the right knowledge to the right team members in a timely manner. He/she will also respond to requests for help in attaining knowledge. Characteristics of the role:

- **Is a role, like a tester, programmer, quality assurance, not necessarily a single person**
- **Team member, not management**
- **But can act as a connection between the team and management**
- **Would be an advocate for the team, not management**
- **Would be technical to a good degree**
- **Would set up the initial infrastructure for team knowledge flow (e.g. document storage, wikis, physical work environment)**
- **Would be proactive, every day, in making sure that obstacles to the team's progress are identified and addressed (e.g. putting the right people together, asking people if they have knowledge needs, advocating training when needed)**
- **Would not get in the way of the team**

Do you have any thoughts on such a role

For this project?
For software projects in general?

Figure 17. Modified Interview Guideline Question 8

3.3.2.3 Step 3: Audio Transcription and Participant K-map Construction

Eleven participant K-maps were constructed from the audio recordings in the same manner as in the pilot study. That is, for each recording, the researcher listened to the audio while simultaneously drawing and annotating each relevant K-flow discussed. Participant K-map construction took place from mid-October through November of 2011.

3.3.2.4 Step 4: Participant K-map Validation

The contents of each of the eleven participant K-maps were verified as in the pilot study. That is, the researcher and each participant went over his or her K-map's K-flows one by one, with the participant validating, clarifying, or modifying each. A sample participant K-map from the pilot study can be seen in Figure 8. K-map validation sessions took place from late October through November of 2011. The PhD student assistant observed, but did not assist with, the first three validation sessions.

At this point, one adjustment was made to one of the eleven participant's role. After drawing and validating this person's map, it was clear that he played a major part in the Radar Project's technical management. In particular, he was the main point of contact for the customer for technical issues, investigating the issues, assigning their resolution to Radar Project software engineers, and tracking their progress. It is understandable that the Radar Project lead had designated him as an analyst, given his considerable domain knowledge. However, his performance of extensive technical management tasks would almost certainly bias the software engineer perspective that we were seeking. He was, therefore, recast to the manager role category, resulting in two analysts, three developers, three testers, and three managers. Although this depleted the pool of software engineers by one, it was a necessity.

3.3.2.5 Step 5: Project K-map Construction

The researcher next consolidated the eight individual software engineer K-maps into a single Radar Project Software Engineer K-map for use in the upcoming focus group. The sanitized map is shown in Figure 18. The map is roughly divided into three vertical areas. Moving from left to right, there is the Outside World, containing knowledge sources and sinks exterior to the development company, the development company itself (Company), and the Radar Project (Program), which occupies most of the right-hand half of the map. People are represented as single stick figures, while groups of people (e.g. all software engineers) are shown as three overlapping stick figures. Knowledge sources and sinks are represented by a variety of symbols, such as disk drives, documents, and boxes. K-map commentary is denoted by a box with a clipped corner. Last, K-flows are represented by unidirectional and bidirectional arrows, indicating the direction(s) of knowledge flow. Solid arrows represent explicit-to-tacit, tacit-to-explicit, and explicit-to-explicit K-flows, while dashed arrows indicate tacit-to-tacit K-flows.

The three manager maps were also consolidated into a single map (the Radar Project Manager K-map). Aside from triangulation purposes, the researcher had created the Manager K-map as a personal reference for use in the focus group. She had done so for the pilot study, using the Manager K-map as an aid for ideas for provoking conversation when there was a lull. For the case study, there was never a need to refer to the Manager K-map for this purpose. There are, however, some interesting observations regarding the Manager K-map as compared with the Radar Project K-

map. These observations are discussed and the Manager K-map presented later in section 4.1.

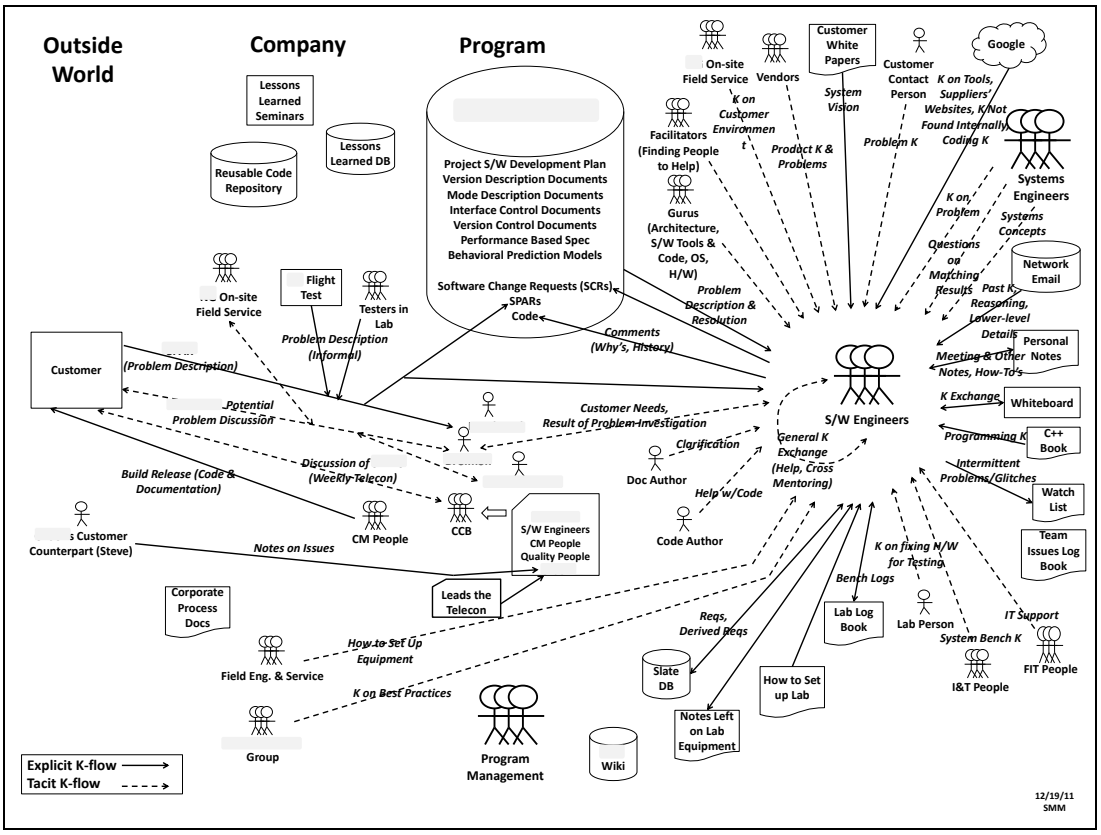


Figure 18. Radar Project Software Engineer Knowledge Map (Sanitized)

3.3.2.6 Step 6: Software Engineer Project K-map Focus Group

A focus group was conducted on December 21, 2011 to analyze the Radar Project Software Engineer K-map. It took place at the development company in the same conference room as the study kick-off meeting. The three software engineers in attendance were chosen by the researcher. There were originally two analysts, three developers, and three testers in the pool of potential focus group participants, all of whom had been interviewed. The pool was lessened due to the inability to attend the

focus group by two of the testers (one for medical reasons and the other not desiring to participate) and one of the developers being tied up on a project. However, the ability to create a diverse team of one analyst, one developer, and one tester still existed. Of course, the remaining tester was used by default. The selection of both the analyst and the developer was made by choosing the people who had most recently participated in Radar Program maintenance activities. In addition to the three software engineers, the researcher and the researcher's PhD student assistant, who had participated in the participant interviews and observed some of the participant K-map validation sessions, were in attendance. The PhD student assistant was present to take notes, help the researcher keep the group conversation going, enhance study triangulation, and, on a more practical note, to complete his assignment of observing and participating in qualitative research methods. The session was not audio or video recorded, just as the pilot focus group was not, as we wanted to maintain a non-threatening atmosphere in which participants could feel free to openly discuss their perspectives regarding Radar Project K-flow problems.

The session began with the researcher recapping the study procedures that had led to the construction of the project K-map. The 4' by 3' K-map had already been taped to the wall in front of the conference table, next to a whiteboard. On the whiteboard, we drew a table with columns labeled "#", "K-flow", "Problem", and "Solution(s)". All three participants were directly facing the map and whiteboard within easy viewing distance. The researcher then explained the K-map structure, pointing out and giving general descriptions of sources, sinks, flows, and the difference between explicit and

tacit knowledge. She also briefly guided them through the sections of the map, pointing out such groupings as flows between software engineers and 1) systems engineers, 2) project artifact (e.g. documents, code) repositories, 3) personal resources, and 4) software testing laboratory resources. The purpose of this verbal map overview was twofold. First, the participants needed to be familiar with the notation and layout of the map in order for it to be useful to them during the session. Second, the researcher desired a catalyst for the spontaneous initiation of a discussion of map K-flows. Indeed, the analyst, unprompted, began discussing the area of the map concerned with software engineers' interaction with the testing lab equipment, expressing his frustration with the process required to change the equipment from the configuration it was found in, which was not always the same, to the configuration that was needed to conduct his software tests. The lack of knowledge of how to configure the lab equipment for testing became #1 on our whiteboard table of problematic K-flows. Discussion of other K-flow obstacles and potential solutions and the filling-in of the whiteboard table continued until the participants had exhausted their thoughts. The participants were then asked to cooperatively prioritize the whiteboard K-flows from most to least likely to result in software process improvement if action were taken to alleviate the K-flow obstacle. The approximately one hour and fifty-five minute session concluded after informing the participants that the researcher might have to contact them once more for brief follow-up interviews.

3.3.2.7 Steps 7 and 8: Participant Questionnaires

The next step in the data collection process was the construction and distribution of a participant questionnaire. As with the pilot study, the purpose of the questionnaire was to confirm that the study procedures had actually captured K-flows that the participants as a whole (not just the focus group attendees) considered to be compelling areas for SPI. The questionnaire's contents were based on the prioritized K-flows identified by the focus group as problematic, the focus group's suggested improvements to these flows, and the researcher's focus group notes. As with the pilot study, the questionnaire asked the respondent to rate how likely particular alterations to the K-flows would be to 1) improve the time that it takes him/her to complete his/her work (a proxy for project cost), 2) improve his/her ability to meet deadlines (a proxy for project schedule), and 3) improve the quality of his/her work (a proxy for end-product quality). Two slightly different versions of the questionnaire were developed, as there were two types of participants: software engineers (analysts, developers, and testers) and managers. The questionnaire for managers (referred to as the "Manager Questionnaire") was constructed by "reversing" the Software Engineer Questionnaire's questions. That is, rather than ask the questions in terms of how the manager thought a particular K-flow change would affect him/her, it was asked in terms of how he/she thought it would affect the software engineers. This wording adjustment was necessary to use the Manager Questionnaire for triangulation purposes. One additional question was also added to the Manager Questionnaire. The question concerns the resources that the manager would be willing to allocate for a Knowledge Flow Facilitator role. This question was added to address research

question six ("What evidence exists to justify the creation of a dedicated project KM role?"). Piloting of the Software Engineer Questionnaire was performed in mid-January of 2012 using a former UMBC Information Systems PhD student who is experienced in the creation, distribution, and analysis of the results of research questionnaires. Only minor wording adjustments were needed. The appropriate questionnaires were administered online using the Survey Monkey survey software application (SurveyMonkey 2009-2011) to the eight Radar Project software engineers and the three managers on January 16, 2012. All responded within four weeks.

3.3.2.8 Steps 9 and 10: Follow-up Interviews

Follow-up interviews were originally planned as the last step in the case study methodology. One-on-one structured interviews were to be conducted with those participants for whom the researcher had additional questions triggered by the questionnaire results. During questionnaire response analysis, it became clear that a more timely and convenient method with which to pursue the answers to such questions would be e-mail. The researcher found that she needed questions answered as she did her analysis, in an iterative fashion, rather than all at once. This method was also a better option for the development company. The Radar Project team members were extremely busy at this point, and the scheduling and conduct of up to eleven 45 minute interviews, as originally planned, would have been both difficult and disruptive.

3.3.3 Validity Concerns

To affirm the design quality of an empirical study, of which a case study is one type, the design must address its validity in relation to four generally accepted areas: internal validity, construct validity, external validity, and reliability. (Yin 2008) The following sections discuss how this case study design addresses each of these four validity types.

3.3.3.1 Internal Validity

Internal validity applies to studies that make causal claims. That is, it defines the extent to which one can infer that one study variable has caused a change to another. It applies to all research types (e.g. experiments, case studies). Given the above definition, internal validity for case studies applies only to the explanatory case study, one that infers causality. (Yin 2008) This research study is exploratory in nature, making no causal claims, and is, therefore, not subject to internal validity scrutiny in the sense defined here.

However, many researchers hold case studies of all kinds to other standards for internal validity, as qualitative researchers can never capture an objective "truth" or "reality." (Merriam 2009) For example, strategies can be used to increase the credibility of a case study's findings. A widely-used method to do this is through "triangulation." Triangulation is the use of multiple methods in the observation of the same phenomena. (Denzin 1978) This case study utilized the internal validity check of triangulation in several ways. To begin, to provide triangulation from a study

participant perspective, software engineer participants were chosen from different, although occasionally overlapping, Radar Project roles (analyst, developer, tester). Additionally, managers were also used as participants to contrast their perceptions with those of the software engineers. Regarding the study procedures, two researchers, the dissertation author and her PhD student assistant, participated in all eleven participant interviews, the results of which provided the data for the construction of the individual participant K-maps. Both researchers also participated in the Radar Project K-map focus group, the results of which were used to develop the questions composing the Software Engineer and Manager questionnaires. Last, an additional interview with a former member of the Radar Project team was conducted (by the researcher only) to capture his perspective of the role he had played while on the team and why certain current team members considered him to be an excellent project knowledge facilitator.

