2007

A grounded theory of software process improvement model adoption

W. Grant Norman
West Virginia University

A Grounded Theory of Software Process Improvement Model Adoption

W. Grant Norman

Dissertation
Submitted to the College of Human Resources and Education
At West Virginia University
In partial fulfillment of the requirements for the degree of

Doctor of Education
in
Technology Education

Van Dempsey, Ph.D. Chairman
Dan Hartley, Ed.D.
Cheryl Prichard, Ed.D.
Neal Shambaugh, Ph.D.
Abhishek Srivastava, Ph.D.
Jaci Webb-Dempsey, Ph.D.

Department of Technology Education
Morgantown, West Virginia
2007

Keywords: Software process improvement, CMMI®, organizational change, grounded theory, software engineering, software developer

Abstract

A Grounded Theory of Software
Process Improvement Model Adoption

W. Grant Norman, MSSE

This study, using a grounded theory methodology, analyzed data collected from software developers and IT professionals on software process improvement (SPI) adoption. The study is presented within a backdrop of organizational change steps described by John P. Kotter in his 1996 book, Leading Change. Software quality problems and failures have caused many financial losses, injuries, and even deaths. In the mid 1980s, as a means of mitigating these problems, the Department of Defense (DoD) and Carnegie Mellon University (CMU) began work on the Capability Maturity Model (CMM®) In 2001, the model was superseded by a more robust model, the Capability Maturity Model Integration (CMMI®) These models were designed to provide descriptive, key process improvement areas for organizations to achieve greater maturity in their software and systems development. Organizations could then be appraised at specific maturity levels. According to CMU, SPI improves quality and reliability of software products. The DoD and several organizations now require companies to be appraised at a certain maturity level prior to being awarded a contract. From the onset, there have been difficulties in the adoption of these SPI models. Some of these difficulties can be attributed to organizational change issues. Through grounded theory analysis, a substantive theory was developed, The Theory of Software Process Improvement Model Adoption. This theory contributes to the body of knowledge by providing data and analysis from numerous IT professionals and software developers. This study also provides suggested key organizational change concerns for better SPI adoption practices.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would first like to thank all the participants who made this study possible. Without their willingness to share their experience there would have been no data to analyze. Next, I would like to thank the fine professors of West Virginia University for their help in this endeavor, especially the entire faculty of the College of Human Resources and Education, Technology Education department, the College of Business and Economics, and the Lane Department of Computer Science and Electrical Engineering. All of your efforts are appreciated and have contributed to my ability to perform this study. I would also like to thank all of my committee members, each have contributed so much in making this possible. Additional thanks go out to Dr. Van Dempsey, who stepped in as chairman upon the departure of my previous chairman. Your willingness to take over this toward the latter part of the endeavor, and even continue upon your departure from West Virginia University is again greatly appreciated. Thanks to my lovely wife, Anita Carter, for putting up with my many days in the dissertation dungeon, and supporting me during the time we missed being together. Lastly, and by no means least, my thanks to Dr. George Trapp, former chairman of the Lane Department of Computer Science who when I mentioned to him while I was still completing my Masters in Software Engineering that I was considering going on for a doctorate, in his wise and brief manner, stated, “Go for it!”

Chapter 1 – Introduction

General Background

> On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after lift-off. The rocket was on its first voyage, after a decade of development costing $7 billion. The destroyed rocket and its cargo were valued at $500 million. It turned out that the cause of the failure was a software error in the inertial reference system (Arnold, 1996).

Upon failure of the rocket's computer to understand its position, the rocket self-destructed and 10 years of work and 7 billion dollars of investment were gone. This example is not the only software caused disaster, but possibly one of the most significant in terms of cost. Worse than the Ariane disaster, sometimes bad software kills.

> The Therac-25, a computerized radiation therapy machine, massively overdosed patients at least six times between June 1985 and January 1987. Each overdose was several times the normal therapeutic dose and resulted in the patient's severe injury or even death. Overdoses, although they sometimes involved operator error, occurred primarily because of errors in the Therac-25's software. The manufacturer did not follow proper software engineering practices (Leveson & Turner, 1993).

The Therac-25 incident is one of the first recorded incidents where deaths occurred. Three people were killed. The following table is a listing of an additional seven software problems that contributed to the deaths of well over 400 people between 1991 and 2003.

Table 1

Some Deaths Related to Software Problems from 1991 - 2003

Year	Deaths	Description
2003	3	Software failure contributes to power outage across the Northeastern U.S. and Canada.
2001	5	Panamanian cancer patients die following overdoses of radiation, amounts of which were determined by faulty use of software.
2000	4	Crash of a Marine Corps Osprey tilt-rotor aircraft partially blamed on “software anomaly.”
1997	225	Radar that could have prevented Korean jet crash hobbled by software problem.
1997	1	Software-logic error causes infusion pump to deliver lethal dose of morphine sulfate. Gish Biomedical reprograms devices.
1995	159	American Airlines jet, descending into Cali, Colombia, crashes into a mountain. Jury holds maker of flight-management system 17% responsible. A report from Aeronautica Civil of the Republic of Colombia, digitized by the University of Bielefeld in Germany found that the software presented insufficient and conflicting information to the pilots, who got lost.

1991	28	Software problem prevents Patriot missile battery from picking up SCUD missile, which hits U.S. Army barracks in Saudi Arabia.

(Gage & McCormick, 2004).

Need for the Study

As illustrated above, death and destruction from poorly written software is a reality in today's world. As a means of resolving this problem, a software development process model, such as the Capability Maturity Model Integration, (CMMI®), from Carnegie Mellon University's Software Engineering Institute (SEI), provides a descriptive framework which helps eliminate poorly written software (Whitten, 1995, p.12). As Watts Humphrey, a founder of the SEI observed, "Some organizations have addressed the problem of developing large scale software systems by adopting the concept, taken from the manufacturing community, of a defined and managed process" (Humphrey, 1995, p. 4). Although there are important differences, the concepts from a manufacturing point of view are also well suited to software (Humphrey, 1989, p. 3). One must look at software as a manufacturing process and therefore, address all the typical production issues: customer requirements, design, inventory, manufacturing processes, quality control, shipping, training, support, and maintenance.

Software engineering is defined as an: "…engineering discipline which is concerned with all aspects of software production" (Sommerville, 2001, p. 6). Therefore, based on Whitten's, Humphrey's and Sommerville's observations, the software engineer, utilizing Software Process Improvement (SPI) frameworks, is a software *manufacturer* concerned with customer requirements, design, inventory, manufacturing

processes, quality control, shipping, training, support, and maintenance of software products.

CMMI® Defined

The Carnegie Mellon University SEI defines CMMI® as follows:

> A model is a simplified representation of the world. Capability Maturity Models (CMMs) contain the essential elements of effective processes for one or more bodies of knowledge. These elements are based on the concepts developed by Crosby, Deming, Juran, and Humphrey. Like other CMMs, Capability Maturity Model Integration (CMMI®) models provide guidance to use when developing processes. CMMI® models are not processes or process descriptions. The actual processes used in an organization depend on many factors, including application domain(s) and organization structure and size. In particular, the process areas of a CMMI® model typically do not map one to one with the processes used in your organization (CMMI Product Team, 2002, p.1).

Process Improvement and Organizational Change

This study is centered on the following premise: "In order to prevent or at least mitigate some of the problems of poorly written software, software engineers need to successfully adopt an SPI strategy." Research has shown that "higher levels of process maturity as assessed by the SEI's CMM are associated with significantly higher product quality" (Harter, Krishnan, & Slaughter, 2000, p. 464). Unfortunately, knowing a solution and successfully implementing it can be two entirely different issues. Implementation takes us into the world of organizational change. When SPI is implemented, changes

are being made to the organization, and in the beginning of a change effort, you do not have a complete sense of what is ultimately involved (Kotter, 1996, p.139).

As a means of identifying some of the organizational changes issues involved with SPI, this study will bring a greater understanding of the adoption issues. It will look at SPI through the lens of organizational change.

This study focuses specifically on the adoption concerns of software developers and IT professionals of the CMMI® SPI model. This approach will obtain an understanding of implementation phenomena as they relate specifically to SPI adoption and organizational change. Software developers and IT professionals know there is bad software written, and additionally, for the most part, they know there are SPI models that can benefit them in the development of better software.[1] Therefore, the purpose of this study is to identify the software developer and IT professional SPI adoption concerns and more clearly understand how this relates to organizational change issues.

Problem Statement

As shown in Table 1, the existence of poorly written software is well documented. Software Process Improvement is a strategy to eliminate poorly written software (Humphrey, 1995, p. 4). However, implementation and adoption of SPI at the software developer and IT professional level is frequently unsuccessful (West, 2004, pp. xviii – xxi). The problems of these SPI implementations, at the software developer and IT professional level, are not clearly understood. This study will bring more understanding to these problems and discuss some conclusions and recommendations.

[1] Verified by a pilot study of software developers by the author performed in November, 2004.

Research Questions

Through interviews, observations, and analysis of related literature, using an adapted, qualitative grounded theory approach, what will emerge as substantive theories in adopting Software Process Improvement strategies? As with any implementation of new or different work practices within an organization, what organizational change substantive theories, if any, will also emerge?

Interviews, observations, and literature analysis. With the grounded theory approach, the primary source of data collection is interview data obtained through both personal, face to face discussions and through email exchanges. Observational data is also included as part of the grounded theory approach, analyzed in the same manner. Observational data is obtained through the researcher's daily activities operating within a software development environment, and includes project notes, emails, meeting minutes, and other observations of software developers and IT professionals. For the purpose of this study, observational data is held strictly confidential and when directly referenced, the data will be cited as anonymous. Lastly, beyond the normal review of literature, grounded theory also utilizes existing literature as a part of the overall analysis of data collected by the researcher as part of looking for emergence to the research questions. These three areas of data collection make up the adapted, grounded theory approach utilized by this study and will be further discussed under the "Methodological Summary" section.

Expected Results

Software developers and information technology (IT) professionals serve on the frontline of the software product creation battle. Virtually every component of the

software product passes, at one time or another, through one of these IT professionals. Yet, there is little direct information on SPI implementation issues or organizational change issues reported from the software developer and IT professional points of view – information that can support improvement practices by bringing to light one of the most, if not the most, important views on SPI issues.

For example, if one were examining problems on an automotive assembly line, the workers on the line would be some of the first people to seek information from. By the token of their "hands-on" work experience; the assembly line worker most likely will have extremely relevant input. Similarly, in an SPI implementation effort or planning, the software developer and IT professional would have extremely important input. Additionally, implementation of new work practices, such as SPI, creates organizational change issues that need to be better understood from the software developer and IT professional point of view. This qualitative study allows for the emergence of that data by not constricting software developer responses to quantitative surveys or presupposed hypotheses, but by letting the data be revealed (Glaser, 1978) as it is obtained and analyzed through a qualitative grounded theory methodology.

Methodological Summary

This methodology will allow for a constant comparison of data from interviewing software engineers and IT professionals, observing them in their work environment, and reviewing literature as it relates to the data collected and observed. It is through the triangulation of this data that this qualitative analysis will gain trustworthiness.