3.3.3.2 Construct Validity

Construct validity is the accuracy with which the phenomena to be measured are operationalized. The overall phenomenon requiring measurement for this research is the degree to which software process improvement (SPI) transpires due to the removal of project-level K-flow obstacles. In the software engineering literature, SPI is generally considered to have three goals: reducing the time to product completion, increasing the ability to meet scheduled deadlines, and optimizing end-product quality. Therefore, if any of these three goals is met, SPI has occurred to some degree.

The unit of analysis for the study is a software development project. Members of the project team are used as the source of the data necessary to answer the six research questions enumerated in Chapter 1. Therefore, the three SPI goals above were operationalized as measurements relative to the individual team members. Specifically, the three SPI goals were operationalized as the degree to which an individual's

- time to perform his/her work would be reduced,
- ability to meet his/her scheduled deadlines would be increased, and
- work quality would increase

due to the removal of a project-level K-flow obstacle. It is reasonable to assume that individual team members' improvements in these three areas contribute to the overall improvement of the three above project SPI goals, respectively.

Some construct validity concerns do exist due to the methods used to gather the research data. Two qualitative methods, structured interviews and a focus group, and one quantitative method, a questionnaire, are used. With structured interviews, there is always the risk that interviewees are not expressing or cannot accurately express their true viewpoints due to factors such as nervousness, political reasons, and/or the lack of the ability to suitably articulate their thoughts. Our major effort to lesson these risks was to conduct a kick-off meeting with the team to introduce ourselves, explain

the purpose of the research, discuss the anonymity and protection of each individual's data, and reinforce that the study was purely academic and had nothing to do with their employment status. There is also the risk that the interviewer does not fully or partially accurately capture the individual's responses. To offset this risk, two researchers were present at every interview, providing a degree of confirmation regarding the interpretation of the interviewee responses (i.e. triangulation).

An additional effort to assure that interview responses were accurately captured was through "respondent validation." (Merriam 2009) After the researcher finished drawing a participant's K-flow map from the audio recording of his/her interview, she and the participant reviewed the K-map, covering every knowledge resource and K-flow, making modifications if necessary. This provides strong confidence in the interpretation of all participants' interview data.

Data collection using focus groups also has certain associated validity risks. The unit of analysis for this study's focus group was the K-flow obstacle. The purpose of the three-software engineer focus group was to identify K-flow obstacles and suggest modifications that they believe would have the ability to result in SPI. Some risks associated with focus groups are members who do not participate to any meaningful degree, domination of the group by one or more participants, and, as with interviews, participants not expressing or not accurately expressing their opinions. Both the researcher and the PhD student assistant actively participated in the proceedings with the intent of keeping the group interested, active, and expressive. Again, some degree

of triangulation was achieved regarding the results due to the involvement of two researchers.

The final data collection method, a questionnaire, is used to gather team members' perceptions as to the effect of suggested K-flow modifications on SPI. Therefore, the unit of analysis for the questionnaire is the K-flow modification. The modifications were drawn from those resulting from the focus group. Possibly the biggest risk to a questionnaire is the wording of the questions in order that they be correctly interpreted by respondents. In an attempt to mitigate this risk, the questionnaire was piloted on a former UMBC Information Systems Department PhD student who is well-versed and experienced in questionnaire construction. Only minor wording changes were necessary.

3.3.3.3 External Validity

External validity is the degree to which the results of a study are generalizable beyond the particular study itself. It is common for case studies to come under scrutiny when it comes to generalizability. Case studies are, in fact, not generalizable in the same manner as a study that uses statistical methods to select its population to represent a larger universe. Such studies are said to rely on "statistical generalization," whereas case studies rely on "analytical generalization." (Yin 2008) With analytical generalization, the goal is to expose results that will lend support to (generalize to) a broader theory. In the case of the subject study, that broader theory is that the identification and removal of software project-level K-flow obstacles will result in

software process improvement. However, given that no other studies of this type exist for project-level K-flow, this study is exploratory in nature and will provide the first evidence for the stated theory. Later studies may then use this study and its results to generate further theory support.

3.3.3.4 Reliability

Reliability in research generally refers to the extent to which, when replicating a research study, one arrives at the same results. The reliability of a study using qualitative methods cannot be assessed in this traditional sense. In fact, replication of a "qualitative study" will not yield the same results, given that it is human behavior that is being measured and/or observed. Complicating the traditional interpretation of reliability relative to qualitative research is that it is humans in turn who are doing the measuring and/or observing and the interpreting of the study data.

Reliability relative to qualitative methods is better measured in terms of "consistency" and "dependability." (Lincoln and Guba 1985) That is, different researchers may interpret the data collected by the same qualitative study differently. Therefore, the study's reliability is better measured by assessing the degree to which the results are consistent with the data collected. If the consistency of the results is deemed high, then the study may be considered dependable. Clearly, we consider the results of this study to be consistent with the data that we collected and, therefore, "reliable" in this sense. We have documented our transition from data to results in Chapter 4. Others may perhaps arrive at a different conclusion regarding the study's

reliability. To further substantiate the reliability of this study, we have done as much as is reasonably possible to create an "audit trail." (Lincoln and Guba 1985) That is, we have documented our study, both within this dissertation and in field notes and logs, to the degree that we believe that another researcher could follow our procedures. Again, others may perhaps judge differently.

Chapter 4: Results and Discussion

The results of the industrial case study are presented below. The chapter begins with a comparison of the Software Engineer and Manager Radar Project K-maps that were generated from the individual participant K-maps. The procedure for the selection of the Software Engineer and Manager Questionnaire questions is then explained, followed by the presentation and discussion of the questionnaire results.

4.1 Project K-maps

Figure 19 shows the Manager Radar Project K-map that was constructed from the three individual manager K-maps. Comparing the overall structure of the Software Engineer Radar Project K-map (hereafter referred to as the SE K-map) to the Manager Radar Project K-map (hereafter referred to as the Manager K-map), one can notice considerably fewer K-flows flowing between the software engineers and other knowledge resources in the Manager K-map. Recall that the interview questions from which the individual manager K-maps were generated were identical to those used for the software engineer interviews (Figure 6), but worded such that the managers were asked about project knowledge and K-flow relative to the software engineers, rather than to themselves. This discrepancy in the quantity of K-flows involving the software engineers lends support to the notion that managers are not fully aware of the quantity and type of knowledge resources that software engineers utilize. This in turn lends support to our belief that it is the software engineers, the people "in the trenches," who are best qualified to detect project-level K-flow obstacles. Other

differences in the SE and Manager K-maps, such as one map including a knowledge resource that the other doesn't, are minimal in comparison to the above.

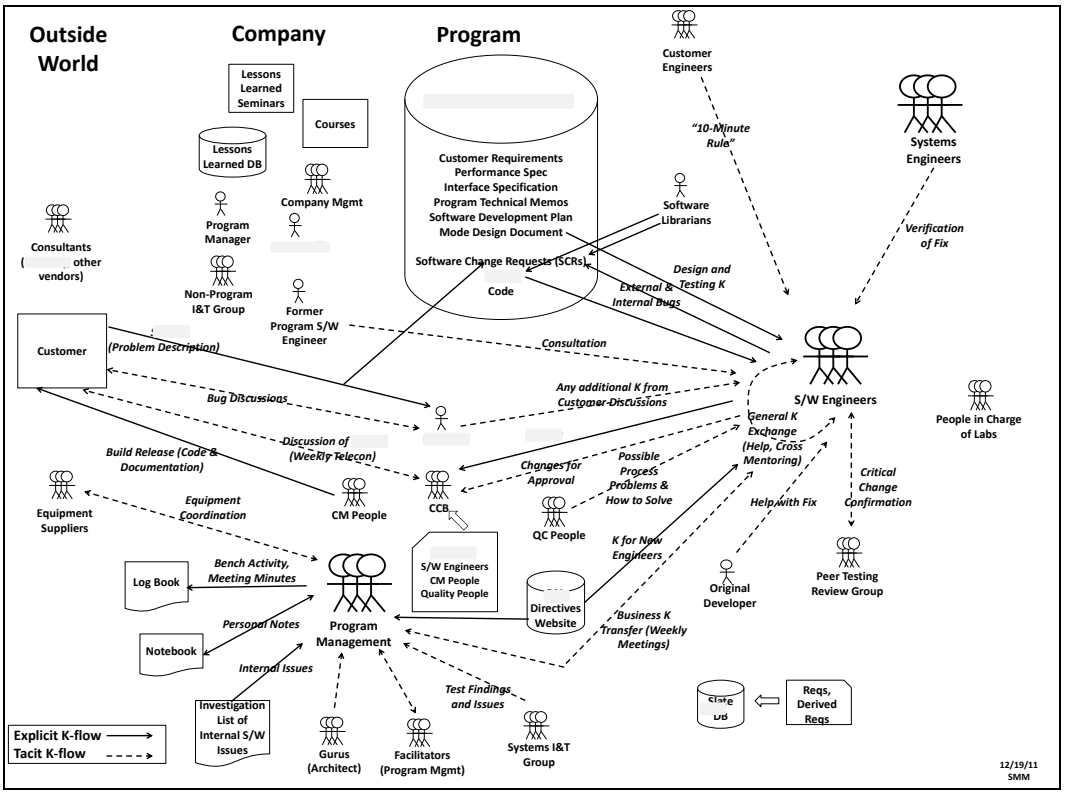


Figure 19. Manager Radar Project Knowledge Map (Sanitized)

4.2 Questionnaires

Table 6 gives the K-flows identified by the Radar Project focus group as candidates for SPI, along with the group's suggested potential improvements for each. The K-flows are organized from the most likely to result in SPI to the least, according to the consensus of the focus group. K-flows #1 and #2 by far dominated the focus group discussion. The members felt very avid about the frustrations and hardships caused them personally by these problematic K-flows and tended to expound upon them.

#	K-flow	Problem(s)	Potential Improvement(s)	Questionnaire Location
1	Program technical and administrative knowledge	Engineers' work time is wasted trying to locate documents they need Engineers' work time is wasted identifying the correct versions of the documents they need	Better document storage organization Better document versioning	Section 2, Question Set 2 Section 2, Question Set 3
2	Knowledge regarding how to configure lab equipment for testing	Engineers' work time is wasted Engineers' allocated lab time is wasted	Someone responsible for maintaining equipment configuration Constant integration and testing personnel presence Standard default equipment configuration Configuration "cookbook"	Section 1, Question Set 2 Section 1, Question Set 3 Section 1, Question Set 4 Section 1, Question Set 5
3	Bench log (testing) knowledge	Test case knowledge is not always recorded, so the log is not very useful	Replace the bench log with a person who has the equivalent knowledge	Section 1, Question Set 3
4	Non-persistent design knowledge	Impromptu design knowledge (e.g. whiteboard notes) and rationale is lost	Implement management-approved time to formalize non-persistent knowledge when it is created	Section 3, Question Set 1
5	Knowledge from engineers' personal notes and experiences	Engineers do not share personal notes and experiences that could be of value to other engineers	Implement management-approved time to formalize personal notes and experiences	Section 3, Question Set 2

Table 6. Prioritized K-flows for Potential SPI Identified by Focus Group

Notice that the K-flows listed in Table 6 fall into one of three categories: project documentation knowledge (K-flow #1), test lab knowledge (K-flows #2 and #3), and engineering knowledge (K-flows #4 and #5). Both questionnaires (Software Engineer and Manager) were, therefore, organized to reflect these three knowledge areas, with an additional section to address research question 6 concerning the creation of a

dedicated project knowledge management role. The column labeled "Questionnaire Location" in Table 6 gives the location of questions corresponding to each proposed potential solution. As with the pilot questionnaires, questions were in the format of a balanced five-point Likert scale, ranging from the most positive to the most negative answer. The option of "Don't Know" was also included. Each question concluded with an open-ended area for any additional associated comments. Formatting was identical to that of the pilot study questionnaires (previously described in section 3.2.3).

Questionnaire responses were received from all eleven study participants. Comparison between the responses from those who participated in the focus group versus those who did not does not show any pattern of more positive or negative responses from either group.

One interesting observation regarding open-ended responses was made. Each of the questionnaire's thirteen "question sets," which will be defined below, had an open-ended comment area associated with it. Seven of the eight software engineer respondents made essentially no comments (there were two short, negligible ones) in the open-ended comment areas. The one software engineer who did make comments made them in nine of the areas. This engineer was an analyst who was the most frequently mentioned "guru" during the initial interviews. All three managers made comments, with the lowest-level manager making two, the assistant project manager making eleven, and the project manager commenting in all thirteen of the open-ended

areas. It is unclear to us as to why this pattern emerged. But there appears to be a correlation between a participant's level in the Radar Project's organizational chart (the software engineer "guru" was high on the chart) and the number of comments that the respondent was willing or able to make.

4.3 Questionnaire Results and Discussion

Table 7 gives a summary of the Software Engineer Questionnaire results. Table entries are the percentages of software engineers responding that they perceive a potentially positive effect on SPI for all suggested modifications within a given K-flow modification category for time savings, meeting deadlines, and work quality. For example, the 69% in the Documentation Location/Version category represents the percent of software engineers who perceive a positive effect on time savings for the two suggested modifications from the questionnaire (questionnaire Section 2, Question Sets 2 and 3).