This grounded theory approach will be primarily based on the responses obtained from software developers and IT professionals actively involved in the creation

of software projects. These responses will be specifically focused on their perspectives of the implementation and the adoption of SPI in their work. Their understanding and views of the need for (or lack of need for) process improvement will be captured and evaluated.

The constant comparative approach (Glaser, 1978) will allow for expansion and development of specific coding themes, which in turn, will lead to the continual development of additional data collection until such time the study reaches a saturation point. Saturation is the point where there are diminishing returns in the examination of new data within specific categories (Gasson, 2004, p. 84). The study will be built upon the results of this saturation and documented in a narrative presentation, with the addition of graphs and diagrams to help explain the conclusions obtained form these "real-life" interviews and observations. Grounded theory's combination of inductive and deductive components shall allow and the emergence of SPI implementation concepts for further study.

Chapter 2 – Literature Review

The Beginning of Software Process Models

In September, 1986, Frederick P. Brooks, Jr., author of *The Mythical Man-Month*, presented at the International Federation for Information Processing (IFIP) Tenth World Computing Conference an article entitled “No Silver Bullet” (Brooks, 1995, p. 179). The article discusses how a silver bullet, one single shot, would extinguish a werewolf. In this monster analogy, he states, “The familiar software project has something of this character…usually innocent and straightforward, but capable of becoming a monster of missed schedules, blown budgets, and flawed products” (pp. 180-181). Yet unlike the elimination of the werewolf, Brooks goes on to say, “There is no single development, in either technology or management technique, which by itself promises even one order of magnitude improvement in productivity, in reliability, in simplicity”(p. 180).

At this same time, the framers of the SEI Capability Maturity Model, the CMM®, had just started their initial work (Paulk & Garcia, 1994, p. 1), on the model described by Watts Humphrey (1989) in their own attempt to bring productivity, reliability, and simplicity to the development of software and software engineering projects. Where Brooks notes, “The essence of a software entity is the construct of interlocking concepts…I believe the hard part of building software to be the specification, design, and testing of this conceptual construct” (p. 182), the CMM® was addressing these issues by developing a look at best practices in both software and non-software organizations (Jalote, 1999, p. 5).

The SEI Software Process Program has focused on software process as a means of improving the ability of software organizations to produce software

> according to plan. This focus on software process is based on the premises that 1) the process of producing and evolving software products can be defined, managed, measured, and progressively improved and 2) the quality of the software is largely governed by the quality of the process used to create it (Humphrey, Kitson, & Gale, 1990).

In Pittsburgh, December, 1986, at the Software Engineering Institute, discussions were being held on some of the specific aspects of software engineering issues (Harvey, 1986). One of the major concerns, then, and as it is today, was the management of Software Configuration Management (SCM). As defined in the CMMI® some 15 years after this workshop: "The purpose of Configuration Management is to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits models" (Paulk, Curtis, Chrissis, & Weber 1993, p. 184). However, during the workshop the concerns for this process area of software engineering were quite obvious:

> No one at the meeting thinks that SCM is being used effectively as a management tool; in fact, just the opposite. Although there have been many corporations with solid SCM programs, many others produce software today with either no program whatsoever or programs that hinder rather than help. What is wrong with the SCM programs today (Harvey, 1986, p. 2)?

The answer to Harvey's question in 1986: there was little Software Configuration Management in place within the smaller contractors working for the large corporations or government. (p. 2). In conclusion, Harvey observes from the workshop output:

> A good configuration management team could make the difference between products coming in on time and within cost, and those coming in late, full of bugs, and with greater costs... I would conclude that what seems to be wrong in Software Configuration Management today is that too many software engineers don't seem to think they are missing much without a solid knowledge of SCM. If they can be shown the importance of SCM, then perhaps they will be more eager to learn its concepts and to use it more often and more effectively in the software development field today (p. 10).

Not only were there concerns over the quality of software configuration management, the 1980s was a time when many of the quality and management mavens were also marketing their wares. As Marash observed:

> Eventually, several TQM [Total Quality Management] gurus emerged, each with his own interpretation of TQM. During the 1980s, Juran, Deming, Philip Crosby, Armand Feigenbaum and others received widespread attention as philosophers of quality, and many large U.S. corporations introduced--some quite successfully--TQM concepts that led to a reemphasis on quality (Marash, 2001, p. 2).

The Japanese, and other competitors in the world, had learned to emphasize quality in their manufacturing during the same period when American manufacturing quality was declining. "The Japanese products not only have higher quality, but they also have lower prices" (Reid, 1990, p. 4). As things continued to decline, the messages from the quality people continued "…quality must be built in at the design stage" (Deming, 1986, p. 49) and "Poor quality costs your company money. Good quality saves your company money. It's as simple as that" (Harrington, 1991, p. 190).

Where examples during 1960s and 1970s, such as the Gemini and Apollo space missions, seemed to convince us that American Technology was the best in the world, the 1980s showed us, sometimes painfully and horrifically just how limited we were. The space shuttle Challenger showed us in 1986 that our nation's space program was still a viable target for both poor quality and poor management decisions. The management at the manufacturer of the o-ring that failed had been notified some eight months prior to the failure (Boisjoly, 1995). At this time, while the Software Engineering Institute (SEI) was just starting to formulate the CMM® and the space shuttle was destroyed, several other people were being subjected to the same kinds of technological *tragedies* by the Therac-25 radiation treatment machine. As observed by Leveson & Turner:

> The Therac-20, a predecessor of the Therac-25, employed independent protective circuits and mechanical interlocks to protect against overdose. The Therac-25 relied more heavily on software. Moreover, when the manufacturer started receiving accident reports, it, unable to reproduce the accidents, assumed hardware faults, implemented minor fixes, and then declared that the machine's safety had improved by several orders of magnitude. The design of the software was itself unsafe. The Therac-25 supported a multitasking environment, and the software allowed concurrent access to shared data. This precarious implementation caused program failure under certain conditions (Leveson & Turner, 1993).

The 1980s truly seemed to be a decade of poor quality in American technology. This, in turn, led to the decline of some long-standing American businesses. Reid sums it up in his discussion about Harley-Davidson:

> Harley-Davidson learned many other lessons – often the hard way – along the road from euphoria of independence in 1981 to the brink of bankruptcy in 1985 to market leadership today…to survive in today's competitive world, the U.S. manufacturers must make customer satisfaction their ultimate goal…" (Reid, 1990, p. 7).

The same lessons were, and many cases still are, needed to be learned by software development contractors and companies. There are still numerous cost over-runs and the likelihood of a software product being finished on time and within budget are still very, very slim. "Only 16.2% of projects will be completed on time, on budget" (Software Productivity Center, 1999). Humphrey, referencing the 1980's, also warned of more risk to the public from poor software as our society continues to become even more computerized (Humphrey, 1989, p.13).

Maturity, models, and CMMI®. Just what is meant by software maturity? Chuck Connell effectively uses a baseball analogy to describe software development maturity by comparing Little League players to professional players. Little leaguers run aimlessly around the field and may or may not make the correct play. Professional players practice and regularly work on improving their individual and teams skills to the level of where correct performance becomes more automatic (Connell, 2002).

Basically, the software development environment at Level 1 in CMMI® is represented by the Little League team. "So, there is nothing structured in Maturity Level 1, and being Level 1 is a bad thing" (Kulpa & Johnson, 2003, p. 32).

CMMI® is sometimes criticized for being too ambiguous. This ambiguity requires the adopters to do a lot of work interpreting the model serve their needs. (Kulpa & Johnson, 2003, p. 2). Where many think of the CMMI® as a cookbook for process improvement, some become disappointed when they discover the recipes are not complete. Instead of something such as "1 cup sugar, 3 cups flour," CMMI® would state "Define how *your* company would make a cake." For many organizations, this may be the first time they have had to actually evaluate their own software engineering processes (West, 2004, pp. 1-45).

This defining of processes within an organization is frequently difficult (pp. 1-45). To begin, unless management has made the commitment to drive and continue to support such process implementation and definition, it becomes almost an impossible task (Niazi, Wilson, & Zowghi, 2005, p. 157). As Humphrey observed, "This calls for a management team that takes care in making commitments and then insists on extraordinary efforts to meet them" (1989, pp. 70-71). Bob McFeeley, in discussing his IDEAL℠ model for implementing process improvement, noted: "Commitment and sponsorship are key. Without strong, informed, and steadfast commitment and sponsorship from senior management, the effort is doomed from the start" (1996, p. 13).

Humphrey, in 1989, referring to the need for process self-assessment stated "Management is often so focused on finding solutions that it fails to define the problems" (p. 35). The first step in defining these problems is to understand the current processes

within the software development environment. A popular method for capturing, evaluating, and testing this understanding is by use of the IDEAL[SM] model. This model is designed to work with not only CMMI®, but any other Software Process Improvement methodology (McFeeley, 1996, p. 7). It is made up of five steps listed below (p. 6):

Table 2

The IDEAL Model

Activity Name	Activity Purpose
1.0: The Initiating Phase	Learn about process improvement, commit initial resources, and build process infrastructure.
2.0: The Diagnosing Phase	Establish current levels of process maturity, process descriptions, metrics, etc. Initiate action plan development.
3.0: The Establishing Phase	Establish goals and priorities, complete action plan.
4.0: The Acting Phase	Research and develop solutions to process problems. Expand successful process improvements to entire organization.
5.0: The Leveraging Phase	Prepare for the next cycle through the IDEAL model. Apply the lessons Learned to refine the SPI process.

(McFeeley, 1996, p. 7).

The model is repeated multiple times until such time that the process improvement infrastructure is in place and goals have been achieved. (McFeeley, 2004).

While self-appraisal through a model such as IDEAL℠ is a good first step for any process improvement, for the long haul, implementation of a model requires both management and staff willingness to continue working through levels of process maturity (West, 2004, pp. 61-70). In the staged representation of the CMMI® model, those levels are: 1 - Initial, 2 – Managed, 3 – Defined, 4 - Quantitatively Managed, 5 – Optimizing (Wegerson & Williams, 2002, p. 1).

As previously mentioned, most organizations are at Level 1 if they have not started on any process improvement. To achieve Level 2, Managed, is a considerable amount of work. This includes adopting seven process areas and numerous specific practices, a dozen generic practices within multiple generic goals – all identified and documented with the specific organization's plans for achieving (CMMI Product Team, 2002). If achieved in 1½ years, an organization has made remarkable progress (the average is two years). Achieving Level 3 also takes an enormous amount of time, usually another two years (Software Engineering and Analysis Team, 2002, p. 25).

Issues in Software Process Improvement adoption. While many agree that Software Process Improvement is beneficial, the actual adoption of process improvement many times is not successful (Ngwenyama & Nielsen, 2003, p. 100). Many times social and organizational situations are not even considered and the process improvement models make assumptions about the organizational culture, reducing software development environments to little more than mechanistic input-output processes (p. 100). This view does not take into account that software process improvements are based in organizational change with complex organizational forces at work (Rainer & Hall, 2001, p. 176).

Additionally, many adopters may feel that there is just too much bureaucracy in the adoption of a Software Process Improvement model (Herbsled & Goldenson, 1996, p. 323). However, those surveyed reported software process models do provide a roadmap to help establish improvement strategies (p. 326). Yet, implementation issues were often evident due to lack of adequate time. Other major issues in creating a successful implementation were significant monitoring of progress by managers and involvement of technical staff. Involving software developers and giving them a clear vision of what needs to be accomplished is definitely critical to the success of the implementation (p. 328-329).

Organizational Change

The implementation of Software Process Improvement (SPI), by definition, implies a change in the organization (Fantina, 2005, p. 3). As previously discussed, in reference to the IDEAL^{SM} Model, the Management Steering Group (MSG) and the Software Engineering Process Group (SEPG) are chartered with discovering ways to improve or *change* the existing software development environment in order to obtain process improvement (McFeeley, 1996). While this change is specifically targeted towards SPI, it is still relevant to examine it against a backdrop of organizational change issues.

One of the first organizational change issues is personnel resistance to change. This resistance from software developers must be dealt with if process implementation will have any chance of success. As Piderit observed, “Successful organizational adaptation is increasingly reliant on generating employee support and enthusiasm for proposed changes, rather than merely overcoming resistance” (2000, p. 783).

Software developers must buy-in to the concept of process improvement. Simply stating that Software Process Improvement is being implemented will not produce the needed change. If it is going to succeed, it is important that the developers become an integral part of that implementation. There must be "internal process ownership" by the developers (Niazi, Wilson, & Zowghi, 2005, p. 157). Without a feeling of ownership, it is difficult for any employee to support change. As Kotter identified, "…employees generally won't help, or can't help, if they feel relatively powerless" (1996, p. 102). This inability to support, then, may manifest itself as resistance, when in fact, it is just a matter of not empowering the employee.

Motivation. Part of overcoming resistance is creating motivation to change. Employees (and managers alike) must have a "tangible dissatisfaction with the status quo and an eagerness for something measurably better" (Luecke, 2003, p. 19). There must be a sense of urgency. They need to clearly see status quo will lead to the eventual downfall of the organization and/or their specific position within the organization.

Kotter discusses this sense of urgency as the first of eight steps to transform an organization. This first step is critical, yet, over 50% of the companies he has watched failed at this phase. Senior executives frequently underestimate how difficult it is to move employees from their comfort zones (1995, p. 60).

Therefore, in attempting to implement software process improvement, it is necessary to make certain the employees are: (a) motivated to change, (b) have a sense of urgency, and (c) feel empowered to make such changes. One method of accomplishing this is to put people into a new organizational context which creates new

roles, relationships and responsibilities for the employee (Beer, Eisenstat, & Spector, 1990, p. 159). Employees must actively work to achieve specific goals of these new roles and responsibilities which lead to change. The focus is on the activity and not necessarily participation or culture (p. 159). Schaffer and Thomson believe companies waste large sums of money and time on activity based programs rather than focusing on results (1992, p. 81).

Additionally, it could be argued that by measuring results, the motivation of software developers, could be substantiated and not just an example of the purported Hawthorne effect[2], as observed by Herzberg in 1968, "…attitudes toward the job changed artificially merely because employees sensed the company was paying more attention to them" (Herzberg, 2003, p. 87). Motivation and attitudes hadn't really changed; it is just a reaction to all the attention suddenly by management that made it appear to change (Franke & Kaul, 1978).

The motivation of software developers may be best measured not by activity and training, or by social interactions, but by the focusing on the results of their work. Results driven methodology is management taking action when it appears to directly lead to improvement (Schaffer & Thomson, 1992, p. 82). This, in turn, can lead to successful change; when software developers see the results of their improved work, it also increases their motivation to move from the status quo.

Building a coalition. A second area in achieving successful implementation of change and SPI is the building of a coalition of individuals with enough power to guide the change. As Kotter points out in his second step in avoiding errors, "...groups without

[2] Individual behaviors may be altered because they know they are being studied was demonstrated in a research project (1927 - 1932) of the Hawthorne Plant of the Western Electric Company in Cicero, Illinois.

strong line leadership never achieve the power that is required" (1995, p. 62). How many people are needed for a coalition? Luecke suggests that in order to affect organizational change, 75% of the managers must be on board with the idea of change (2003, p. 34). When the managers are not part of the change effort coalition, then they are usually part of the problem: "The project managers did not see that their attitudes and perceptions were major obstacle[s] sic for implementing SPI" (Nielsen & Kautz, 2004, p. 11). Additionally, there is a fear associated with making changes from the status quo. "Breaking from the status quo means taking action, and when we take action, we take responsibility thus opening ourselves to criticism" (Hammond, Keeney & Raiffa, 1998, p. 50).

Successful coalitions can be built when there is a high sense of urgency within the managerial ranks (Kotter, 1995, p. 62). Yet, the implementers of the changes will usually be the employees, so managerial coalition will only drive change to a certain point. If SPI is to succeed, the management must also gain buy-in from the software developers. As observed in one case study of process implementation, "Yet, through it all, most of the … staff did appear to believe … [the] motivations were to simply make things better" (Norman, 2003, p. 35). There was a sense that although there was a huge task to accomplish, there was a *team* moving in the direction of accomplishment (p. 35). This sense of a coalition helped provide the implementation guidance that was needed to succeed (Kotter, 1995, p. 63).

Vision. Successful organizations have clear vision that is agreed upon and shared by others within the organization. There is total commitment to the vision and all have a crystal-clear understanding of the goals (Lynn, 1998, p. 90). This represents the

concept of vision at its best. Additionally, unless we can see change as an ongoing process and a “stream of interactions…as opposed to episodic events” eventual implementation problems will be difficult to overcome (Tsoukas & Chia, 2002, pp. 568-569). This is all part of creating a meaningful and accurate vision.

Not having vision can quickly derail any change or process improvement strategy. With no clear vision, any project or implementation can drift along, costing millions of dollars and thousands of wasted man hours. A recent example of lack of vision was reported on CNN in February of 2005. Over $170 million has been spent on a project the FBI calls Virtual Case File. According to the report:

> The FBI had recently admitted the Virtual Case File technology, which had been delivered by contractor Science Applications International Corp. (SAIC), had failed to meet the bureau's requirements and that much of the time and effort invested had been lost. Mueller said the FBI and the contractor shared the blame (Frieden, 2005).

When project requirements are not understood or properly conveyed, it is usually because of a lack of vision and scope for the project (Wiegers, 1999, p. 96). The same is true for implementing change within an organization. If there is not a clear path to where employees are headed, it is difficult for them to find their way (Beer, Eisenstat, & Spector, 1990, p. 161-162; Kotter, 1995, p.63). This can frequently happen because management is only aware of part of the problem (Luecke, 2003, p. 34) and has only a partial vision. “A project that lacks a clearly established and well-communicated direction is an invitation to disaster” (Wiegers, 1999, p. 96). You should be able to communicate your vision and get a reaction to it in five minutes or less (Kotter, 1995, p.

63). Then once it is communicated, it needs to be communicated repeatedly by no less than a factor of ten (p. 63).

Communication and diffusion. Many times, visions are created but never adopted. Even visions with intrinsic merit sometimes fail to spread (Senge, 1994, p. 227). The best vision, the most excellent project or plan, all will be for naught if it isn't diffused amongst those responsible for implementing the change. As Rogers noted, "Diffusion is the process in which an innovation [change] is communicated through certain channels over time among the members of a social system. It is a special type of communication, in that the messages are concerned with new ideas" (Rogers, 2003, p. 5).

In implementing process improvement strategies or change within an organization, the concept of *diffusion* may be a more appropriate word in that it connotes "social change" (p. 6). On the contrary, *communication* implies a broader sense of information dispersion. An advertisement, a book, a television show, even a fortune cookie message, all are types of communication that expect the receiver to prepare themselves to make an effort to understand. The concept of diffusion, a social change, implies that the communications are readily received, implemented, or acted upon. Visions spread when they are reinforced with communication and commitment by the communicators (Senge, 1994, p. 227). Within an organization, implementation frequently involves a number of individuals to champion (or oppose) as a means of communicating (Rogers, 2003, p. 403). Communication is not easily spread by a single individual.

Kotter observes that frequently, even after a relatively good vision is developed, the communication of the message to the organization is very weak – frequently representing as little as .0001% of the organization's total communications (1995, p. 63). This error of under-communicating makes transformation next to impossible. The troop's hearts and enthusiasm are captured by repeated communication (p. 63). Yet, managers frequently wonder why employees aren't motivated to adopt the new vision.

In more successful organizations, the executive management rarely misses a communication opportunity to spread the word (p. 64). It is by this broad, multi-channel communication methodology that makes for effective diffusion of the vision. Harrington noted "…drastic change requires clear and direct top management communication to all employees" (Harrington, 1991, p. 43).

Saunders, in her article in the *Harvard Management Communication Letter* (1999, pp. 1-3), discusses a dozen tips for communicating change within an organization. These suggestions on communication are an alternative to organizations' typical "Big Bang" announcements, which usually are destined to fail. Some of these suggestions include making the area of change clear, openness with the employees even if it is bad news, and repeating the communications in various media and formats (pp. 1 -3).

As noted above, communication of change within an organization, such as software process improvement requires extensive communication efforts. It is like a major advertising campaign. As Luecke declares, "Communicate relentlessly" (2003, p. 60).

Empowering people to act. A critical mass for change emerges when people believe they truly have a voice (Axelrod, 2001, p. 3). All the excellent communication of a divinely inspired vision or Software Process Improvement will not have one bit of impact if the individual employees are not empowered to act. Sometimes employees understand the new vision but something seems to be blocking their path (Kotter, 1995, p. 64). The obstacle may be real or imagined, but either way it must be removed.

No organization has the time or power to get rid of all the obstacles. If it is people causing the blockage, they must be dealt with fairly and a way consistent with the new vision. However, action is necessary if one is to maintain credibility of the change effort and help those that are obstructed to achieve (p. 65). Additionally, if there is fear – either real or imagined – in those needing to be empowered it is necessary to deal with it before any progress can be made. "No one can put in his best performance unless he feels secure…*Secure* means without fear, not afraid to express ideas, not afraid to ask questions" (Deming, 1986, p. 59).

To create a positive and empowered atmosphere, Luecke suggests several tactics such as encouraging innovative thinking, demonstrating respect for employees, eliminating micromanaging, encourage risk taking and being tolerant of failures to name a few (2003, p. 28).

Process improvement requires that business processes have ownership. The first criterion is ownership (Harrington, 1991, p. 46). However, without the power to act, ownership of a process means little. The improvement team must have sufficient power to act on selected change areas (p. 46). Otherwise, as some employees have been known to lament, "I have all responsibility and none of the authority" As Axelrod

observed, “For people to take initiative, they must have access to information and a stake in the decision process” (2001, p. 3). When people feel a lack of ownership, do not feel as if they are a stakeholder, or simply ignored when they speak up, like these obstacles they face, they too will become obstacles to change.

Short term goals. As mentioned earlier, the time it takes for an organization to gain CMMI® Level 2, Managed, is approximately two years (Software Engineering and Analysis Team, 2002, p. 25). Any type of process improvement or change is going to take some time. Unfortunately, years of mediocre performance cannot be undone instantly. Harrington remarks in reference to the time it takes for business process improvement, “This time commitment could last for a few months or a few years depending on the pace of the improvement and the extent of the change required” (1991, p. 51). In short, it will take time.

People like to see results of their efforts. While a process improvement strategy may ultimately take several years to be completely realized, both individual employees and management need to see progress.