		K-flow Modification Category		
		Document Location/Version (2 mods)	Lab Equipment Configuration (4 mods)	Knowledge Sharing (2 mods)
SPI Category	Save time	69%	97%	88%
	Meet deadlines	50%	91%	81%
	Higher quality	25%	66%	74%

Table 7. Summary of Software Engineer Questionnaire Responses

Table 8 gives a summary of the Manager Questionnaire results. Table entries are the number of managers responding that they perceive a potentially positive effect on SPI for all suggested modification within a given K-flow modification category for time savings, meeting deadlines, and work quality. As there were only three managers, the number of managers, rather than the percent (as in Table 7) who responded in the positive range (above "no effect") for each of the modifications is given.

SPI Category	K-flow Modification Category		
	Document Location/Version (2 mods)	Lab Equipment Configuration (4 mods)	Knowledge Sharing (2 mods)
Save time	2 1	3 3 3 2	1 2
Meet deadlines	2 1	3 3 3 2	1 2
Higher quality	2 1	2 1 2 1	1 2

Table 8. Summary of Manager Questionnaire Responses

Tables giving the Software Engineer Questionnaire questions, along with detailed discussions of corresponding responses, are presented below. As described in section 3.3.2.7, the Manager Questionnaire contains the same questions as the Software Engineer Questionnaire, but with the questions "reversed," asking them in terms of how the manager thought a particular K-flow change would affect the software engineers, rather than him/her. To simplify the presentation of the responses, the tables are displayed with the questions in the form presented in the Software Engineer Questionnaire. However, responses from both questionnaire types (Software Engineer and Manager) are shown.

4.3.1 Laboratory Equipment Configuration and Use

Section 1 of the questionnaire concerns the configuration and use of the laboratory equipment utilized by the team's software engineers for software testing. It consists of five "question sets." The focus group, which consisted of three software engineers (one analyst, one developer, and one tester), clearly expressed that obtaining knowledge to correctly configure the lab equipment in order to conduct software tests was trying, time consuming, and brought with it a great deal of personal frustration. Section 1's first question set (Question Set 1) does not mention a concrete solution to the K-flow obstacle, but is used to determine if the team's software engineers as a whole, not just those in the focus group, perceive this K-flow to be an impediment to the time needed to do their work, to meet their deadlines, and/or to produce quality work. Although manager responses are included in the results, recall that these results were used for triangulation purposes only. It is the software engineers' perceptions that are the focus of this study. Each of the remaining sets (Question Sets 2 through 5) asks for the respondent's perspective on the effect of one of four different modifications to the knowledge flow (about lab equipment configuration), as proposed by the focus group, such that there is the potential for improvement in time savings, meeting deadlines, and/or work quality. There is also an open-ended area after each of the five questions sets such that the respondent may choose to elaborate on his/her answer(s). Figure 20 shows the first three Section 1 question sets and corresponding responses, and Figure 21 the last two.

Section1: Laboratory Equipment Configuration and Use

Key

NFG = Non-focus group
FG = Focus group
M = Manager
O = Observer
A = Analyst
D = Developer
T = Tester

	Question Set 1: If the amount of time it takes me to configure the lab equipment before testing were decreased, it would							Question Set 2: If the condition and configuration of the lab equipment were consistently overseen, it would							Question Set 3: If a dependable person were always available to help me with the configuration and use of the lab equipment , it would						
	Sum	NFG	FG	A	D	T	M	Sum	NFG	FG	A	D	T	M	Sum	NFG	FG	A	D	T	M
save me a large amount of time when doing my work.	3	2	1		1	2		2	2	0		1	1	2	2	0	2		1	1	1
save me some time when doing my work.	5	3	2	2	2	1	3	6	3	3	2	2	2	1	6	5	1	2	2	2	2
have no effect on the time to do my work.	0	0	0					0	0	0					0	0	0				
increase the time to do my work by some amount.	0	0	0					0	0	0					0	0	0				
increase the time to do my work by a large amount.	0	0	0					0	0	0					0	0	0				
Don't know	0	0	0					0	0	0					0	0	0				
greatly increase my ability to meet deadlines.	2	1	1		1	1		0	0	0				2	0	0	0				1
somewhat increase my ability to meet deadlines.	6	4	2	2	2	2	3	8	5	3	2	3	3	1	8	5	3	2	3	3	2
have no effect on my ability to meet deadlines.	0	0	0					0	0	0					0	0	0				
somewhat decrease my ability to meet deadlines.	0	0	0					0	0	0					0	0	0				
greatly decrease my ability to meet deadlines.	0	0	0					0	0	0					0	0	0				
Don't know	0	0	0					0	0	0					0	0	0				
greatly increase the quality of my work.	0	0	0					0	0	0				2	0	0	0				1
somewhat increase the quality of my work.	4	3	1		1	3	2	6	4	2	1	2	3	0	6	4	2	1	2	3	
have no effect on the quality of my work.	4	2	2	2	2	0	1	2	1	1	1	1		1	2	1	1	1	1	0	2
somewhat decrease the quality of my work.	0	0	0					0	0	0					0	0	0				
greatly decrease the quality of my work.	0	0	0					0	0	0					0	0	0				
Don't know	0	0	0					0	0	0					0	0	0				

Figure 20. Section 1, Laboratory Equipment Configuration and Use (Part 1)

Section1: Laboratory Equipment Configuration and Use

Key

NFG = Non-focus group
FG = Focus group
M = Manager
O = Observer
A = Analyst
D = Developer
T = Tester

	Question Set 4: If the lab equipment were consistently found in a standard (default) configuration before testing, it would							Question Set 5: If a "cookbook" of how to configure the equipment in order to run specific tests existed, it would						
	Sum	NFG	FG	A	D	T	M	Sum	NFG	FG	A	D	T	M
save me a large amount of time when doing my work.	0	0	0				2	0	0	0				
save me some time when doing my work.	8	5	3	2	3	3	1	7	4	3	2	2	3	2
have no effect on the time to do my work.	0	0	0					1	1	0		1		1
increase the time to do my work by some amount.	0	0	0					0	0	0				
increase the time to do my work by a large amount.	0	0	0					0	0	0				
Don't know	0	0	0					0	0	0				
greatly increase my ability to meet deadlines.	1	1	0		1		2	0	0	0				
somewhat increase my ability to meet deadlines.	7	4	3	2	2	3	1	5	3	2	2	1	2	2
have no effect on my ability to meet deadlines.	0	0	0					3	1	1		2	1	1
somewhat decrease my ability to meet deadlines.	0	0	0					0	1	0				
greatly decrease my ability to meet deadlines.	0	0	0					0	0	0				
Don't know	0	0	0					0	0	0				
greatly increase the quality of my work.	0	0	0				1	0	0	0				
somewhat increase the quality of my work.	5	4	1	1	2	2	1	4	3	1	1	1	2	1
have no effect on the quality of my work.	3	1	2	1	1	1	1	3	1	2	1	1	1	2
somewhat decrease the quality of my work.	0	0	0					1	1	0		1		
greatly decrease the quality of my work.	0	0	0					0	0	0				
Don't know	0	0	0					0	0	0				

Figure 21. Section 1, Laboratory Equipment Configuration and Use (Part 2)

Question Set 1 – Reduction in Lab Equipment Configuration Time. The question in this set asks respondents to speculate on the effect that a reduction in the time to configure the lab equipment for testing would have on the time it takes them to do their work, their ability to meet deadlines, and the quality of their work. Again, this particular question set was utilized to confirm that all software engineers, not just the focus group participants, perceive the lab configuration K-flow to be problematic. All software engineers (analysts, developers, and testers) responded in the affirmative range (above "no effect") for time savings and meeting deadlines, with time savings marginally receiving the highest ratings. All three managers also responded affirmatively for these two categories, demonstrating their agreement with the software engineers' opinions. When asked about the effect on work quality, four software engineer and one manager response drop to the neutral category. This decline is noteworthy, yet unremarkable, as a reduction in the time to perform one's work does not necessarily correlate with one's work quality. In summary, software engineers perceive the time that it takes to configure the lab equipment for testing to be problematic when it comes to the time it takes them to do their work and their ability to meet their deadlines. Managers appear to be aware of this problem. Although not all of the managers feel equally as strong (as demonstrated by their comments associated with this question set) about this K-flow's impact on SPI, one states

> This decrease would certainly lower frustration levels, help to optimize lab usage, and reduce integration time, all of which, contribute to

meeting overall cost and schedule constraints and improve overall morale on a project.

Question Set 2 – Consistent Oversight of Lab Equipment Condition and Configuration. The question in this set is the first of four proposed solutions to the lab equipment configuration K-flow deficiency. It asks respondents to speculate on the effect that having the condition and configuration of the lab consistently overseen by designated personnel would have on the time it takes them to do their work, their ability to meet deadlines, and the quality of their work. All software engineers responded in the positive range with respect to time savings and meeting deadlines, with time savings receiving the highest ratings. Managers also responded in the positive range for these two categories. Regarding the modification's effect on software engineers' work quality, two of the software engineer responses drop from the positive range to neutral. The manager responses follow the same pattern, with one manager response dropping to neutral. What is interesting to note here is the smaller drop to neutral regarding quality between this question set and Question Set 1. When an actual K-flow modification is suggested, software engineers are more positive about its effect on work quality than when simply asked in general about the effect of a reduction in lab equipment configuration time. One will see that this is true across three of the four potential lab configuration K-flow modifications (Question Sets 2 through 5), even for managers. This seems to imply that it is not simply the reduction in time that it takes to configure the lab equipment that affects work quality,

but more so the additional help, in the form of shared knowledge or a reduction in the knowledge needed to configure the equipment, that benefits work quality.

Question Set 3 – Human Help with Lab Equipment Configuration and Use. The question in this set asks respondents their perspectives on the effect of always having a dependable person available to help them with lab equipment configuration and use. Both software engineer and manager responses follow the same pattern as those for Question Set 2, with all responding in the positive range for time savings and meeting deadlines. There was a two-respondent drop to neutral for software engineers as well as for managers regarding the effect of the K-flow modification on work quality. At this point in the questionnaire, cost begins to show itself as an issue with management. Two managers express concern about the return on investment associated with funding a person for full-time lab help.

> The cost of keeping a full time lab manager would have to be weighed against the cost savings achieved from the engineers.

> Historically the cost of a constant presence overwhelms the benefit to the developer. Knowledgeable people are always available for telephone support or to be called in for in-person support.

As previously stated, the cost of resources and return on investment were recurring themes, mainly for managers, throughout this research investigation.

Question Set 4 – Lab Equipment Standard Configuration. This question asks respondents their perspectives on the effect of always finding the lab equipment in a standard (default) configuration before testing. Once again, both software engineer and manager responses were all in the positive range for both time savings and meeting deadlines. There was a two-respondent drop to neutral for software engineers and a one-respondent drop for managers regarding work quality. One of the analysts expresses why he believes that finding the lab equipment in a standard configuration would result in SPI regarding time savings as follows.

> One expects to spend time configuring a lab for a test that requires a unique configuration. The productivity loss (and frustration) comes from finding the lab in an unexpected or unusable configuration.

One of the managers, implicitly referring to cost, suggests the following scenario for further time savings.

> Better yet, if the responsible person for the lab configured the lab systems specifically for the work scheduled for the upcoming lab session, even more time would be saved.

Question Set 5 – "Cookbook" of Lab Equipment Configurations. This question asks respondents their perspectives on the effect of having a "cookbook" on how to

configure the lab equipment in order to run specific tests. Both software engineers and managers responded the least positively to this modification relative to those in Question Sets 2 through 4 for time savings, meeting deadlines, and work quality. It is interesting that the presence of an explicit (codified) knowledge resource is not as appealing to the software engineers as a tacit knowledge resource (i.e. a person) as proposed in Question Set 3. One of the engineers expresses his conflicting feelings regarding an explicit versus tacit knowledge resource to solve this K-flow problem as follows.

> If the "dependable person" . . . were available, the cookbook isn't needed. However, such a dependable person is not likely to be around every time one goes to use the lab.

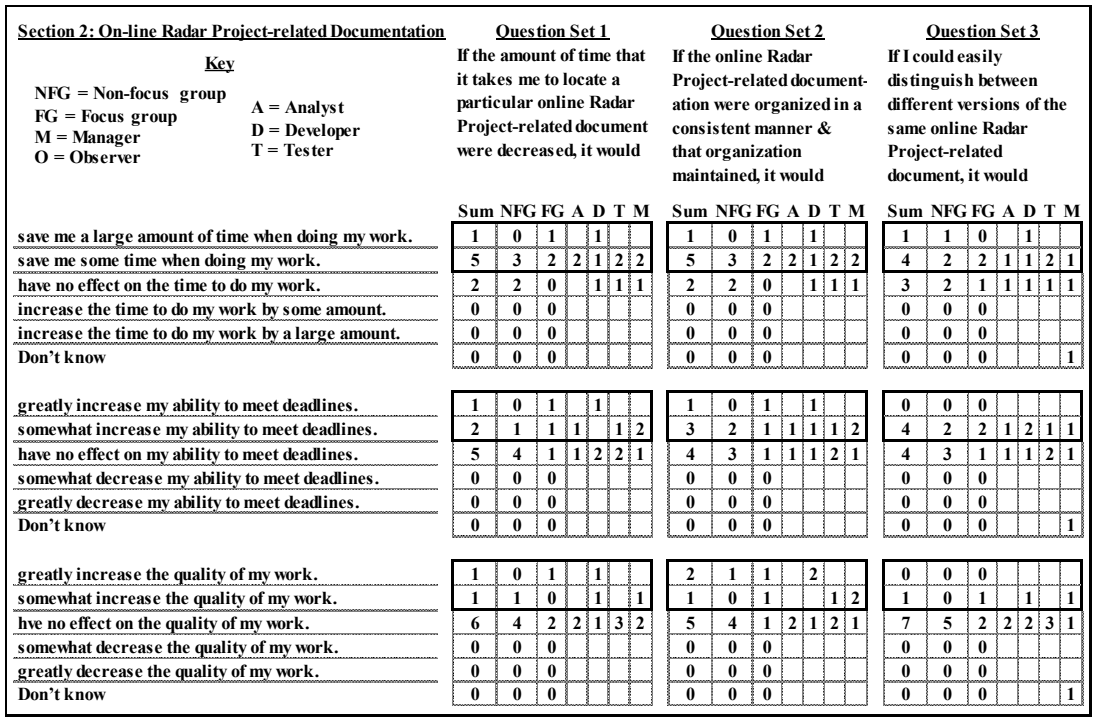
A manager points out a common problem with codified knowledge such as an equipment configuration cookbook.