In contrast to lengthy activity centered programs, focusing on results driven improvements can provide rapid measurable achievements (Schaffer & Thomson, 1992, p. 85). Schaffer and Thomson additionally suggest that selecting one or two quickly achievable goals that are important to internal customers can satisfy short term goals with real results (p. 85). “There is no motivator more powerful than frequent successes” (p. 86).

Managers need to look for ways to achieve clear and measurable performance improvements (Kotter, 1995, p. 65). Clearly planning for these short-term victories is

significantly different than hoping they will occur (p. 65). Senior managers frequently understand the necessity of the need for change, but misunderstand what it takes to accomplish those changes (Beer, Eisenstat, & Spector, 1990, p. 158). As with any large endeavor, it takes several small steps to achieve the overall goal. Kotter observes, "Commitments to produce short-term wins help keep the urgency level up and for detailed analytical thinking that can clarify or revise visions (1995, p. 66).

These short term wins and gradual metamorphosis are absolutely essential to achieving an enormous change effort. They represent the momentum and foundation that help people with the day-to-day steps that are needed in order to achieve the desired end results. However, Kotter warns that sometimes the momentum can be lost, and in fact, the entire change effort can be destroyed by declaring victory too soon (1995, p. 66). As Kotter experienced, "Until changes sink deeply into a company's culture, a process that can take five to ten years, new approaches are fragile and subject to regression" (1996, p. 66). Quick, immediate wins are only the path to the organizational change and not the victory in and of themselves.

Institutionalization. Change stays when it becomes the way people do things in an organization (Kotter, 1995, p.67). CMMI® states, "…institutionalization implies that the process is ingrained in the way work is performed (CMMI Product Team, 2002, p. 33). There must be management support and organizational commitment for change to succeed (Ittner & Larcker, 1997, p. 523). Management needs to develop an organizational culture to support the long term process management (p. 523). Processes reinforced by a culture of openness and commitment become and integral part of the continuous work practices (Nielsen & Kautz, 2004, p. 20). Optimized and

mature organizational processes and control increase the maturity of an organization (Ravichandran & Rai, 2000, p. 390). Change must "seep into the bloodstream of the corporate body" (Kotter, 1995, p. 67).

As with any human habit, it takes persistence, hard work, and time to change it. Those that have embarked upon a weight loss diet, and for many this means several diets, know how easy it is to backslide. Organizational change and transformation are no different. It takes persistence, with several short-term, results oriented wins; it takes hard work by management and the employees in which management fosters an environment of trust and encouragement; and, lastly, it takes time for any human effort to become an institutionalized success.

Summary

The successful implementation of CMMI® is a significant change to an organization. Software developers and IT professionals are at the forefront of this change and understanding their perceptions on Software Process Improvement is critical to a successful implementation. Again, as mentioned earlier, it is not something that can just happen over night. Some organizations have argued that all they need to do is write up all of their processes, make certain they fit the model, and then self-assess and instantly call themselves a CMMI® organization. It is nowhere near that easy of a task

In this review of the literature, change does not just *happen.* It is an extremely difficult and complex issue that must be dealt with at a very high level within an organization. CMMI® will frequently take at least 2 years for each of the first two, Level 2 and Level 3, implementations. Levels 4 and 5, for those organizations which want to

achieve the highest certification, could easily be at least a 5 – 6 year process. Just on the basis of needed manpower and cost, the implementation of change at this level is an extremely costly initiative (Harter, Krishnan, & Slaughter, 2000, p. 464). Therefore, it simply does not make sense to take any of this lightly or rely on an ad hoc implementation – unless, that is, the company wishes to remain a Level 1 organization.

Chapter 3 – Research Design

Research Design

This study will construct substantive theories regarding Software Process Improvement adoption as discovered with software developers and other information technology professionals. An adapted, qualitative grounded theory approach will serve as a means of achieving the research (Glaser, 1996). This design is based on qualitative grounded theory from multiple sources, including; (a) the originators of grounded theory, Barney Glaser and Anselm Strauss; (b) information systems grounded theory researchers, Susan Gasson and Cathy Urquhart; (c) nursing grounded theory researcher, friend, and West Virginia University Assistant Professor, Alvita Nathanial; and, (d) other grounded theory and qualitative researchers and writers as referenced throughout the study. However, wherever possible, this application of grounded theory will most closely follow the original Glaser and Strauss (1967) model as described and updated in Barney Glaser's 1978 work, *Theoretical Sensitivity.*

One of the key factors of a grounded theory approach is that it attempts to discover complete theoretical explanations to particular phenomena (Nathanial, 2003, p. 42; Streubert & Carpenter 1999,, p. 100). The focus of this study is the examination of the phenomena of Software Process Improvement (SPI) adoption by software developers and information technology professionals. This study will use the backdrop of CMMI® as the SPI model and much of John P. Kotter's numerous works (1995, 1996, 2002) on organizational change for behavior discussions.

As Kotter observed, adoption of a new practice or process within a work area presents change to the individual's work reality (2002, p. 2). In attempt to understand

how software process adoption is perceived in this new reality, grounded theory provides the structure for constant comparative analysis of data collected directly from those people working in professional information technology environments. Through interviews, observations, and relevant literature, grounded theory allows the researcher to build theories through analyzing the data, data that is "grounded" at the phenomenon level. . As Chenitz and Swanson (1986) observed, "The purpose of a grounded theory study is to understand the concerns, actions, and behaviors of a group and explain those patterns of behavior at a higher level of abstraction" (p. 79). Glaser (1978) stressed, "The goal of grounded theory is to generate a theory that accounts for a pattern of behavior which is relevant and problematic for those involved" (p. 93). In this study, the behavior to be observed is specifically the information technology professional's response to Software Process Improvement and their perceptions of the changes this presents to the work environment.

Lincoln and Guba (1985), in discussing quantitative research versus qualitative, remarked that qualitative research is more adaptable in dealing with multiple realities and more sensitive to the influence and value patterns that may be observed (p. 40). Since this study will be centered on the discovery of emergent theory in Software Process Improvement adoption, which ultimately is behavioral in nature, a qualitative grounded theory approach is best suited. The methodology accounts for both the behavioral aspects and generation of theory as part of the grounded theory analytical process.

Participants

A purposeful sample of software developers and information technology professionals was utilized to gather the data for this study. They were selected using the following criteria:

- Currently employed within a software development, information technology, or other related data processing project
- Educated in software development through a university program such as computer science, computer engineering, information systems, software engineering, or other information technology program or self-educated in software development, computer programming, systems analysis, or other information technology skill set
- General familiarity with Software Process Improvement concepts, models, or implementation

Additionally, as a means of providing at least some heterogeneity, all participants met the following criteria:

- A minimum of at least six years, mid to senior level, related work experience in software development, computer programming, systems analysis, or other information technology professional skill set
- Viewed their primary career area as information technology
- Worked on at least one or more projects staffed with multiple software developers or IT professionals in which either processes were or were not followed

These participants were all males, ranged in age from 24 to 52, and were identified through professional contacts and work activities involving the development of software or management of software development projects. All participants were strictly volunteers and their identities and employers are strictly confidential. The participants worked within a variety of organizations including large information technology contracting, university IT instruction, and small IT software development companies and a wide variety of projects in areas of law enforcement, health sciences, and academia, network and Internet applications.

The decision to choose this group of participants was made to reflect two groups of IT professionals and software developers: 1) those experienced with process models and successful implementation of those models on projects they worked on, and 2) those IT professionals and software developers that were acquainted with the concepts of Software Process Improvement models but had no direct involvement with a successful implementation or were new to the process model adoption environment.

Researcher Point of View

As the principal researcher in this study, my background includes both self-learning of software and programming skills and university education in a Masters of Science in Software Engineering programming completed in August of 2003.

I began software development in the early 1980's approximately six years after obtaining a Bachelor of Arts degree in Humanities and English Literature. My first areas of software development were based on automated word processing systems until the first generation of IBM personal computers was released in 1983. Upon obtaining an original IBM PC, I immediately began programming in database software, developing

numerous sales and marketing programs to help with my work as a computer peripheral salesman in Houston, Texas. The next 20 plus years I worked as a software developer in a wide variety of manufacturing and accounting organizations, basing my development on thorough evaluation of customer needs and requirements.

As a lead software developer and project manager, I soon discovered the limitations of the typical “code and fix” methodology, whereby software is developed by writing a program – showing it to a user, then re-writing in a continual loop until either the customer is satisfied, gives up, or is out of money.

I am now a program manager for a health science organization managing nine software developers and other contracted computer support personnel for Center for Disease Control’s (CDC) National Institute for Occupational Safety and Health (NIOSH) in Morgantown, West Virginia. I also am a member of the company’s Software Engineering Process Group (SEPG) and I am actively involved with supporting the continued implementation of CMMI® within the company as they attempt to move from a CMMI® Level 2 to a Level 3 over the next two years. This has given me the unique perspective, both from a basis of university study and practical work experience in the areas of Software Process Improvement and dealing with the organizational change of such improvement strategies.

SPI, in my experience as I enter into this study, is *not* a panacea for development issues. Additionally, implementation in an organization, even with management support, is an extremely difficult and frequently failed endeavor. SPI is very costly, it is difficult to maintain management support and investment, customers do not like the

extra expense and time to develop their products, and for the most part, many software developers see it as just added work to an already hectic schedule.

With these biases and point of view, I will approach this study fully knowing that much of the work will be difficult. However, as observed in the introduction and repeated throughout other portions of this study, if we are going to continue to give more and more power to software in our daily lives, it is critical that we learn how to improve the quality of the product before more lives are lost and billions of additional dollars are wasted.

As a means of managing my own biases and assumptions I bring to this study, as part of my grounded theory research work I will be frequently quoting statements from the interviewees and my theoretical memos. Additionally, data from my participant observations will be clearly noted. As grounded theory encourages review of the literature as part of the research, I will utilize other's work in supporting the presentation of the collected data. By using frequent words of the participants, supporting literature of other process implementation research, and clearly identifying my participant observations when used, I hope to bring greater strength and trustworthiness to this study.

Study Limitations

Limitations of this study are the same as any grounded theory study – time to complete a fully saturated study that produces a *formal* theory (Gasson, 2004, p. 84). Instead, the goal is to produce substantive theories based primarily on the collected interview data. Over the course of years, enough studies may be conducted to ultimately produce a formal theory (p. 85).

Admittedly, this study is also limited by the number of participants and the specific selected purposeful sample (Patton, 2002, p. 563). All participants were personally approached by the researcher and asked to be interviewed. This limited the sample to only those software developers and IT professionals that the researcher personally knew, and thereby immediately excluded all other software developers and IT professionals.

In a pure Glaserian grounded theory approach (Glaser, 1978), interviews are conducted with more open ended questioning, with each question leading to additional questions. The researcher does not record the interview and makes notes and memos based on the interview. For the purposes of this study under the dissertation structure, an interview protocol of specific questions was created and utilized in all the interviews and the interviews were recorded or conducted via email. Although these procedures may not have limited the trustworthiness of the study, it did limit the evolution of the study as a pure Glaserian grounded theory study. A pure grounded theory research methodology does not contain the highly specific research questions normally associated with a dissertation format, but rather broader questions that are designed to elicit emergence and are frequently revised as research proceeds through immediate data analysis (Glaser, 1978).