> History has shown that the cookbook is good for the newbee only. That very quickly it is obsolesced or simply not referenced.

It is possible that his view is shared by other respondents and, thus, the apparent preference for a tacit resource.

4.3.2 Online Radar Project-related Documentation

Section 2 of the questionnaire concerns the perceived difficulty of retrieving explicit online Radar Project-related knowledge (i.e. documentation). This knowledge flow was rated as the most problematic by the three software engineer focus group members (see Table 6). They expressed that locating a desired document from the project's online internal document repository was both time consuming and frustrating. They also stated that, even when a document is located, it may have several versions, and that it can be difficult or even impossible to determine which is the most recent. This questionnaire section, the results of which are given in Figure 22, consists of three question sets. The first question (Question Set 1) is used to confirm that software engineers in general, not just those from the focus group, perceive the retrieval (explicit flow) of online Radar Project-related knowledge to be problematic. Like Question Set 1 in Section 1, this first Question Set does not reference a concrete solution to the K-flow obstacle. Question Sets 2 and 3 present the focus group's suggested modifications to the flow for SPI.



Section 2: On-line Radar Project-related Documentation

Key

NFG = Non-focus group
FG = Focus group
M = Manager
O = Observer
A = Analyst
D = Developer
T = Tester

	Question Set 1: If the amount of time that it takes me to locate a particular online Radar Project-related document were decreased, it would							Question Set 2: If the online Radar Project-related documentation were organized in a consistent manner & that organization maintained, it would							Question Set 3: If I could easily distinguish between different versions of the same online Radar Project-related document, it would						
	Sum	NFG	FG	A	D	T	M	Sum	NFG	FG	A	D	T	M	Sum	NFG	FG	A	D	T	M
save me a large amount of time when doing my work.	1	0	1		1			1	0	1		1			1	1	0		1		
save me some time when doing my work.	5	3	2	2	1	2	2	5	3	2	2	1	2	2	4	2	2	1	1	2	1
have no effect on the time to do my work.	2	2	0		1	1	1	2	2	0		1	1	1	3	2	1	1	1	1	1
increase the time to do my work by some amount.	0	0	0					0	0	0					0	0	0				
increase the time to do my work by a large amount.	0	0	0					0	0	0					0	0	0				
Don't know	0	0	0					0	0	0					0	0	0				1
greatly increase my ability to meet deadlines.	1	0	1		1			1	0	1		1			0	0	0				
somewhat increase my ability to meet deadlines.	2	1	1	1		1	2	3	2	1	1	1	1	2	4	2	2	1	2	1	1
have no effect on my ability to meet deadlines.	5	4	1	1	2	2	1	4	3	1	1	1	2	1	4	3	1	1	1	2	1
somewhat decrease my ability to meet deadlines.	0	0	0					0	0	0					0	0	0				
greatly decrease my ability to meet deadlines.	0	0	0					0	0	0					0	0	0				
Don't know	0	0	0					0	0	0					0	0	0				1
greatly increase the quality of my work.	1	0	1		1			2	1	1		2			0	0	0				
somewhat increase the quality of my work.	1	1	0		1		1	1	0	1			1	2	1	0	1		1		1
hve no effect on the quality of my work.	6	4	2	2	1	3	2	5	4	1	2	1	2	1	7	5	2	2	2	3	1
somewhat decrease the quality of my work.	0	0	0					0	0	0					0	0	0				
greatly decrease the quality of my work.	0	0	0					0	0	0					0	0	0				
Don't know	0	0	0					0	0	0					0	0	0				1

Figure 22. Section 2, Online Radar Project-related Documentation

Question Set 1 – Time to Locate a Particular Online Radar Project-related Document. This question was utilized to verify that software engineers perceive the location of online Radar Project-related documentation to be a problematic knowledge flow. It asks respondents to speculate on the effect that reducing the time to locate a specific document would have on the time it takes them to do their work, their ability to meet deadlines, and the quality of their work. Software engineers overall perceive online document location to be only moderately problematic regarding saving time and minimally problematic regarding meeting deadlines and work quality. This is true even in the case of two of the focus group participants, despite their ranking of document location as being the most problematic K-flow. Speculating, perhaps upon further reflection when responding to the questionnaire, they re-evaluated their assessments of this K-flow, causing their responses to fall in line with those of the

non-focus group respondents. For example, one of the software engineers from the focus group commented at the end of this question set as follows.

> The Radar Project directory structure is a mess. I have links to the documents that I use the most. Finding a document whose location I don't know can take a very long time, but I don't search for such documents very often.

So, although he acknowledges that the K-flow is a problem, he has found his own way around it.

Manager perceptions follow the same general pattern as that of the software engineers, demonstrating management's consensus with the engineers that the location of online documentation is a moderate problem.

Question Set 2 – Online Radar Project-related Documentation Organization and Maintenance. This question presents a potential solution to the explicit knowledge K-flow dilemma. It asks respondents their perceptions of the effect of having online Radar Project-related documentation organized in a consistent manner, and that organization maintained, on their time to do their work, meeting their deadlines, and the quality of their work. Software engineer responses were moderate with respect to improvements in work time and meeting deadlines. This is not unexpected, as software engineers do not strongly perceive, as demonstrated by Question Set 1, the

explicit documentation location K-flow as being particularly problematic. It is worth noting that, although responses relative to work quality can also be summarized as moderate, two software engineers felt that removing this K-flow obstacle would greatly improve their work quality. A perception of moderate improvement in all three categories was reported by managers.

Question Set 3 – Distinguishing Between Versions of the Same Online Radar Project-related Document. This question set asks respondents to speculate on the effect on SPI of a second potential modification to the online Radar Project-related documentation location K-flow. The modification concerns improving software engineers' ability to distinguish between different versions of the same document. Focus group participants expressed that, even if they located the document that they were searching for, it was an additional effort to determine which version was the most recent. However, this question yielded even milder software engineer perceptions regarding work time, meeting deadlines, and work quality than Question Set 2. Additionally, one of the managers answered "Don't know" for all three categories, commenting that he has no insight as to whether or not this is an issue.

4.3.3 Ad Hoc and Personal Knowledge

Members of the focus group communicated obstacles regarding two Radar Project tacit K-flows. They were not the "hot buttons" that K-flows #1 and #2 were. However, the intensity of conversation was not negligible. The first tacit K-flow obstacle discussed was that it is not uncommon for engineers to spontaneously gather

and talk over current Radar Project matters, such as software design options. Accompanying such discourse might be, for example, informal white board drawings and/or text. This type of tacit knowledge sharing was described as very beneficial to the team and with worthy results. However, such ad hoc knowledge is rarely captured, according to the focus group, for future reference. One common problem, as described by a focus group member, is that design decisions are occasionally made during such informal, impromptu engineering knowledge-sharing sessions. However, when the rationale for a design choice is questioned, its origin and evolution are lost.

A second problematic tacit K-flow, as described by the focus group, is the lack of sharing of software engineers' personal notes and experiences with each other. During initial participant interviews, some software engineers mentioned that they keep notebooks of things that they consider of utmost importance to their work and cannot afford to lose. But it was implied that the notebooks' contents were needed for personal use only and, therefore, the knowledge in them not shared.

Figure 23 shows the question sets and corresponding results for the focus group's suggested modifications to the above two K-flow obstacles.

Section 3: Ad Hoc and Personal Knowledge

Key

NFG = Non-focus group
FG = Focus group
M = Manager
O = Observer

A = Analyst
D = Developer
T = Tester

	Question Set 1: If ad hoc knowledge (e.g. whiteboard notes from an informal design discussion) were captured and made conveniently available, it would							Question Set 2: If software engineers reliably documented and shared their personal notes and experiences with each other, it would						
	Sum	NFG	FG	A	D	T	M	Sum	NFG	FG	A	D	T	M
save me a large amount of time when doing my work.	1	1	0		1			1	1	0		1		
save me some time when doing my work.	7	4	3	2	2	3	1	5	3	2	1	2	2	2
have no effect on the time to do my work.	0	0	0				1	2	1	1	1		1	
increase the time to do my work by some amount.	0	0	0					0	0	0				1
increase the time to do my work by a large amount.	0	0	0					0	0	0				
Don't know	0	0	0				1	0	0	0				
greatly increase my ability to meet deadlines.	1	1	0		1			1	1	0		1		
somewhat increase my ability to meet deadlines.	7	4	3	2	2	3	1	4	3	1	1	2	1	2
have no effect on my ability to meet deadlines.	0	0	0				1	3	1	2	1		2	1
somewhat decrease my ability to meet deadlines.	0	0	0					0	0	0				
greatly decrease my ability to meet deadlines.	0	0	0					0	0	0				
Don't know	0	0	0				1	0	0	0				
greatly increase the quality of my work.	1	1	0		1			1	1	0		1		
somewhat increase the quality of my work.	6	4	2	2	2	2	1	4	3	1	1	2	1	2
hve no effect on the quality of my work.	1	0	1			1	1	3	1	2	1		2	1
somewhat decrease the quality of my work.	0	0	0					0	0	0				
greatly decrease the quality of my work.	0	0	0					0	0	0				
Don't know	0	0	0				1	0	0	0				

Figure 23. Section 3, Ad Hoc and Personal Knowledge

Question Set 1 – Capture and Availability of Ad Hoc Knowledge. Question Set 1 asks respondents their perceptions of the capture and availability of ad hoc knowledge on the potential for SPI. Software engineer responses for time savings, meeting deadlines, and work quality were all in the positive range, with the exception of one "no effect" response for work quality. This was somewhat unexpected, given the prioritization of this K-flow as the fourth out of a total of five (see Table 6), and the much smaller amount of conversation covering this issue during the focus group. This question also produced the highest ratings for improvements in work quality throughout questionnaire Sections 1 through 3.

Management, however, had a different perspective, as illustrated by their lower questionnaire ratings and the following comments by two of the managers.

> Normally, engineers record the notes they need during such discussions.

> No reason this can't or isn't being done. There are procedures for storing white papers and technical notes in configuration management.

Question Set 2 – Documentation and Sharing of Personal Notes and Experiences. This question set asks respondents to indicate their perception of the effect of documenting and sharing their personal notes and experiences on SPI. Responses for improvements in time savings, meeting deadlines, and work quality were all in the moderate range for both software engineers and managers, with one exception. One manager responded that documenting and sharing personal notes and experiences would actually increase software engineers' work time. It would, of course, require additional time for engineers to document things that they have not been required to in the past. However, this manager's response implies that he believes that the additional documentation time would not result in a net time savings due to the resultant knowledge sharing. In contrast, no other respondent, engineer or manager, thinks that there would not be at least a break-even in documentation time versus time savings due to knowledge sharing.

4.3.4 Inclusion of a Knowledge Flow Facilitator (KFF) Role

This section presents and discusses the results related to research question six, which concerns evidence to justify the inclusion of a dedicated project-level knowledge management role. It begins with the results of Section 4 of the questionnaire, which asks respondents to indicate their perception of the effect of the inclusion of a Knowledge Flow Facilitator (KFF) role on the Radar Project itself and on software projects in general. It then presents a summary of the portion of the participant interviews that concern a project-level knowledge facilitator role. It concludes with a discussion of an interview with a former Radar Project team member who was perceived by the software engineers as having fulfilled a knowledge facilitation role while assigned to the Radar Project.

4.3.4.1 Questionnaire Results

Question Set 1 – Radar Project Knowledge Flow Facilitator. This first question set asks the respondents their perceptions of the effect of having a Radar Project KFF role on their work time, meeting their deadlines, and their work quality. As shown in Figure 24, all software engineers responded in the positive range for all three categories, with meeting deadlines receiving the highest rankings. Note the shift from other question set responses in the perception that meeting deadlines and work quality would be more positively affected by a KFF than would saving time. That is, for virtually every other question set on the questionnaire, software engineers ranked time savings as the most positively affected aspect of SPI. This leads us to believe

that engineers correctly interpreted our definition of a KFF's role as not doing the software engineers' work for them, either partially or wholly, but as providing assistance to allow the engineers themselves to perform their work more efficiently and effectively.

Section 4: Knowledge Flow Facilitator Role — **Question Set 1**

Key

NFG = Non-focus group
FG = Focus group
M = Manager
O = Observer
A = Analyst
D = Developer
T = Tester

Having a Knowledge Flow Facilitator role on the Radar Project would

	Sum	NFG	FG	A	D	T	M
save me a large amount of time when doing my work.	1	0	1		1		2
save me some time when doing my work.	7	5	2	2	2	3	
have no effect on the time to do my work.	0	0	0				1
increase the time to do my work by some amount.	0	0	0				
increase the time to do my work by a large amount.	0	0	0				
Don't know	0	0	0				
greatly increase my ability to meet deadlines.	3	1	2		2	1	
somewhat increase my ability to meet deadlines.	5	4	1	2	1	2	2
have no effect on my ability to meet deadlines.	0	0	0				1
somewhat decrease my ability to meet deadlines.	0	0	0				
greatly decrease my ability to meet deadlines.	0	0	0				
Don't know	0	0	0				
greatly increase the quality of my work.	2	1	1		2		
somewhat increase the quality of my work.	6	4	2	2	1	3	2
hve no effect on the quality of my work.	0	0	0				1
somewhat decrease the quality of my work.	0	0	0				
greatly decrease the quality of my work.	0	0	0				
Don't know	0	0	0				

Figure 24. Knowledge Flow Facilitator Role (Part 1)

Regarding management, two managers responded in the positive range for all three categories, while one answered "have no effect" for all. For the two managers who had positive responses, their choices of the most positive effect on SPI differed from

that of the software engineers, as they saw time savings as the greatest benefit of a Radar Project KFF role. **T**he highest-level manager perceives that the tasks of the KFF are already performed by other roles on the Radar Project.

> History has shown this person's knowledge quickly becomes out of date and it is cost prohibitive to have this as a standalone role. The data manager on the job should be able to provide 90% of what this role would be in terms of locating and collating documentation but the data manager is not a technical person who could "answer questions". That role generally falls to the team leads.

Also, notice the recurring theme of manager concern with cost.

Question Set 2 – Knowledge Flow Facilitator on Software Development Teams in General. This second question set rewords Question Set 1 in terms of having a KFF role on software teams in general, not just on the Radar Project. In other words, would a KFF be justifiable as a "standard" role, such as a requirements analyst, tester, or architect, on software development teams? Figure 25 shows that software engineers all responded in the positive range for SPI regarding time savings, meeting deadlines, and work quality for project software engineers in general, not just those on the Radar Project. Meeting deadlines marginally received the most positive endorsement.

Section 4: Knowledge Flow Facilitator Role (continued)

Question Set 2

Key

NFG = Non-focus group
FG = Focus group
M = Manager
O = Observer
A = Analyst
D = Developer
T = Tester

Having a Knowledge Flow Facilitator role on *software development teams in general* would

	Sum	NFG	FG	A	D	T	M
save team members a large amount of time when doing their work.	1	0	1		1		2
save team members some time when doing their work.	7	5	2	2	2	3	
have no effect on the time for team members to do their work.	0	0	0				1
increase the time for team members to do their work by some amount.	0	0	0				
increase the time to for team members to do their work by a large amount.	0	0	0				
Don't know	0	0	0				
greatly increase team members' ability to meet deadlines.	2	1	1		2		
somewhat increase team members' ability to meet deadlines.	6	4	2	2	1	3	2
have no effect on team members' ability to meet deadlines.	0	0	0				1
somewhat decrease team members' ability to meet deadlines.	0	0	0				
greatly decrease team members' ability to meet deadlines.	0	0	0				
Don't know	0	0	0				
greatly increase the quality of team members' work.	1	0	1		1		
somewhat increase the quality of team members' work.	7	5	2	2	2	3	2
have no effect on the quality of team members' work.	0	0	0				1
somewhat decrease the quality of team members' work.	0	0	0				
greatly decrease the quality of team members' work.	0	0	0				
Don't know	0	0	0				

Figure 25. Knowledge Flow Facilitator Role (Part 2)

As with Question Set 1, managers perceived time savings as the greatest SPI benefit, once again differing from the views of the software engineers. Likewise, the same manager, who responded with "have no effect" for all three Question Set 1 categories, did so again. He also reiterated his belief that the KFF role is already integrated into other roles on software teams, and that making the KFF a standalone role is cost prohibitive. He did add the following thought.

> Note that program composition and standards regarding data management and documentation are very similar from program to program.

This possibly indicates that this manager interpreted the phrase "software teams in general" in the question as meaning other software teams within the development organization, rather than the world in general outside of the organization, which was the intent.

An oversight by the researcher was the exclusion of a manager-specific question that was the final question on the pilot Manager Questionnaire. This question was, "How much overhead would you be willing to allocate towards a Knowledge Flow Facilitator for a project that you manage?" The purpose of the question was to further probe a manager's perceptions as to the importance of a KFF role when resources were at stake. Possible responses were as follows.

- None
- Less than one-half a person
- One-half a person
- One person
- More than one person
- Don't know

As the Manager Questionnaire had already been distributed and the results received, the researcher used e-mail to send the question to the three manager participants. She set the context for each of the managers by reiterating the final two Manager Questionnaire questions concerning the inclusion of a KFF role and reminding them how they had responded. The three e-mails were sent to the managers on February 7, 2012. Results were received back from all three managers by February 22nd. The results received were "none," "one half a person," and "one person." It is interesting that the response of "none" was received by the highest-level manager participant, who is also the Radar Project manager (the gatekeeper whom the researcher used for access to the Radar Project). The response of "one-half person" was received by the assistant manager, and "one person" by a manager at the next-lowest level, who directly supervises engineers regarding technical issues. The concern over cost decreased and the relevance of the role increased, according to the corresponding comments, as the management role moved down the Radar Project organizational chart. This seems to indicate that the perceived return on investment increases the closer the manager is to the "trenches." Of course, one cannot ignore that it is upper-level management whose job it is to keep a project on budget and, therefore, has a different frame of reference when it comes to the allocation of funds.

4.3.4.2 Participant Interviews

The participant interview guideline questions (Figures 6 and 7) include a question (question eight) regarding the creation of a project-level "knowledge facilitator" role. The question is specifically worded as, "I've been thinking about whether having a designated 'knowledge facilitator' role would help a project team to be more

efficient. I was wondering about your thoughts on this idea." The interviewee had just been introduced to the notion of knowledge facilitation through the conversation provoked by the just-asked question, "Are there people on the project team that you consider 'knowledge facilitators?' That is, they know how to get the knowledge people need or direct them to it." Additionally, the interviewer provided a definition of what he/she meant by a "knowledge facilitator" and introduced the term Knowledge Flow Facilitator (KFF). For consistency between interviews, a prewritten definition (Figure 17) of a KFF was read to the interviewee.

Two aspects of a KFF role were addressed by two separate interview questions. The first was whether the interviewee thought that such a role might benefit the Radar Project. The second was whether he/she thought that it might benefit software development projects in general. The researcher analyzed the results of the discussions by transcribing the portions of the audio recordings relevant to the KFF role verbatim and then looking for common subjects. The quantity of transcription data was not large enough to apply any formal qualitative analytic technique, such as coding. (Strauss and Corbin 1998) The subjects identified are discussed below.

KFF Personality. Some interviewees described personality characteristics that they believed would be necessary for a KFF to have to effectively perform his/her job. Some of the characteristics mentioned were that a KFF needs to naturally want to help people, act as a partner within the team, be technical and "in the middle of things," be able to lead, but not want to build his/her own "empire," be able to

interface well with management, and be a good team advocate. Additionally, it was mentioned that the KFF would need to have the authority to do his/her job. In summary, as expressed by one interviewee, the "right person" would need to be chosen in order for the role to be effective.

Stove Piping. The theme of "stove piping," was mentioned several times. In a matrix organization, it is a common phenomenon that people become very specialized and focused in what they do ("stove-piped") and, not intentionally, keep their specialized knowledge to themselves. The KFF role was seen by some as a way to promote cross-pollination between specialties, resulting in a better-rounded team relative to technical knowledge. In the words of one software engineer,

> We don't have twenty years [of] experience; we have one year's experience twenty times because we make so many of the same mistakes over and over again. The knowledge facilitator would say, "You've got this problem, this person's already solved it over there."

Doubts about the Need for Knowledge Sharing Facilitation. Much pride was expressed by software engineers and managers alike regarding the quality of the technical personnel on the Radar Project and within the development company in general. Because of the perceived quantity and quality of technical knowledge available, and the opinion that most engineers are not afraid to ask questions, some interviewees were not sure that facilitation would add much in the area of knowledge

sharing. One software engineer conveyed his self-confidence in performing his work as follows.

> With 37 years of experience behind me, I've been there, I can fix it. … So, I have a good enough knowledge base that I can either fix a problem or contact the right person.

Another said,

> … I bump into people. I kind of sense about where they are. I'm not really missing anybody. If I need somebody to do something, I pretty much know how to find the person I need to find …

The participants in this case study are generally older, have a large amount of engineering experience, and have been with the development company for a long period of time. So, the above statements certainly seem reasonable. However, it would be informative to know the opinions of younger, less experienced engineers regarding their ability to function on their own and find help (and in a timely manner) when needed. For example, the least experienced software engineer (about nine years of experience, far below the other engineers) mentioned several times in responses to other interview questions that he relied heavily on one of the Radar Project's system engineers for domain knowledge and that he often had to wait for the knowledge.

Some Knowledge Facilitation Already Exists. Some interviewees, both software engineers and managers, conveyed that knowledge facilitation already takes place on the Radar Project, at least to some degree. Generally, this facilitation was attributed to the existence of the position of "lead engineer." The Radar Project is divided into groups, each with an emphasis on some specialized aspect of radar engineering. Each of these groups has a lead engineer. When the KFF role was described to one of the software engineers, he immediately responded,

> When you get into it, that's the lead engineer. That's the job that the lead engineer has.

Knowledge Facilitators Have Existed in the Past. One intriguing theme was that a few interviewees mentioned two people in particular who had worked on the Radar Project in the past and characterized them as "natural" knowledge facilitators. That is, no one had assigned knowledge facilitation as part of their Radar Project duties. They simply took it upon themselves to keep people informed and point them in the right direction. One of these "facilitators" was also mentioned during the focus group, with all three attendees expounding upon the enormous help that this person had been to all of them personally and the Radar Project in general. He was apparently of particular help in the software testing lab. (Recall that the need for assistance with the configuration of lab equipment was rated as a major knowledge need. See Table 6.) Given the number of times that this person was mentioned in interviews and the focus group, the researcher decided to schedule an interview with him to try to determine

exactly what he had provided to the Radar Project team relative to knowledge facilitation that they were not able to fully articulate and also now appear to lack. The results of that interview are summarized in the next section.

4.3.4.3 Former Project Member Interview

The researcher was able to schedule a one-on-one (her PhD student assistant was not present), semi-structured interview with the former Radar Project team member. The interview was conducted on February 14, 2012 at the development company in the same conference room where the other participant interviews had been conducted. The interview guideline questions used are shown in Figure 26. Again, the purpose of the interview was to probe the former member's interpretation as to why several Radar Project members participating in the study considered him to be a valuable and effective knowledge facilitator during his time on the Radar Project. The researcher also wished to explore the "hole" that seemed to have been left, as sensed during the focus group, regarding knowledge facilitation when the member left the project.

1. Could you tell me about the tasks that you performed for the Radar Project?
2. Tell me about how communication worked within the project team.
3. Did other people consult you for knowledge to help them perform their tasks?
4. Were there people on the project team that you considered "knowledge gurus?" That is, they seemed to know something about everything.
5. Were there people on the project team that you considered "knowledge facilitators?" That is, they knew how to get the knowledge people needed or directed them to it.
6. Would it surprise you to know that members of the Radar Project considered you to be a team knowledge facilitator?
7. I've been thinking about whether having a designated "knowledge facilitator" role would help a project team to be more efficient. I was wondering about your thoughts on this idea.
8. Is there anything else that you would like to share with me about the use or flow of knowledge within the project?
9. Is there anything else about the project in general that you would like to share with me?

Figure 26. Former Team Member Interview Guideline Questions

The interview began with the researcher explaining that it was being conducted because the former member's name had come up in study participant interviews and in the focus group (the former member was aware that the case study was underway). She also let the former member know that his name had come up in a positive manner to assure him that the interview would not be a negative experience and that he was not perceived negatively by his former Radar Project colleagues.

The interviewee had worked on the Radar Project for five years in a lead role. He also filled the roles of other leads when those leads were not available. He had considerable knowledge and had the authority to judge whether or not software developed by others was of sufficient quality and sufficiently met customer

requirements to be released as an official build. At an early point in the interview, before the topic of knowledge facilitation was broached, the former member described one aspect of his character as follows.

> I'm told that I guess I'm like a far-reacher for theories and thinking, like trying to think ahead, as in coming up with ideas of stuff that could be a problem … coming up with ideas to try to mitigate any potential problems that might arise."

When discussing his view of how communication worked on the Radar Project when he was a member, he remarked that

> … one thing that you find at least in our world, especially being in this real-time environment is not a lot of people can wrap their heads around this … [gives an example] That's some of the insight that I help to provide. And I understand that. That's the perspective that I happen to be fortunate enough to have.

Both of these self-observations, stated before the discussion of knowledge facilitation began, lend support to the theory that this person has a conceptual frame of reference, whether natural or learned, that is not common to the development company's project members, at least for the projects that he has worked on. It may be this ability, at least

in part, that was the source for his reputation as an excellent technical knowledge facilitator while on the Radar Project.

When asked if the Radar Project team members had used him as a knowledge resource, he said that was the case, and typically on a daily basis because of his role. He said that it often had to do with troubleshooting situations concerning system and debugging knowledge and knowledge having to do with the testing lab. Putting this together with the focus group's discussion of how frequently they consulted him while in the testing lab, and their current disgruntlement with the lack of lab equipment configuration knowledge, we consider it not a far reach that this is the knowledge "hole" that was left when the former member left the Radar Project and that the focus group participants had difficulty articulating.

Outside of technical knowledge, the former member acknowledged that he was also consulted on knowledge issues such as how to write a "politically correct" e-mail or approach another team member for assistance. In summary, he confirmed that he perceived that he had acted, in addition to his regular duties, in a facilitator-type role while on the Radar Project.