Grounded theory is a means of inductive analysis, which sometimes is treated as suspect by the scientific community in that it introduces subjectivity into the research (Glaser, 1978, p. 85). However, acknowledging this and an understanding of the researcher's bias, the limitations of participants interviewed, and acknowledgement of the processes and variations of this grounded theory approach in a dissertation format,

should bring greater trustworthiness to the study. In the examination of the adoption of an SPI model, this study revealed many theories for future research. It is the development of these substantive theories, knowledge of future questions to ask, and the experience gained along this journey that will be the ultimate value of this research.

Chapter 4 – Methodology

Qualitative Grounded Theory Methodology

Grounded theory "contributions to knowledge are not generated from existing theory, but are *grounded* in the data collected from one or more empirical studies" (Gasson, 2004, p. 80). This is one of the unique features of this research methodology (Gasson, 2004; Glaser, 1978, 1992; Strauss & Corbin, 1998). Specifically in the area of software process improvement adoption, the methodology works very well since there has been limited understanding and little or no previous studies (Schreiber & Stern, 2001, p. 57). This limited understanding in the adoption of software process models is the primary choice for the use of the grounded theory methodology.

Data collection. This research study gathered data from three areas: (a) interviews of information technology professionals (b) observations of information technology professionals in the work environment and (c) SPI and qualitative information systems research literature (Gasson, 2004, p. 81; Streubert & Carpenter, 1999, p. 101). The primary data source has been taken from the interviews, with observations and literature used as a means of triangulation of the study (Glaser, 1996; Patton, 2002). The interview data was captured through the use of the initial interview protocol as listed in Appendix A. This protocol was designed around 12 questions of which only questions five through nine were used for raw data analysis. Additionally, a card with the seven CMMI® process areas was shown to the participants so they would be clear on the definitions of each area.

As data was collected and analyzed, utilizing a constant comparison process (Gasson 2004, p. 80; Schreiber & Stern, 2001, p.57), decisions on additional interviews or data collection strategies were determined (Glaser, 1978). This is one of the unique features of grounded theory in that data is analyzed as it is collected and additional data collection is based on this analysis until it reaches a point of saturation, a point where little new information is garnered. (Gasson, 2004, pp. 80-84; Glaser, 1992, p. 43; Urquhart, 2001, p. 107).

The following six steps were modified and adapted, in conjunction with Glaser (1978), for the development of the grounded theories within this study using the work of Harry, Sturges and Klingner, (2005, p. 6). The steps are presented bottom up – from Step 1 Open Coding up to Step 6 Theory.

Table 3

Grounded Theory Approach for this Study

6. Theory	Creation of substantive theory based on thematic interrelations
5. Interrelating the explanations	Combining of thematic data into explanations of the phenomena under study
4. Testing the themes (interviews, observations, literature)	Several layers of analysis of data relevance between themes from interviews, observations, and literature – triangulation of data
3. Themes	Refining categories into predominant core categories and themes
2. Categories	Classify data into initial categories working toward themes and core categories
1. Open coding	Analysis of interview data and memo creation

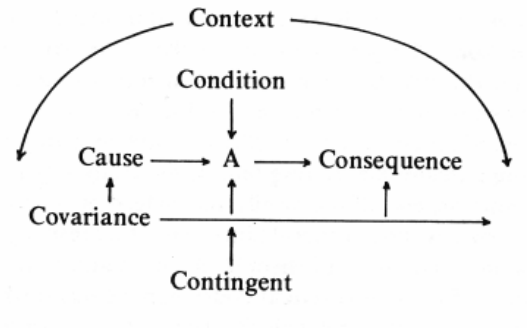
Open coding. Open coding in this study is the classification of data and themes by looking for patterns and categories (Gasson, 2004, p. 82; Glaser, 1978, p. 57). Data analysis proceeds in a manner incorporating procedures described by several grounded theory researchers (Gasson, 2004; Glaser 1978; Glaser & Strauss, 1967; Schreiber & Stern, 2001; Urquhart, 2001). Some of the questions the researcher asks during this process are: (a) what is this data a study of (b) what category does this incident indicate, and (c) what is actually happening in the data (Glaser, 1978, p. 57)? This is performed prior to the research of any related literature to the specific topic (Glaser & Strauss, 1967; Glaser 1978). However, as discussed previously, literature and observations will be incorporated as part of the grounded theory analysis as a means of triangulation. This is further discussed under the section "Testing the Themes."

It is best to utilize the bulk of the literature evaluation after the substantive theory is developed (Urquhart, 2001, p.121). The literature is then treated as another portion of the collected data and analyzed through the same process.

Categories. Categories are the step of identifying relationships between the open code identifiers created using open coding (Gasson, 2004, p. 83; Urquhart, 2001, p. 115). This is where the relationships between categories begin to emerge into themes (Urquhart, 2001, p.115). We need to be sensitive to the emergence of these insights into the relationships between categories (Gasson, 2004, p. 83). In looking for emergence insights in categories, Glaser (1978, p. 74) suggests the use of what he calls the "Six C's" coding family as shown in Figure 1:

Figure 1

Glaser's Six C's



For this study, only the cause and consequence within context are considered. Using the constant comparison methods (Glaser, 1978), causes and subsequent consequences can be differentiated and understood. It is important in the examination of data from this study to have a clear understanding of the context around these causes and consequences in that they can sometimes be easily mixed up (Glaser, 1978, p.74). As part of preventing his problem, as categories are created, Glaser suggest the use of what he calls "Temporal ordering" (Glaser, 1978, p. 78) in conjunction with the "Six C's "coding family. Careful consideration of the temporal ordering is made.

Theoretical memos are those thoughts and ideas that analyst discovers as they are working on the coding of data (Gasson, 2004, p. 83; Glaser, 1978, p. 83). It is a means of taking notes on the work as it is being performed which "lead naturally to abstraction and ideation" (Glaser, 1978, p. 83). As the work proceeds and the researcher has flashes of ideas or insight, they must stop and memo – if the researcher skips this step, then the work is not truly grounded theory (p. 83).

Themes. Themes provide a method of further refining categories (Gasson, 2004, p. 83). "It is important to explicitly state the research analysis objectives before and during coding" (p.83). Glaser refers to this as the creation of "core categories" - those categories that eventually lead to development of theory (Glaser, 1992, p. 75), the ultimate goal of grounded theory.

Testing the themes. Themes or core categories emerge from the analysis of the interview data. A means of adding rigor to the study and increasing trustworthiness, data from literature and observations are also analyzed as part of the constant comparison process. Their relevance to the analysis is compared to the thematic "grounded" data (Harry, Sturges & Klingner, 2005). Not only does this help with triangulation of the data, it provides an additional check for the researcher's interpretations of the "grounded" data.

Interrelating the explanations. "With our analysis firmly grounded in extensive, triangulated data, we refer to the themes as *explanations* – emphasizing the power of our analysis to develop a theory" (Harry, Sturges & Klingner, 2005, p. 9). The creation of substantive theory is the goal of this research. This final interrelating of the data leads to that goal. The creation of formal theory is much broader and extensive use of grounded theory (Glaser & Strauss, 1967). For the purpose of this study, substantive theories are adequate.

Repeat all steps as needed. Research iteration and constant comparison describes the iterative and constant cycling of the grounded theory process (Gasson, 2004, p. 84). As data is collected and analyzed, the researcher makes decision at that immediate time as to what other interviews or data collection is needed. Glaser refers

to this as a series of “double back steps” (Glaser, 1978, p. 16). “As one moves forward, one constantly goes back to previous steps” (Glaser, 1978, p.16). Unlike most research studies where data is first all collected then analysis proceeds, grounded theory is built on the constant comparative methodology which means data collection and analysis are cycled through repeatedly until such time that theoretical saturation is achieved (Gasson, 2004, p. 84). “Theoretical saturation is reached when diminishing returns from each new analysis mean that no new themes, categories, or relationships are emerging and new data findings confirm findings from previous data” (p. 84).

Specialized Methodology Tools

Pure Glaserian grounded theory involves the researcher interviewing participants with nothing more than possibly a pen and a note pad. Not even a tape recording of the interviews (Glaser, 1978). The world has changed significantly since the creation of grounded theory and today’s researchers are familiar with the use of many specialized office tools such as tape recorders, MP3 recorders, computers, and various word processing and research analytical software products.

Consequently, in breaking from the true Glaserian grounded theory, one of the first resources employed in this research has been the use of automated equipment. All interviews were either recorded and transcribed or received via email file attachments from the participants. While there exists several qualitative research computer programs, after examining a few, the researcher decided to write specific database programs and reports to assist in the analysis.

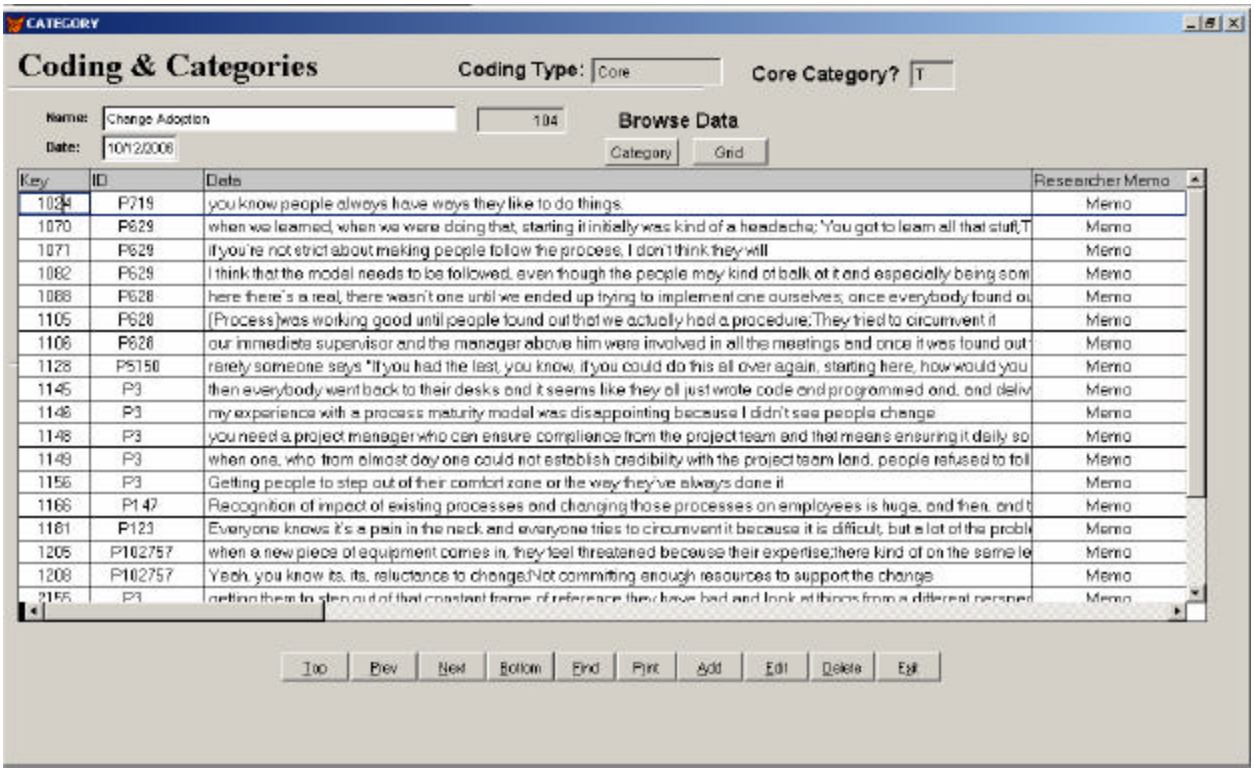
Some of this data analysis has been attached as Appendix B. The data was captured into a Visual FoxPro database, through various data tables. Interview data

was segmented down to one or two key thoughts – either a complete sentence or pieces of multiple sentences. In all cases, the interview data represents the raw data from the participant.