Regarding his view of a Radar Project Knowledge Flow Facilitator role, he conveyed that he thought it would be a good addition. He expressed that he had performed half of the responsibilities while on the Radar Project that the researcher listed in the formal definition of a KFF. However, he believed that such a role would work only

under one condition. That condition is that the person would also have to perform a technical role on the project to both earn the technical respect of his teammates and to truly fulfill the role. As he expressed it,

> If you were trying to facilitate somebody with knowledge and help to dictate how something was to work on the program and know that you're getting the right people, you need to have enough knowledge about what's going on, or why something is the way it is …

His opinion of a KFF role for software projects in general was less keen, as he felt that he really hadn't had enough time to consider such a situation.

Chapter 5: Conclusions

This chapter presents a summary of the findings of this dissertation research. It begins with a discussion of the findings from the industrial case study and how they answer the dissertation research questions. It then moves to a comparison of the case study and pilot study findings, and closes with a discussion of the case study limitations.

5.1 Summary of Industrial Case Study Findings

This section summarizes the industrial case study findings and explains how the findings answer each of the dissertation research questions. To review, the research questions are as follows.

I. Current State of Project-level KM

1) What knowledge resources are available to software project engineers?
2) What are the knowledge flows within a software project?
3) What is the context within which knowledge flow occurs?
4) What problems are associated with project knowledge flow?

II. SPI Benefits

5) In what ways could project-level knowledge flow analysis benefit SPI?

III. Project KM Oversight and Facilitation

6) What evidence exists to justify the creation of a dedicated project KM role?

5.1.1 Research Question 1: Knowledge Resources

Concerning the first research question, during interviews and one-on-one K-map validation sessions, Radar Project software engineers (analysts, developers, and testers) enumerated and described the knowledge resources that they consult to do their work. As illustrated in the Radar Project Software Engineer Knowledge Map (Figure 18), Radar Project engineers rely on both explicit (codified) and tacit (knowledge in people's minds) knowledge resources. (Tacit resources are denoted by stick figures, whereas explicit resources are denoted by inanimate objects, such as documents and disk drives.) One can see that there are roughly the same quantities of explicit knowledge resources as tacit surrounding the software engineers, leading one to the conclusion that the reliance on explicit and tacit knowledge by the software engineers is about the same. However, what is not evident from the figure is the frequency of use of and the quantity of knowledge gleaned from each of the resources. The reliance of software engineers upon other people, especially other software engineers, for technical assistance was clear during participant interviews. When the eight individual K-maps were constructed from the audio recordings of the software engineer interviews, each showed reliance by the engineer interviewee on one or more (typically multiple) of their peer engineers for technical knowledge. When these tacit knowledge resources are combined and drawn on the Radar Project Software Engineer K-map, their usage frequency and knowledge quantity, however, are not evident. Using parallel evidence from the individual K-maps, Radar Project

software engineers' reliance on tacit knowledge resources far outweighs their reliance on explicit resources.

The vast majority of resources used by engineers reside within the Radar Project itself. Although some company resources, such as a lessons learned database and a reusable code repository, were mentioned, most do not appear to be utilized by the project engineers, or at least by those who participated in this research. Some resources outside of the company are used, such as vendors, the development company's on-site field service people, and the Internet. But there exists a layer of management, technical leads, and specialized personnel (e.g. configuration management people) between the software engineers and the customer. It was mentioned more than once during initial interviews with both the engineers and managers that direct contact between the engineers and customer personnel is discouraged. This was not articulated as mistrust or other negative concern, but simply as the established process by which communication with the customer occurs.

5.1.2 Research Question 2: Knowledge Flows

The second research question is related to the first in that it concerns the identification of the flows that occur between the resources shown on the Radar Project Software Engineer K-map (Figure 18). Explicit K-flows are indicated by solid lines and tacit K-flows by dashed lines. An explicit K-flow means that the knowledge being transmitted is codified, such as knowledge from a document or diagram. A tacit K-flow means that the knowledge being transmitted is not codified, such as verbal

assistance from a teammate. One can see that there are roughly the same quantities of explicit K-flows as tacit surrounding the software engineers, leading one to the conclusion that the transmission of and reliance on explicit and tacit knowledge by the software engineers is about the same. However, what is not evident from the figure is the "heaviness" of the K-flows. That is, how frequently they occur, how knowledge-intense they are, and their duration. As mentioned in the discussion of Research Question 1 (Section 5.1.1), the reliance of software engineers upon each other for technical assistance was quite evident during participant interviews. When these tacit knowledge flows are combined and drawn on the Radar Project Software Engineer K-map, their heaviness, however, is not evident. Tacit knowledge flows between Radar Project software engineers and other tacit knowledge resources (i.e. people) are, in fact, far heavier than explicit flows.

5.1.3 Research Question 3: Knowledge Flow Context

K-mapping revealed that the common catalyst (or context) for K-flow between software engineers and available knowledge resources, especially other software engineers, is the need for technical knowledge required to progress with their work. This is not unexpected, as software engineers' tasks are highly technical in nature. Clarification of the problem at hand to be solved (as the project is in a maintenance stage) is also frequently sought.

5.1.4 Research Question 4: Knowledge Flow Problems

The software engineer focus group identified five problematic K-flows, which are prioritized in Table 6 from the most likely to result in SPI to the least, according to the consensus of the focus group. For each of these K-flows, the group also identified potential improvements as candidate modifications for SPI. Conclusions regarding all software engineers' perceptions, not just the perceptions of those in the focus group, regarding SPI effectiveness are given below.

- The lack of knowledge with which to configure the lab testing equipment is considered the K-flow whose mitigation or removal would be the most likely to result in SPI in the areas of saving software engineers time in doing their work and meeting their deadlines.
- Consistent lab equipment oversight, having a dependable person available to help with the configuration and use of the equipment, and having the lab equipment found in a standard (default) configuration are all considered viable options, to varying degrees, for SPI in the areas of saving software engineers time in doing their work and meeting their deadlines. The one explicit knowledge solution (i.e. a "cookbook" of equipment configurations) was received far less favorably than the three "people" solutions just enumerated, supporting the preference by engineers for tacit, human resources over explicit resources.
- Project documentation location and version identification does not appear to be as problematic to the Radar Project software engineers in general as was

portrayed by the focus group. This may be due to the selection of focus group members whose involvement with the Radar Project is the most recent. That is, they are more likely to have searched for project documents more recently than the non-focus group engineers. Regarding this particular K-flow, the focus group may not be representative of the entire software engineer participant population.

- Software engineers would find it beneficial in the areas of saving them time to do their work, meeting their deadlines, and increasing their work quality if ad hoc knowledge were captured and made conveniently available. This K-flow improvement was considered by far the most beneficial modification in the area of increasing individual work quality. The importance of this K-flow modification, especially on work quality, was unanticipated, as it was given a priority of four out of five by the focus group as to its potential effect on SPI. Perhaps this improvement is viewed as a partial solution to the problem of the "stove piping" of knowledge that was discussed in section 4.3.4.2.
- Documenting and sharing of personal notes and experiences is seen by software engineers as being of moderate benefit to SPI in all three categories.

Conclusions regarding managers' perceptions of the suggested K-flow improvements are summarized below.

- Managers are in agreement with software engineers that the most problematic K-flow is that of lab equipment configuration knowledge.

- Managers are also in agreement with the engineers that consistent lab equipment oversight, having a dependable person available to help with the configuration and use of the equipment, and having the lab equipment found in a standard (default) configuration are all viable options, to varying degrees, for SPI in the areas of saving software engineers time in doing their work and meeting their deadlines.
- Managers are in agreement with the engineers that project documentation location and version identification is not as problematic to the Radar Project software engineers in general as was portrayed by the focus group.
- Managers differed greatly from software engineers in their views regarding the capture and availability of ad hoc software engineer knowledge. They see it as essentially no benefit to software engineers in time savings, meeting deadlines, or work quality. According to their explanations in the open-ended comment area, they believe that software engineers already capture this knowledge on their own.
- Managers agree that documenting and sharing of personal notes and experiences is of moderate benefit to SPI in all three categories.

5.1.5 Research Question 5: Project-level Knowledge Flow Benefits to SPI

We believe that substantial findings were made regarding the ways that project-level K-flow analysis is of benefit to SPI. Foremost is that the people "in the trenches," i.e. the software engineers, demonstrated the ability to identify problematic project-level K-flows that they perceive to negatively impact the time that it takes them to perform

their work, meet their deadlines, and produce quality work. They were also able to propose solutions to these problematic K-flows. These solutions were confirmed as perceived as positive steps towards SPI not only by the software engineers who proposed them, but also by fellow project software engineers. Likewise, software engineers were able to identify K-flows that they perceive as indispensable and advantageous to their efficiency and effectiveness, exposing K-flows that can be exploited for SPI. In general, the above provides support for the use of the experiences and observations of project-level personnel for SPI purposes. Last, these findings also lend support for the efficacy of project-level knowledge mapping, in the context of this study design, for identifying viable SPI opportunities.

5.1.6 Research Question 6: Evidence for a Dedicated Project KM Role

There is considerable support by software engineers and more moderate support by managers for a Knowledge Flow Facilitator (KFF) role on both the Radar Project and software development projects in general.

- Engineers intuit that having a KFF role on the Radar Project would benefit them by helping them to save time, meet their deadlines, and improve the quality of their work.
- Engineers rated meeting deadlines and improving work quality over time savings relative to implementing a Radar Project KFF role. Time savings had the most positive ratings for virtually every other K-flow modification on the Software Engineer Questionnaire.

- Regarding a KFF role on software development projects in general, software engineers once again perceived that such a role would help other software engineers save time, meet their deadlines, and improve the quality of their work.

In addition, software engineers expressed during their individual interviews that the person(s) filling the KFF role would need to be the "right person," that is, technical, "in the middle of things," and having the ability to lead. The leadership characteristic was endorsed not only explicitly during engineer interviews, but also implicitly by the fact that the two former Radar Project team members who were deemed "natural facilitators" both held team leadership roles when on the project (one was a lead engineer and the other, whose interview was discussed in Section 4.3.4.3, was the team software integration lead).

Conclusions regarding managers' perceptions of the inclusion of a KFF role on the Radar Project and in general are summarized below.

- Managers were more moderately positive regarding the inclusion of a KFF role on the Radar Project.
- Managers rated saving software engineers' time over meeting their deadlines and improving their work quality regarding the effect of a Radar Project KFF role. This is opposite to the software engineer responses.

- Regarding a KFF role on software development teams in general, managers responded identically as they did to the inclusion of a Radar Project KFF role.

The above software engineer and manager findings substantiate the need for further investigation into the inclusion of a project-level KFF role.

5.2 Comparison of Pilot and Industrial Case Study Findings

This section compares the findings of the pilot study (ERP3 Project) with the industrial case study (Radar Project) relative to the six research questions.

The ERP3 Project and the Radar Project were selected by convenience. Therefore, the values of the projects' parameters were not under our control. Table 9 shows the values of some of the major parameters that characterize each of the projects.

Parameter	ERP3 Project	Radar Project
Organization type	Public university	US Department of Defense contractor
Organization mission	Education	Software and hardware development
Organization size	Medium (~1,600 employees)	Very large (~10,000 employees at the Radar Project location)
Project size (in members)	~30 full-time at peak	~70 full-time equivalent at peak
Project category	Enterprise Resource Planning (ERP)	Embedded systems
Development process	Non-specific	Company defined
Customer type	Internal	External
Management style	Functional	Matrix
Team member location	Co-located	Co-located
Average participant time at organization	~11 years	~19 years

Table 9. Comparison of ERP3 Project and Radar Project Descriptive Values

One can see that these two projects' attributes differed in many ways. However, they still covered only a small portion of the many variations of the universe of software projects. For example, neither involved development within the commercial software development sector or the open source community. Neither team was small (e.g. less than 10 members) and neither followed a "book-defined" development process, such as the Rational Unified Process (RUP) (Kruchten 2000) or SCRUM (Schwaber and Beedle 2002). Opportunities are plentiful for the replication of this study utilizing permutations of these and other project parameter values.

5.2.1 Knowledge Resources, K-Flows, and K-Flow Context

The pilot study and the case study both exhibited dependence by software engineers on a large number of both explicit and tacit knowledge resources and flows (first and second research questions). As explained in section 5.1.2, the Radar Project engineers lean more towards a reliance on within-project tacit knowledge exchange than codified knowledge. It is not as evident which category of knowledge the ERP3 Project (the pilot study) participants relied on more, as there was heavy dependence on the codified knowledge stored on the ERP3 Project shared disk drive, yet an overwhelming opinion among engineers that team co-location was a major contributor to the project's success. However, in conclusion, both explicit and tacit knowledge resources and flows play major roles in software engineering knowledge capture, supply, and sharing.

Regarding knowledge flow context (third research question), the major catalysts for K-flow with respect to the ERP3 Project was the need for either domain or technical knowledge. Domain knowledge was typically needed for understanding the workflow of the university's various service areas (e.g. student records, student financials), and was found within the university itself. Technical knowledge was typically needed to help understand how the Enterprise Resource Planning (ERP) software that was being installed functioned and to help with its installation and customization. This knowledge came from a variety of sources such as fellow software engineers, on-site consultants, the ERP vendor, other university web sites, and Internet searches. That is, the "reach" of the desired knowledge was long, coming from the outside world (e.g. the ERP3 vendor and the Internet), through the organizational and department levels, to the project level. Although the major catalysts for K-flow for The Radar Project software engineers was also the need for domain or technical knowledge, it leaned toward the technical, such as the need for knowledge regarding the configuration of the lab equipment. Also, as previously described, the "reach" of the desired knowledge was typically short, frequently able to be located within the Radar Project team itself. One reason for this short knowledge reach, as discussed in section 5.1.1, is that a layer of management, technical leads, and specialized personnel (e.g. configuration management) exists between the software engineers and the project's external customer. This was not the case with the ERP3 Project, as its customer was internal (the university itself). The Radar Project software engineers' general lack of ability to directly interface with the customer may, to some extent, explain the high interdependence of software engineers for technical knowledge.