Each piece of data is numbered for easy reference by this element number in the discussion of the study results. Also, for making use of constant comparison component of grounded theory, manual queries of the data to display it in different groupings was utilized as a means of helping to generate themes and core categories. One of the primary queries places all participant data on the same screen with an open coding or core category. Each of these individual pieces of participant data was then examined and a memo was written in response. The memo served as the first step in generating theory and allowed for future constant comparison of the memo data as recommended by Glaser (1978, p.16) A sample of one of these data screens is shown below:

Figure 2

Sample Data Analysis Open and Core Category Coding Screen

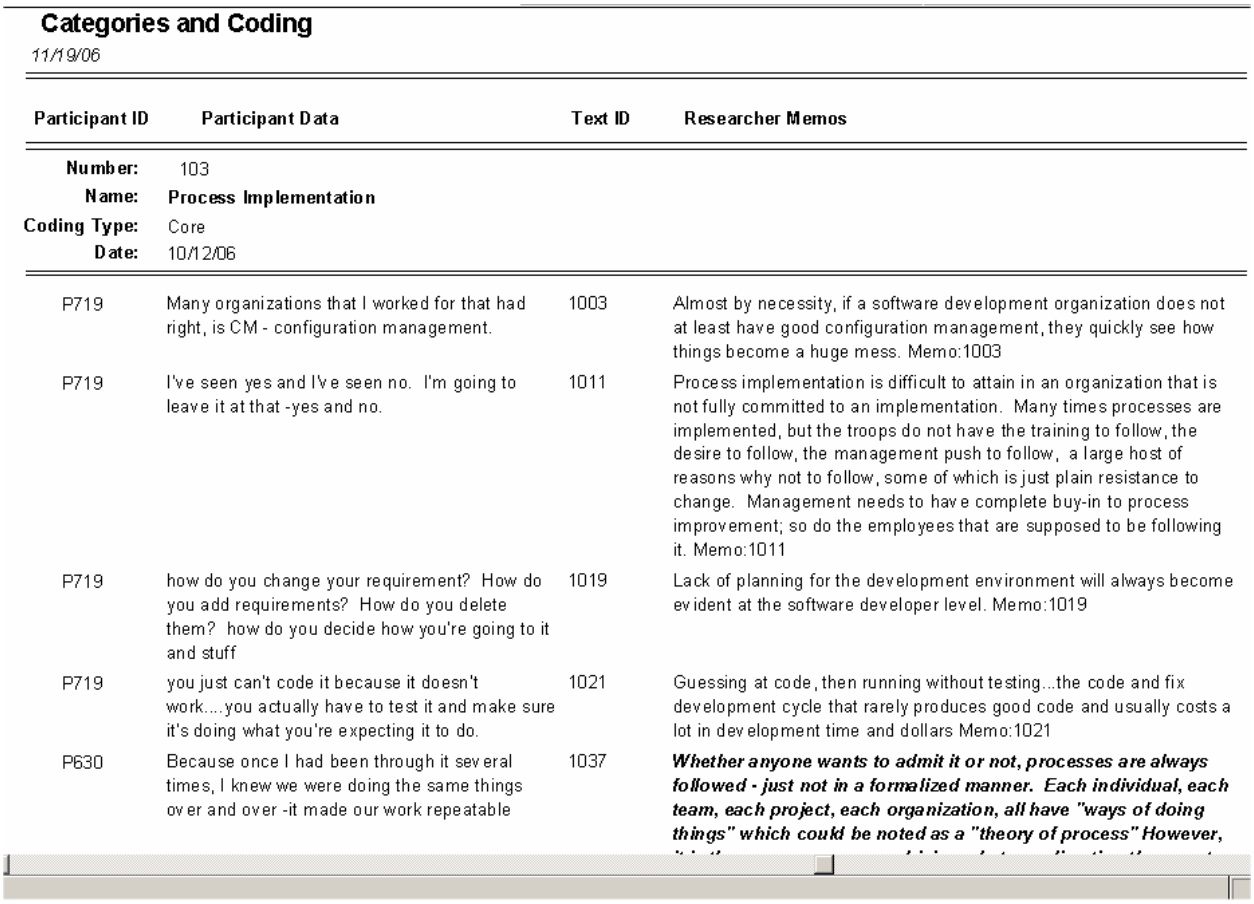


This use of the computer database proved to be an extremely useful tool in the analysis of the data, especially in the area of continuous memoing while reviewing and analyzing the data. As can be seen in the Figure 2 image, each data item has a corresponding number and a corresponding researcher memo with the same number. While not shown in the image, by simply clicking on the word “Memo”, a corresponding memo appears concerning the specific data statement.

For additional ease of viewing the data, a specific report was written to generate both the participant and researcher data side by side. An example of that report appears below and the full report is in Appendix B:

Figure 3

Sample Report Utilized for Constant Comparison of Participant and Memo Data within Categories and Open Codes



Categories and Coding

11/19/06

Participant ID	Participant Data	Text ID	Researcher Memos
Number:	103		
Name:	**Process Implementation**		
Coding Type:	Core		
Date:	10/12/06		
P719	Many organizations that I worked for that had right, is CM - configuration management.	1003	Almost by necessity, if a software development organization does not at least have good configuration management, they quickly see how things become a huge mess. Memo:1003
P719	I've seen yes and I've seen no. I'm going to leave it at that -yes and no.	1011	Process implementation is difficult to attain in an organization that is not fully committed to an implementation. Many times processes are implemented, but the troops do not have the training to follow, the desire to follow, the management push to follow, a large host of reasons why not to follow, some of which is just plain resistance to change. Management needs to have complete buy-in to process improvement; so do the employees that are supposed to be following it. Memo:1011
P719	how do you change your requirement? How do you add requirements? How do you delete them? how do you decide how you're going to it and stuff	1019	Lack of planning for the development environment will always become evident at the software developer level. Memo:1019
P719	you just can't code it because it doesn't work....you actually have to test it and make sure it's doing what you're expecting it to do.	1021	Guessing at code, then running without testing...the code and fix development cycle that rarely produces good code and usually costs a lot in development time and dollars Memo:1021
P630	Because once I had been through it several times, I knew we were doing the same things over and over -it made our work repeatable	1037	***Whether anyone wants to admit it or not, processes are always followed - just not in a formalized manner. Each individual, each team, each project, each organization, all have "ways of doing things" which could be noted as a "theory of process" However,***

By constant comparison of data by review, queries, and analysis, core categories and themes emerged allowing for testing of data with observations and literature and interrelating for the eventual generation of substantive theory.

Chapter 5 – Results

Course of Study

The study was conducted using interview and observational data collected between November, 2004 and October, 2006. Interviews used questions listed in Appendix A. The interviews were conducted in two fashions: (a) face to face meetings with the individual at a non-work related location, and (b) through email correspondence of the interview questions in a Microsoft Word document which was completed and returned via email by the participant. The majority of the interviews, 12, were conducted face-to-face, which were all recorded on tape and later transcribed.

The interviews were based on informal, open ended questions, encouraging the participant to volunteer any information they felt comfortable sharing. The questions were designed to first provide a relaxed environment, with the initial questions for mostly putting the interviewee at ease. For the basis of the study, only responses to questions five through nine were analyzed, with the expectations that by the time these questions were reached, the interviewee was most likely to be candid in their responses. These questions also most directly addressed the focus area of the study, software process improvement adoption perceptions of the participants. Questions, five through nine, were:

5. Of the projects you have been involved with, have you worked within a process maturity model? If no, how do you feel the use of a process maturity model could have impacted your work? If yes, explain how you feel the process maturity model impacted your work.

6. When processes are implemented are they followed? Please explain.
7. Of the seven process areas listed in CMMI Level 2 – Managed, which do you believe have or has the most impact on your work? Describe what you believe that impact is and how it affects your work.
8. What issues do you see in attempting to implement process models within an organization?
9. If you could make changes to the way software and systems are developed in a project you have worked on or currently are working on, what would you do?

In many of the interviews, additional follow-up questions were added as appropriate to further probe or maintain a conversational flow to the discussion.

The participants were from West Virginia, Pennsylvania, Maryland, North Carolina, Georgia, and Washington D.C. areas. The individuals interviewed all have work histories with one of the following types of organizations: large defense contractor, large government agency, medium-sized scientific research organization, small independent software development company, small scientific research organization, government contractors, and university faculty. Additionally, of the 19 participants in the study, seven were working or have worked within a CMMI® appraised organization (as indicated by an "E" suffix in the participant ID), 12 have not.

The most common factor across all participants was the familiarity of working within a project environment. The average IT and/or software development experience for all participants was almost 18 years. The participants all had experience in the project environment and all were aware of the interactions between software developers

and IT professionals within that environment. While the specific familiarity of CMMI® was widely varied, from just a familiarity through working within an appraised CMMI® organization, all were experienced with the concepts of SPI as a factor within project work environments. All participants were extremely literate, and even the two that did not complete college degree programs, were extremely well educated in their specific IT professional fields and had completed many years of college level work. Unfortunately, women make up a very small percentage of software developers and IT professionals so the participants in this study all were male.

Table 4

Participant Information

ID	Age	Experience	Education	Title
P3	31	7 years	MSIS	SW Engineer/ IT Lecturer
P11	46	20 years	BSCS	Principle Database Architect
P42	41	17 years	Some College	Systems Analyst
P122	38	21 years	BS Business	Systems Engineer
P123	44	20 years	PhD, MS, BS CS	Senior Staff SW Engineer
P147	38	18 years	AS Civil Eng	Senior Project Engineer
P628	37	26 years	BSCS	Software Engineer
P629	25	18 years	BS CS/Math	Senior Network Specialist
P630	29	6 years	MSSE	Software Developer
P719	36	13 years	MS Math, MSSE	Software Engineer
P5150	24	13 years	BSCS,MBA	Senior System Engineer

P11637E	44	16 years	BSCS	Senior Programmer/Analyst
P16238E	42	19 years	MS Info Systems	Project Manager
P18432E	44	18 years	BA, MBA	Technical Manager
P44261E	30	12 years	BS Poli Sci	Project Manager
P45987E	44	22 years	BS, MSCS	Project Manager
P78952E	40	15 years	BSCS	Project Manager
P80327E	52	30 years	BSEE	Lead Software Tester
P102757	47	23 years	Some college	Senior System Analyst

Observational data was collected through the process of the researcher participating in various software development activities over the course of the study. These activities included observation of meetings convened to address specific software development issues such as requirements management, software design, and customer project status presentations. Additionally, observations were made through the exchange of various work emails, reports, and documentation. Interactions between software developers and other professional IT support staff were also noted and observed, as were the organizational structure of the specific projects, management initiatives, and employee reactions to specific tasks and duties.