5.2.2 Knowledge Flow Problems

As shown in Table 9, the ERP3 Project and Radar Project characteristics differed in numerous ways. To what degree, if any, did this affect the overall outcomes between the studies?

To begin, both studies were successful in the location by software engineers of perceived problematic K-flows (fourth research question). Being two different projects, they obviously were not the same K-flows, but fell into both the explicit and tacit categories for both projects. The top five ERP3 Project K-flow obstacles were divided between two themes: project document storage and retrieval problems (explicit K-flow) and intra-team communication (tacit K-flow). The top five Radar Project K-flow obstacles were more diverse, consisting of trouble with project document knowledge retrieval (explicit K-flow), lack of sufficient and/or accurate lab equipment configuration knowledge (explicit and tacit K-flows), insufficient ad hoc knowledge capture (explicit K-flow), and lack of personal knowledge sharing (explicit and tacit K-flows). The larger diversity of K-flow obstacle themes identified by the Radar Project focus group may be due to the higher level of richness of the Radar Project Map (Figure 18) relative to the ERP3 Project K-map (Figure 10). Recalling, the Radar Project Map was constructed from eight individual software engineer K-maps, while only three engineer K-maps were used to construct the ERP3 Project K-map. So, although the ERP3 focus group did not need a highly rich map for the location of K-flow obstacles (that were later confirmed to be perceived as

obstacles by all of the ERP3 engineers), perhaps a richer map would have revealed more diversity.

Both studies were also successful in formulating potential solutions (modifications), as perceived by the focus groups, to the problematic K-flow. All potential K-flow obstacle solutions were also viewed by all project software engineers, not just those on the focus groups, as having some potential for a positive effect on SPI (fifth research question). Intensity of support varied from modification to modification, and between K-flow themes (e.g. project document K-flow versus intra-team communication K-flow), which one would expect, as the software engineers would generally be affected to different levels by each of the obstacles.

Managers' support for K-flow modifications for both projects also varied from modification to modification, theme to theme. Unfortunately, only one of the two ERP3 managers responded to the Manager Questionnaire. His responses indicated that he had no insight into the project document retrieval problem and essentially no support for its solutions. Yet he acknowledged the intra-team communication problem and supported both solutions. It may be that problems with tacit K-flow (e.g. finding the person with the needed knowledge, knowledge sharing) are more apparent to this manager than explicit K-flow problems, as he would be more likely to witness or be consulted about such problems, especially being co-located with the ERP3 team.

Three Radar Project managers responded to the Manager Questionnaire, giving a better comparison of results between software engineers and managers than for the ERP3 Project. Managers' responses fell very generally into the same pattern of responses to the acknowledgement of K-flow obstacles and the proposed solutions as software engineers with the exception of Section 3, Question Set 1, which asked, "If ad hoc knowledge (e.g. whiteboard notes from an information design discussion) were captured and made conveniently available, it would …" Only one of the managers felt that the capture of ad hoc knowledge would somewhat benefit software engineers with reducing their work time, helping them meet deadlines, and improving the quality of their work. The other two managers indicated by their comments that they believed that the capture and sharing of ad hoc knowledge was already being done. We have no speculation as to why this was so and did not have time to pursue this path.

Last, K-mapping was supported by both studies, in the context of this research design, as being an effective technique for identifying viable SPI opportunities. K-mapping was used in two very different project domains with favorable results, adding to the evidence for the efficacy of K-mapping for K-flow obstacle location. (Hansen and Kautz 2004) Additionally, the individual participant K-map validation sessions conducted for both the pilot and industrial case studies were found to be vital to confirming that the researcher and her PhD student assistant had accurately captured the participant interview data relative to knowledge resources and flows. (Lincoln and

Guba 1985) Recall that it was the individual participant K-maps that were the basis for both the software engineer and manager project K-maps.

5.2.3 KFF Role

This section summarizes the findings regarding research question 6, which asks, "What evidence exists to justify the creation of a dedicated project KM role?"

Software engineers from the Radar Project were overwhelmingly open to the proposal of the incorporation of a Knowledge Flow Facilitator (KFF) into the Radar Project. They were equally as positive regarding a KFF role on software development projects in general. Although the opinions of the ERP3 Project engineers were generally positive on both accounts, they were more cautiously so. Looking back, the ERP3 engineers responded to these questions shortly after the ERP3 Project came to its conclusion, whereas the Radar Project engineers responded while the project was ongoing, albeit during a "hold" period where only maintenance activities were being performed. The ERP3 project engineers may have simply felt the way that they responded. Or, perhaps, certain aspects of projects do not look as bad in retrospect. Or, more generally, the stage that a project is in may be likely to skew, one way or the other, the views of the project members.

The single ERP3 manager respondent expressed that a KFF role would have had no effect on time savings, deadlines, or work quality for the ERP3 Project software engineers. He also reported that he would not be willing to allocate any overhead for

a KFF role on a project that he managed. He responded somewhat more positively for a KFF role on software development projects in general. Contrasting this with the three Radar Project Managers, two of the managers saw the KFF role as being beneficial for both the Radar Project and in general and would be willing to allocate one person and one-half a person, respectively. The remaining manager responded "no effect" for both questions in all three categories (time, deadlines, and work quality). He also indicated that he would not be willing to allocate any overhead towards a KFF for a project that he managed. His accompanying comment was quite informative.

> By overhead I read that to mean [company name] overhead dollars as opposed to project dollars. If that is a correct reading, then I would say none. A KFF role on a program would need to be paid by program (customer) dollars which would increase our Level of Effort component to any technical job. … We receive significant scrutiny and challenge by our customers on this category of work. Adding a KFF role to that work category would require significant justification and some proof of pay-off … While I see some merit to the this role, as stated in previous questions I question its ability to stay current and effective without significant investment - an investment competitive bidding would find difficult to support.

Again one sees how significant project budget is to managers, as it is one of their critical responsibilities. His response also raises the question as to whether managers would respond differently if asked if they would be willing to allocate project (customer) funds towards a KFF role.

5.3 Limitations

As with any research study, there are certain limitations that bound its findings. The most significant limitation to this study is that it is a study of a single software development team, and a subset of the team at that. The software team has its own set of characteristics, such as its size, composition, and type of product that it is developing. No claims of generalizability are being made regarding this study. Rather, its purpose is to take the first steps in building a theory regarding the effect of the removal of project-level K-flow obstacles on SPI and to provide propositions for further inquiry.

A second limitation to this research was timing. When the researcher became available to begin the study, the Radar Project was in a "hold" pattern, waiting for its customer to catch up with testing the software that the Radar Project had already delivered. It was, therefore, only performing maintenance on previously delivered software. We were faced with the decision of whether to conduct the study as if development were still underway or as it currently was, in a slower-paced maintenance mode with fewer personnel. We elected to conduct the study from the maintenance mode perspective, not wanting to rely on participant recall of what they

had done and seen during full development, which had been several months prior. To counter this limitation to some degree, participants were chosen who had been through multiple development life cycles and, therefore, were experienced in performing maintenance activities.

Another limitation was that the Radar Project was a U.S. Government classified project, housed in a classified facility. Therefore, we were unable to conduct interviews in situ. Although interviews were not conducted in the engineers' "natural" surroundings, we were able to at least conduct them all in the same non-distracting environment (a conference room). Also due to the classified nature of the project and the facility, we were unable to see the software engineers' working environment, particularly the laboratory where the engineers run their software tests using the radar hardware. As the Software Engineer Questionnaire results revealed, three of the lab equipment configuration K-flow modifications were perceived as the most likely to result in SPI. Although this study is exploratory, not explanatory, in nature, visiting the lab would have been an opportunity to possibly further understand the software engineers' hardships and frustrations with configuring the lab equipment before testing.

Last, this study explored participant perception, which is not necessarily the same as reality. Triangulation in data gathering was our approach to this common research dilemma. The triangulation approaches that we used included using two researchers to conduct the initial interviews and the focus group, interviewing a previous member

of the Radar Project, interviewing managers in addition to the software engineers and constructing a Radar Project Manager K-map, and implementing two questionnaire types (Software Engineer and Manager).

Chapter 6: Contributions

This study contributes to the state of empirical research in the field of software process improvement (SPI) by building upon previous studies and putting forth and supporting a new approach to SPI through the identification and removal of project-level knowledge flow (K-flow) obstacles. Contributions to future research consist of propositions upon which to further explore and improve this approach. Assistance for practitioners is provided through SPI practices used in this study's research design that may be applied to current software development projects. Each of these contributions is described below.

6.1 Research Contributions

The major contribution of this research is that it puts forth and provides support for a new approach for software process improvement (SPI). The approach advocates the application of the business field of knowledge management to the technical field of software engineering for SPI. Although the marriage of these two fields is not a unique idea, this study advocates several novel approaches. For example, essentially all previous studies exploring the effect of KM on SPI promote the application of KM at the software organizational level. That is, they apply KM methods at the top levels of the organization rather than at the "bottom," or project, level. *This research applies KM at the project-level*, taking advantage of the perspectives and experiences of the people "in the trenches," the software engineers. It is our belief that these people are in the position to have the most well-informed, current input about software

development practices and problems and are, therefore*, the most likely population to have the awareness necessary to significantly contribute to SPI*.

Two case studies, a pilot and its follow-on industrial case study, were conducted to test the proposition that project-level K-flow obstacle removal would have a positive effect on SPI. Both studies resulted in the identification of problematic project-level K-flows and viable K-flow modifications such that the modifications, as perceived by project software engineers, would result in SPI. Furthermore, *the characteristics of the two projects utilized in this research differed markedly* (e.g. type of product being development, development process used, team member demographics), *yet the overall proposition was upheld*.

This study also endorses the *use and flow of knowledge from the present in the present*, during software product development. Other KM for SPI techniques rely on "knowledge at rest." That is, they capture project and process knowledge and store it for use on future projects. Although this is an undertaking that has proven its worth (Basili, Caldiera et al. 1994), it does not take advantage of the use and flow of knowledge as it is created, enhanced, and shared in the present, during software product development. *This knowledge, if recognized and encouraged to flow, has the ability to provide "real-time" SPI in the present, as well as contribute to future SPI.*

A strong theme among the empirical studies on KM for SPI is the need to recognize and integrate the use of tacit knowledge for SPI. This is consistent with the recent

movement towards agile software development, which advocates the lessening of codification and the increase of person-to-person knowledge transfer. *This study validated the necessity and desire for tacit knowledge sharing among software engineers.*

A second proposition, that the integration of a project-level Knowledge Flow Facilitator (KFF) role will promote SPI, was supported by the opinions of the software engineers from both the pilot study and the industrial case study. *Engineers viewed the KFF role as a positive step towards SPI for both their projects and software development projects in general.* Again, it should be noted that this proposition, as with the main dissertation proposition, was supported by software engineers from two projects with decidedly different structural, domain, and demographic characteristics.

Last, this research provides further support for the use of Hansen and Kautz's K-mapping technique. (Hansen and Kautz 2004) Although too labor-intensive to recommend widespread usage in software practice, K-mapping is a valuable research tool for a viable approach to 1) capturing the state of an entity's knowledge and knowledge flows, and 2) locating K-flow obstacles.

6.2 Contributions to the Practice

Software engineering is a comparatively young field. As such, the vast majority of the SPI literature is merely prescriptive (e.g. CMMI) or descriptive (industry reports) in

nature. There are but a modest number of empirical studies to help practitioners with SPI. The conduct of this empirical research is another small piece to add to the SPI body of knowledge. And although new and with a minor amount of empirical support, *it is composed of some discrete steps that practitioners can apply to their current SPI efforts*. For example, project managers or other assigned personnel can solicit perceived K-flow obstacles from project personnel, investigate the obstacles, and, if applicable, take steps to eliminate or mitigate them. They can also take steps to exploit existing "positive" K-flows to enhance personnel efficiency and effectiveness.

The study's revelation of the contributions to SPI that can be made at the project-level by software engineers provides practitioners with a perspective on approaching SPI from the "bottom-up." Coupled with this bottom-up approach is the realization that there is an associated "trickle-up" effect. That is, if software engineers are able to shorten the time that it takes them to do their jobs, increase their ability to meet their deadlines, and improve their work quality*, the effects of these improvements trickle up, contributing to project cost reductions, schedule adherence, and production of a better quality end-product*. That is, SPI.

Another unique contribution for practitioners is the recognition that *SPI resulting from K-flow analysis can be produced in "real-time."* That is, suggestions for and adjustments to project knowledge management need not be handled only at the beginning of a project based on the metrics and experiences from previous projects. They can be, and should be, *made continuously throughout the SDLC*. This ties back

to the concept communicated in section 6.1 of the use and flow of knowledge from the present in the present.

Last, this study has provided evidence to practitioners that *K-flow obstacles are considered by software engineers to have negative effects on their efficiency and effectiveness*. Although the evidence presented is in the form of perceptions, it is the perceptions of the software engineers themselves, not outside observers. The obvious solution here is the removal or mitigation of problematic K-flows. Additionally, *continuous K-flow analysis may reveal positive K-flows that can be exploited*. For example, the Radar Project revealed that software engineers prefer tacit knowledge sharing over the consultation of explicit (codified) resources.

6.3 Future Research

As this study was exploratory rather than explanatory in nature, it provides no testable hypotheses, but propositions for future research. One reasonable proposition is to attempt to strengthen this research study by the performance of other case studies using the same or different research design. As this study relied on perceptions for its findings that project-level K-flow obstacle removal has the potential to positively affect SPI, actually removing an obstacle and observing the effect on SPI would be an invaluable extension. Starting with the removal of a known K-flow obstacle would be a low-cost, low-risk approach. If the findings were promising, the more lengthy and costly process of both locating and removing a newly discovered obstacle might be a next step.