This observational data is confidential and is not directly published as a part of the study; however, the observations served as a means of adding trustworthiness to the responses of individual software developers and IT professionals in response to the interview questions. As the researcher was a participant observer, this allowed for greater understanding and better communication of the responses as compared to a

researcher outside of the work environment with little direct project experience and background. However, as noted as possible researcher bias, this familiarity with projects and individuals was not brought into the specific data analysis – analysis of interview data was performed on an anonymous basis and without a mingling of this observational or experiential researcher data. As a part of the grounded theory analysis, the interview data spoke for itself.

As for specific CMMI® practices observed within appraised organizations, such observations were noted, but not incorporated into the study as a significant contribution. The actual CMMI® practices were beyond the scope of this study. While the reactions of those individuals operating within a CMMI® or SPI environment may have varied from those working outside of such a structure, this serves only as data for possible future analysis. This study was not intended to seek the differences or perform a comparative analysis of software developers and IT professionals working within or outside of a CMMI® structured project. The project does not even propose to validate the success or failure of projects based on use of the CMMI® model or even measure the degree of success in a project based on its level of CMMI® adoption would also be a study of interest. The focus, as stated earlier in the research questions was "…what will emerge as substantive theories in adopting Software Process Improvement strategies?" and "…what organizational change substantive theories, if any, will also emerge? " While comparison of individuals from each work environment may be of interest for future studies, it was beyond the scope of this particular study. Regardless of the organization's existing SPI, the focus of this study is on the emergence of

substantive theories of SPI adoption and how they fit within a set of organizational change steps.

Initial Data Analysis and Open Coding

The results of the study identified several areas of interest for the start of open coding. Initial open coding comes from the top or highest level of data analysis, serving as the first set of broadest categorization of the data. By use of the constant comparative process, reviewing and interpreting the participant's comments at this level, these categories were identified. These initial coding categories were identified as 18 areas from the interview data:

Table 5

Initial Categories from Open Coding of Data

Number	Category/Coding Name	Type
100	Model experience	Open
101	Implementation concerns	Open
102	Process hierarchy	Open
103	Process acceptance	Open
104	Change adoption	Open
105	Education	Open
106	Management commitment	Open
107	Communication	Open
108	Process busy work	Open
109	Blind development	Open
110	Development quality	Open

111	Customer expectations	Open
112	Lack of process	Open
113	Process failure	Open
114	Cost of process	Open
115	Organization maturity	Open
116	Team building	Open
117	Accountability	Open

As each of the participant interviews was analyzed, the responses were broken into discrete thought groupings and assigned to an initial open code. As the analysis proceeded, certain categories became more rapidly saturated than others. The top ten categories, which represented roughly 83% of the data, were:

Table 6

Top Ten Open Code Categories

#	Number	Category/Coding Name	Count	% of Data Items
1	103	Processes acceptance	58	21%
2	106	Management commitment	27	9%
3	102	Process hierarchy	25	9%
4	104	Change adoption	22	8%
5	112	Lack of process	22	8%
6	105	Education	19	7%
7	110	Development quality	16	5%
8	101	Implementation concerns	15	5%

9	107	Communication	13	4%
10	113	Process failure	10	3%

The significance of this first pass at the data to open coding was not necessarily that the specific number of responses in each open coding, but rather that 83% of the data items were assigned to these ten out of 18 codes. The immediate determination then, at least for a constant comparison of data in an open coding environment, was that eight of the 18 coding categories were most likely not grounded enough in the data to assist in the generation of theory. Therefore, the decision was made to drop those categories and move the related participant data to the next best fitting category. The results of dropping the eight categories and moving data to next best fit category resulted in 100% of the participant data being placed in the following ten open coding categories:

Table 7

Ten Open Code Categories after Further Open Coding and Data Analysis

#	Number	Category/Coding Name	Count	% of Data Items
1	103	Processes acceptance	59	22%
2	106	Management commitment	37	14%
3	110	Development quality	31	11%
4	102	Process hierarchy	25	9%
5	112	Lack of Process	25	9%
6	101	Implementation concerns	24	9%
7	104	Change adoption	22	8%

8	105	Education	20	7%
9	107	Communication	15	6%
10	113	Process failure	13	5%

While the dropping of the eight coding categories and shifting of data to different categories caused some minor changes in the total participant comments per category, only the "Development quality" category gained substantially, from 16 to 31 participant data items.

Under constant comparison of data, however, the relevance of initial open coding is eventually a non-issue as the data were further analyzed for development into themes or "core categories" in Glaserian terms.

This step was accomplished through a slow and tedious examination of the original 271 participant data items as they appear in the raw database report in Appendix B. While each of the memos is specific to all the captured data, the memos reported in bold italics were the first to significantly appear to support the development of theory. During initial memoing of the data, a flag in the database was set to "True" for any memos that may potentially be part of future substantive theory. This flag was based on the immediate analysis during the constant comparison of data and predicted theory ideas were formulated. This is part of the common sense impressions of the data that Glaser refers to in his generating process (1978, pp. 15-17).

Memoing is accomplished in a very open and free-form manner. In the analysis of each data statement, evaluative or speculative thinking is encouraged and documented. There is no set format or structure to memos (Glaser, 1978. pp. 83-92).

They can be a couple of quick thoughts or several pages. The following is a sample form the memo data:

Table 8

Sample Researcher Memoing from Raw Data

Participant	Interview Data	Researcher Memo
P42	Human nature doesn't, we're lazy, humans are somewhat lazy by nature; If given the opportunity	***The extra effort of managing processes for those that have dodged the bullet in the past always seems like a hassle and a reason to resist. - Change resistance Memo: 1139***
P628	management excluded the developers; one guy who was the big man, he was excluded from the meetings; kind of hard to understand why a supervisor was excluded from the meetings there was some head knocking there	If you intentionally break the communications, you can show to your manager the failure of process - sabotage and keep your job! A hideous perspective of some of those in power (as above) Management Commitment Memo: 1108

The above sample shows two types of data memoing. Where the memo is bold, this is a sample where the first memoing pass established that the data and memo represented a significant contribution toward theory. The second sample, from Participant 628, while important information to consider was not deemed as significant in the building of substantive theory during the initial memoing.

The statement by Participant 42 is flagged as having more significance in the building of theory in that it strongly supports several initial categories: Change Adoption, Process Acceptance, Implementation Concerns, and even, to some extent, Process Failure. The second sample, while informational significant and supportive of Management Commitment (in this case a lack of), it is not as broadly supportive of substantive theory from the researcher's perspective. While it still contributes to the overall building of substantive theory, it is considered, as most of the data, to not be one

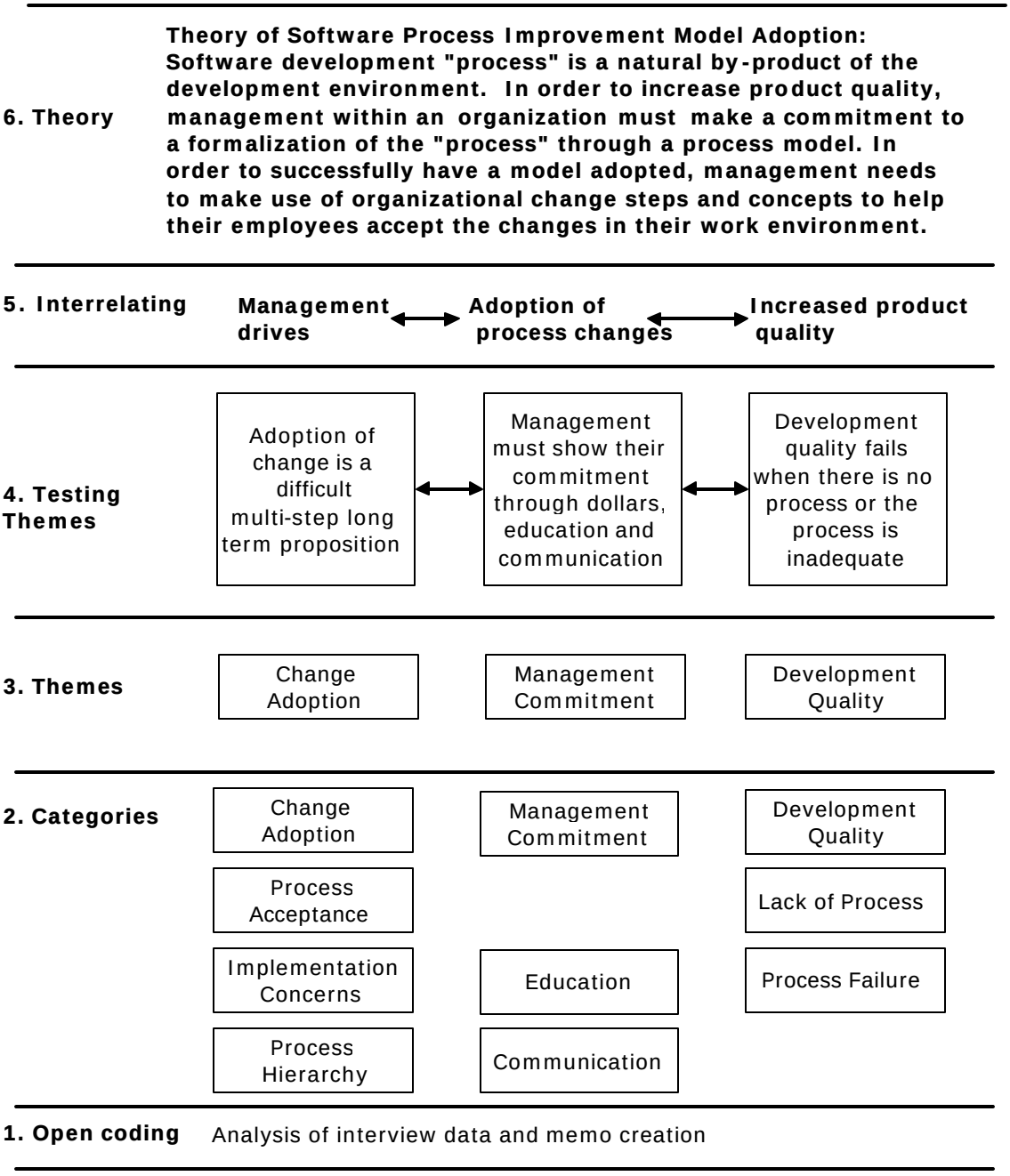
of the more significant pieces of data. This is an example of the researcher's perspective as the grounded theory process is built. Other researchers may see more significance in the statement, but grounded theory analysis is built based on the researcher looking to find what they perceive as the most significant data items.

The use of the specialized database created for this study assisted greatly in the creation of memos by providing immediately linked memo data fields for researcher interpretation of the data. As each data component, be it a phrase or multiple sentences was analyzed, memos were immediately captured in the database. The memo field in the database was completely open ended which allowed for the researcher to make as few as one or two word comments or unlimited pages if so desired. Most responses consisted of one or two sentences. The use of this database system allowed for a rapid development of memos based on the first impression of reading a specific data item and insured an easy method for linking memos to specific data. This was the open coding step of the grounded theory approach for the study.