This research study relied on the technique of K-mapping to locate knowledge obstacles. The technique is long and arduous, but was necessary, as the researcher was completely unfamiliar with the projects and participants that she was to study. It provided a suitable method, within this study design, for her to become more intimate with the workings of the project and the frames of reference of the participants. It also provided a mechanism for her to validate that she had accurately captured the participants' points of view. Certainly an organization would not want to invest such a large amount of time and expense to locate K-flow obstacles, especially on a continuous basis. A valuable area to explore would be how to simplify the procedure for accurate K-flow obstacle location, perhaps by using a method other than K-mapping.

Regarding building on the Knowledge Flow Facilitator (KFF) role research, exploring how knowledge facilitation is currently performed, if at all, within a project could clarify potential KFF characteristics and duties and further explore whether the justification for a KFF role exists. Some reasonable research questions are as follows.

- If project-level knowledge facilitation is currently performed, who performs it?
- How do they perform it? Are there designated duties, or is it ad hoc?
- Is it effective? If so, in what ways?
- Does it result in SPI?

- Is there enough facilitation going on that it makes sense to combine it into a single role?
- How does cost play into the picture?

6.4 Summary

This research was undertaken in the context of the uncontrollable costs and schedule overruns that continue to plague the software engineering field. It recommends the application of the business field of knowledge management (KM) to the technical field of software development as a solution, in part, to this dilemma. It provides a novel proposition and support for the detection and removal of project-level knowledge flow (K-flow) obstacles in order to achieve software process improvement (SPI).

The research design is that of an exploratory case study of an industrial software development project. The subjects utilized were a subset of the project software engineers (analysts, developers, and testers). Data was collected from the engineers regarding the knowledge resources that they consult (both human and non-human), the knowledge that flows among the resources (K-flows), and the context within which K-flow occurs, as they perform their jobs. The data was combined and transformed using a technique called K-mapping into an illustration of the knowledge usage and flow within the project as a whole. A focus group consisting of a subset of the subject engineers used the K-map to select what they perceive to be obstacles to K-flow, causing inefficiencies in their day-to-day work and impacting their work

quality. They also provided what they believe to be ways to mitigate or remove the problematic K-flows.

Using the results of the focus group, a questionnaire was constructed probing the perceptions of all of the software engineer participants, not just those from the focus group, as to what effect, if any, the suggested K-flow obstacle solutions would have on saving them work time, meeting their deadlines, and improving their work quality. Results were generally positive, to varying degrees, for all three categories (time, deadlines, and work quality), supporting the study proposition that the removal or mitigation of project-level K-flow obstacles results in some degree of SPI.

The major contribution of this research is that it puts forth and provides empirical support for a new approach for SPI. The approach differs from others in that it proposes applying KM at the project-level rather than the organizational level, using the perspectives and experiences of the people "in the trenches," the software engineers, rather than management. It endorses the use and flow of knowledge from the present in the present, during the software development process, rather than simply relying on the knowledge gathered from the conduct of previous projects (knowledge "at rest"). It, therefore, provides "real-time" SPI, allowing informed work and process adjustments to be continuously made as the project progresses. Additionally, the approach accommodates the utilization and optimization of tacit (in one's head) as well as explicit (codified) knowledge and K-flows, which is more in line with the movement towards agile software development techniques. Although

this approach to SPI is new, with a minor amount of empirical support, it is composed of some discrete steps that practitioners can apply to their current SPI efforts.

Bibliography

Agila, S. A. and Z. Sun (2004). Knowledge Management: Impact of Knowledge Delivery Factors on Software Product Development Efficiency. IEEE International Conference on Information Reuse and Integration, Las Vegas, NV.

Agostini, A., S. Albolino, et al. (2003). Stimulating Knowledge Discovery and Sharing. GROUP'03, Sanibel Island.

Alazmi, M. and M. Zairi (2003). "Knowledge Management Critical Success Factors." Total Quality Management and Business Excellence **14**(2): 199 - 204.

Arent, J. and J. Norbjerg (2000). Software Process Improvement as Organizational Knowledge Creation: A Multiple Case Analysis. Hawaii International Conference on System Sciences, Maui, Hawaii.

Arent, J., J. Norbjerg, et al. (2000). Creating Organizational Knowledge in Software Process Improvement. 2nd Workshop on Learning Software Organizations, Oulu, Finland.

Basili, V. R., G. Caldiera, et al. (1994). The Experience Factory. Encyclopedia of Software Engineering. J. J. Marciniak, John Wiley**:** 469-476.

Bjornson, F. O. and T. Dingsoyr (2008). "Knowledge Management in Software Engineering: A Systematic Review of Studied Concepts, Findings, and Research Methods Used." Information and Software Technology **50**: 1055-1068.

Bontis, N. (2002). "The Rising Star of the Chief Knowledge Officer." Ivey Business Journal **66**(4): 20-25.

Buono, A. F. and F. Poulfelt (2005). Challenges and Issues in Knowledge Management Greenwich, Information Age Pub.

Crosby, P. B. (1996). Quality is Still Free: Making Quality Certain in Uncertain Times. New York, McGraw-Hill.

Davenport, T. H., D. W. D. Long, et al. (1997). Building Successful Knowledge Management Projects**:** 24.

Davenport, T. H. and L. Prusak (1998). Working Knowledge : How Organizations Manage What They Know Boston, Harvard Business School Press.

Debenham, J. and J. Clark (1994). The Knowledge Audit. Robotics and Computer Integrated-Manufactoring, Pergamon. **11:** 201-211.

Deming, W. E. (1982). Out of the Crisis. Cambridge, MA, Productivity Press.

Denzin, N. K. (1978). The Research Act: A Theoretical Introduction to Sociological Methods. New York, Mc-Graw-Hill.

Desouza, K. C., Y. Awazu, et al. (2006). "Factors Governing the Consumption of Explicit Knowledge." Journal of the American Society for Information Science and Technology **57**(1): 36-43.

Dyba, T. (2005). "An Empirical Investigation of the Key Factors for Success in Software Process Improvement." IEEE Transactions on Software Engineering **31**(5): 410- 424.

Earl, M. J. and I. A. Scott (1999). "Opinion: What is a Chief Knowledge Officer? ." Sloan Management Review **40**(2): 29-38.
Feher, P. and A. Gabor (2006). "The Role of Knowledge Management Supporters in Software Development Companies." Software Process: Improvement and Practice **11**(3): 251-260.
Fowler, M. (2003). UML Distilled: A Brief Guide to the Standard Object Modeling Language, Addison-Wesley Professional.
Fowler, M. and J. Highsmith (2001). The Agile Manifesto. Software Development Magazine.
Hansen, B., J. Rose, et al. (2004). "Prescription, Description, Reflection: The Shape of the Software Process Improvement Field " International Journal of Information Management **24**(6): 457-472.
Hansen, B. H. and K. Kautz (2004). Knowledge Mapping: A Technique for Identifying Knowledge Flows in Software Organisations. Lecture Notes in Computer Science. T. Dinsoyr. Heidelberg, Springer Berlin. **3281:** 126-137.
Hansen, M. T., N. Nohria, et al. (1999). "What's Your Strategy for Managing Knowledge? ." Harvard Business Review **77**(2): 106-116.
Humphrey, W. (2000). Introduction to the Team Software Process. Reading, Addison Wesley.
Humphrey, W. (2005). The Personal Software Process: A Self-Improvement Process for Software Engineers. Upper Saddle River, Addison Wesley.
Juran, J. M. and F. Gryna (1988). Juran's Quality Control Handbook. New York, McGraw-Hill Book Company.
Kitchenham, B. (2004). Procedures for Performing Systematic Reviews, Keele University and Empirical Software Engineering National ICT Australia Ltd.
Kitchenham, B. A., T. Dyba, et al. (2004). Evidence-based Software Engineering. 26th International Conference on Software Engineering.
Kruchten, P. (2000). The Rational Unified Process: An Introduction. New York, Addison-Wesley.
Kulpa, M. K. and K. A. Johnson (2008). Interpreting the CMMI: A Process Improvement Approach. New York, CRC Press.
Lanzara, G. F. and L. Mathiassen (1985). "Mapping Situations within a System Development Project." Information and Management **8**: 3 - 20.
Lawton, G. (2001). "Knowledge Management: Ready for Primetime?" Computer **34**(2): 12-14.
Liebowitz, J. (1999). "Key Ingredients to the Success of an Organization's Knowledge Management Strategy." Knowledge and Process Management **Volume 6**(1): 37 - 40.
Liebowitz, J., B. Rubenstein-Montano, et al. (2000). "The Knowledge Audit." Knowledge and Process Management **7**(1): 3-10.
Lincoln, Y. S. and E. G. Guba (1985). Naturalistic Inquiry. Thousand Oaks, Sage Publications, Inc.
Lutters, W. G., M. S. Ackerman, et al. (2000). Mapping Knowledge Networks in Organizations: Creating a Knowledge Mapping Instrument. Americas Conference on Information Systems (AMCIS 2000).

Melnik, G. and F. Maurer (2004). Direct Verbal Communication As a Catalyst of Agile Knowledge Sharing. Agile Development Conference, Salt Lake City, UT.

Merriam, S. B. (2009). Qualitative Reseaerch: A Guide to Design and Implementation. San Francisco, Jossey-Bass.

Monk, A. and S. Howard (1998). "Methods & Tools: The Rich Picture: A Tool for Reasoning About Work Context." Interactions **5**(2): 21-30.

Niazi, M., D. Wilson, et al. (2006). "Critical Success Factors for Software Process Improvement Implementation: An Empirical Study." Software Process: Improvement and Practice **11**(2): 193-211.

Nissen, M. E. (2002). "An Extended Model of Knowledge-Flow Dynamics." Communications of the Association for Information Systems **8**: 251-266.

Nissen, M. E. (2006). Harnessing Knowledge Dynamics: Principled Organizational Knowing & Learning IRM Press.

Nonaka, I. (1994). "A Dynamic Theory of Organizational Knowledge Creation." Organization Science **5**(1): 14-37.

Nonaka, I. and H. Takeuchi (1995). The Knowledge-creating Company: How Japanese Companies Create the Dynamics of Innovation New York, Oxford University Press.

Pandey, A. M., Corey; Paul, Manoj; Kameli, Nader; Boudigou, Francoise; Vijay, Vivek; Eapen, Abraham; Sutedjo, Imelda; Mcdermott, Wesley (2003). Application of Tightly Coupled Engineering Team for Development of Test Automation Software – A Real World Experience. Computer Software and Applications Conference (COMPSAC) 2003, IEEE.

Peratec (2009). Total Quality Management: The Key to Business Improvement, Chapman & Hall.

Petrash, G. (1996). Managing Knowledge Assets for Value. Knowledge-Based Leadership Conference, Linkage, Inc.

Platt, L. E. (1996). Chairman, president and CEO of Hewlett-Packard, quoted in Bowles, J. and Hammon, J., Competing on Knowledge. Fortune magazine, supplement.

Rainer, A. and T. Hall (2002). "Key Success Factors for Implementing Software Process Improvement: A Maturity-based Analysis." Journal of Systems and Software **62**(2): 71-84.

Ravichandran, T. and A. Rai (2003). "Structural Analysis of the Impact of Knowledge Creation and Knowledge Embedding on Software Process Capability." IEEE Transactions on Engineering Management **50**(3): 270-284.

Rodriguez-Elias, O. M., A. Martinez-Garcia, et al. (2009). "Modeling and Analysis of Knowledge Flows in Software Processes through the Extension of the SPEM." International Journal of Software Engineering and Knowledge Engineering **19**(2): 185-211.

Royce, W. (1970). Managing the Development of Large Software Systems: Concepts and Techniques. IEEE WESCON, IEEE.

Salo, O. (2005). Systematical Validation of Learning in Agile Software Development Environment. Lecture Notes in Computer Science. Heidelberg, Springer Verlag**:** 106-110.

Schwaber, K. and M. Beedle (2002). Agile Software Development with Scrum. Upper Saddle River, Prentice-Hall, Inc.

Segal, J. (2001). Organisational Learning and Software Process Improvement: A Case Study. Lecture Notes in Computer Science. Heidelberg, Springer Verlag. **2176:** 68-82.

SEI. (2009). "About the SEI." Retrieved 5/12/09, from http://www.sei.cmu.edu/about/.

Strauss, A. and J. Corbin (1998). Basics of Qualitative Research: Techniques ad Procedures for Developing Grounded Theory. Thousand Oaks, SAGE Publications, Inc.

SurveyMonkey (2009-2011). SurveyMonkey.

Sy, T. and S. Cote (2004). "Emotional Intelligence: A Key Ability to Succeed in the Matrix Organization." Journal of Management Development **23**(5): 437-455.

Trittmann, R. (2001). The Organic and the Mechanistic Form of Managing Knowledge in Software Development. Lecture Notes in Computer Science. Heidelberg, Springer Berlin. **2176/2001:** 22-36.

Ward, J. and A. Aurum (2004). Knowledge Management in Software Engineering - Describing the Process. Australian Software Engineering Conference, Melbourne, Australia.

Wiig, K. M. (1997). "Knowledge Management: Where Did It Come From and Where Will It Go?" Expert Systems With Applications **13**(1): 1-14.

Woods, E. (1998). Knowledge Management (Ovum Report), Ovum Ltd.

Yin, R. K. (2008). Case Study Research: Design and Methods, Sage Publications, Inc.