The complete six levels are shown in the following figure; the figure is read from bottom up, step one up to step six. Each level is then further discussed in detail.

Figure 4

Six Numbered Steps Representing the Steps of this Grounded Theory Emergence



The Six Steps of Grounded Theory Emergence

The six steps led to the development of substantive theory. Those steps, starting at the bottom of the Figure 4 took the analysis from the initial evaluation of the interview data to the creation of substantive theory.

Open Coding

The open coding of the data was the first step in the analysis of the interview data. The first 18 categories, as shown in Table 5 were created from the initial data analysis of the interview data. As the data was read and constantly compared, the next step, Step 2, led to the development of the categories.

Categories

There were initially 18 categories created during the open coding process. The first categories were generated as the general reading of the data occurred as suggested by Glaser (1978, p. 15). Through this familiarization of the data and review some of the discussions, the initial coding categories were developed.

Each data element, the extracted words from the participant interview, was read and analyzed. In response to the interview questions, some of the data elements were easy to immediately classify. A data element such as element 1177, "I would say project planning is probably important…" would immediately fit in the "Process hierarchy" category. However, this would represent only the first pass at the data in the initial open coding. In subsequent analysis of the data item, element 1177 can also be seen as a value judgment by the words "probably important" and would therefore also fit nicely in the category "Implementation concerns." Further analysis would also lead to this data being part of the "Development quality" category also.

As Gasson (2004, p. 84) observed, “The researcher must continually ask whether the analysis of the new data provides similar themes and categories to previous data or whether other patterns emerge.” The first choice category may appear to have the greatest significance for the data, but on the whole, the data may also support previous data or other emergent categories. This is the constant comparison process in action and was utilized during the analysis of the data for this study.

In another example of the data analysis, data element 1069 is not so easy to immediately place. It reads, “If you had someone new on a project or new to CMMI, I think that’s where you are going to encounter your slowdowns.” The choice to place this item initially in the “Education” category was based on someone needing training to better perform. But should this have been initially under “Development quality?” Or possibly “Implementation concerns?” The point is that each data element does potentially have several category fits, and through the constant comparison analysis, these fits were identified. The important technique used in this analysis was to maintain consistency in the placement of specific data statements. As more and more data was analyzed and placed under the categories, and the categories were saturated and edited, and the research led to the next level of the process, refining to core categories and themes.

Themes

Themes and core categories developed as the continual constant comparison process filtered and scoped the data. After the process was repeated several times, particular themes began to emerge. In no specific order, those themes were: (a) Change adoption, (b) Management commitment, and (c) Development quality.

Change adoption. As suspected and indicated in one of the initial research questions, change in the work environment was frequently discussed by the participants. Data such as element 1024, “you know people always have ways they like to do things” and element 1156, “Getting people to step out of their comfort zone or the way they’ve always done it “ were typical of the 22 data items from the interviews.

This theme consistently appeared throughout the discussions, with some participants appearing quite intense in their responses, such as data element 1181 “Everyone knows it’s a pain in the neck and everyone tries to circumvent it because it is difficult…” from participant P123. The theme of adopting change in the workplace was most definitely a concern, maybe to the level of somewhat fearful in some, and quickly rose to the top as one of the primary themes of the study.

Management commitment. If any theme within the data that emerged with significant consistency from most every participant, it would have to be the theme of management commitment. The study participants, in general, seemed to focus the failure of process implementation (or specific projects) as a lack of what was frequently termed, “management buy-in.” Whether it was general employee versus management griping, or a way to remove blame from one’s own issues, or if it was just a genuine and truthful observation, the participants, for the most part, held back little in dropping the responsibility on management.

Some statements were simple and direct, such as element 1191 “Management’s always a problem; Management has to buy-in to it.” Others were a little softer, such as element 1179, “I think you need your management buy-in” but the point was still clear

that the individual felt what needed to be done could only happen with management fully involved.

Development quality. This theme, although inferred repeatedly through discussions of software development and process improvement, was rarely openly stated. It emerged as kind of a "catch all" for a lot of issues that the software developers and IT professionals regularly face in their jobs. Unlike the mature manufacturing process of automobiles, where steps and process are planned to a robotic level, it appeared that most of the software developers and IT professionals were aware that there was not a clear process for the development of software. As in element 1028, "trying to come to an agreement on what we're going to do" the participant displays concern about product quality. More direct statements on the development quality theme were, "Critical applications are being developed without sufficient requirements control and development." This element 1245 expressed the tone of great concern over the quality of products being delivered.

In the "Process hierarchy" open coding category, the statement "Process Product Quality Assurance, I think that's a high impact" from element 1165 is a more direct example of the view of quality by participant P147. Since most of the interviews were conducted with software developers, the more common responses to the ranking of processes were requirements management and configuration management – tangible process activities software developers work with on a daily basis. However, this participant, who was not a software developer, immediately saw development quality as a most significant process area. Still, the quality inference is most always

present, such as the following software developer comment, “If you know what the requirements are, then you can work on how you actually implement” in element 1014.

Testing Themes

The themes were further evaluated for trustworthiness at this point in this grounded theory analysis by reviewing observations and literature in relation to the chosen themes. Where the participant interview data serves as the basis, or grounding of the theory, the literature and observations help triangulate the themes from perspectives other than those of the study participants.

In reference to the management commitment theme, the following statement was from the Software Engineering Institute’s Software Engineering Process Group (SEPG) 2006 conference, “Senior management has limited attention span to process improvement – they are focused on many other issues” (O’Toole, 2006). So maybe the appearance of lack of management buy-in by the study participants is a fairly trustworthy statement. Kautz and Nielsen observed in their journal article, “Management was reluctant and provided little support for the SEPG” (2004, p. 14). In a personal conversation with a corporate division director, the following support of management commitment was made, “Senior management investment in the process improvement is key. Here, senior management buy-in was relatively easy since there were existing contracts and re-competes at risk – it was easy to point to as need” (Anonymous, personal communication, December 13, 2005).

Observations of management commitment to SPI in the workplace were further discussed by Kautz and Nielsen as they compared and contrasted two companies, one successful, the other not. For the successful company, they found the following

characteristics of management: (a) clear leadership (b) management strongly committed (c) project leaders dedicated, and d) management enthusiastic and supportive. For the unsuccessful software process implementation organizations, the characteristics observed were: (a) vague leadership (b) management weakly committed (c) project leaders doubtful and (d) management sympathetic, but not resolute (2004, p.17).

Change adoption has been noted by many business leaders, professionals, and even some SPI authors. In the literature, Mathiassen, Pries-Heje, and Ngwenyama (2002, p. 25) wrote, "Software Process Improvement is a challenging and complex change process. For it to succeed, effective management is required." This echoes the data from many participants in the study, such as element 1208 "Yeah, you know its, its, reluctance to change; not committing enough resources to support the change." Or lack of effective management in the change adoption, as in element 1149, "who from almost day one could not establish credibility with the project team –and, people refused to follow the process he was trying to establish."

Additional support of the themes of the study can found in the words of Steven R. Rakitin, president of Software Quality Consulting, Inc. In his book, Rakitin sums up the software quality issue as: "Software development organizations known for the high quality of their products (such as HP and Motorola) have learned how to measure and control variation. These organizations all have well-defined software development processes" (2001, p. 35). Like other manufacturer's, quality conscious software firms understand the theme of development quality. Participants mentioned this focus on structure, such as in element 1123, "You start with the framework and build everything

around it." The framework for building the software – similar to the injection mold for an automobile bumper – each helps keep the output product at a consistent quality level.

Interrelating

Themes were reviewed for interrelationships and combined to create substantive theory. From the first level of analyzing the interview data and creating memos, then stepping up through the categories, themes and core categories, to testing themes through supporting literature and observations, the final step before the development of the theory was to examine these relationships.

To start with, the goal of the software developers and IT professionals, whether spoken or not, was to develop quality products. The data and interview sessions support this goal – participants were genuinely concerned in doing a good job. Each of the interviews carried a tone of, "Yes, I would like to see things improve." By the very nature of the participants volunteering to be interviewed for this study showed a level of quality concern. So the first assumption on the development theme is the software developers and IT professionals in this study wish to produce quality products.

Next, the interviews revealed repeatedly that the software developers and IT professionals all had a pretty good grasp on the reality of the situation. They spoke candidly about what was working and what was not. Pretty much universally, they indicated that there was a "process" to the development of software products. Although some may have not known specific CMMI® nomenclature or process improvement methodologies, for the most part, the responses indicated an understanding of the need for regular process within the software development environment.

From there, depending upon the specific situation they personally were involved, the question of adopting new Software Process Improvement methods or an SPI model were directly proportionate to their perception of just how much needed to be done. For some, the sense of urgency was high – for others, it was low. However, pretty much all the participants were able to answer the question regarding what they would do to change the way things were being done and the answers pretty much uniformly confirmed their perceptions that there was a need for some type of change. Therefore, it has been acknowledged that there are potentially quality improvement opportunities in their product development and that these quality improvements would most like result in some type of change. This addresses the themes of "Change adoption" and "Development quality."

Lastly, as discussed previously, virtually all participants were quite vocal, albeit to various degrees, in their perception that for change to take place and for quality to improve, management would have to not only buy-in, but drive. The "Management commitment" theme is the last crucial piece to making changes to improve quality. Several participants lamented to the fact that management has not committed, there were no funds, there was no structure, training was missing or not fully completed, and several other indicators of the software developer and IT professional perception that management was just not doing enough to make things better.

Grounded Theory

Through observations and literature, these grounded data were also reviewed, and found to be trustworthy. Therefore, the grounded theory that has emerged from this study may be stated as follows:

Theory of Software Process Improvement Model Adoption: Software development "process" is a natural by-product of the development environment. In order to increase product quality, management within an organization must make a commitment to a formalization of the "process" through a process model. In order to successfully have a model adopted, management needs to make use of organizational change steps and concepts to help their employees accept the changes in their work environment.

The creation of a software product, as emerged from this analysis, has been understood to by its very nature cause the creation of process – a natural by-product of the work. It has been noted that the software developers and IT professionals in the work environment will utilize their individually created processes in order to complete their work. If an assembly line analogy were top be used, the software development team would be like many different individuals placing the body work on a vehicle as it passed, using their own “process” for accomplishing the task. One may choose to use sheet metal screws, while another may use rivets. A third may even choose to directly weld parts to the vehicle. By the time these vehicles would arrive at the end of the assembly line, various levels of quality would be inherent in each one. Management would have little control over the product quality, would have cost variances from rivets to welding equipment, and the end users would end up with cars of various degrees in quality.

The auto industry, many years ago, made the decision to formalize the process on the application of body parts. Initially, workers were trained to consistently place parts on the vehicle (as is still the practice today) and these processes were refined to

the point where specific robotic machines eventually were developed to place the same parts repeatedly on the vehicle. As noted in a U.S. Department of Commerce Office of Technology Policy Report on the automotive industry:

> If the parts do not fit when the manufacturer attempts to put them together, the system has a defect that must be tracked down and eliminated. Thus, auto companies focus a great deal of attention on understanding and improving the manufacturing process. (Fine, St. Clair, Lafrance, & Hillebrand, 1996, p.20)

So as stated in the first portion of the substantive theory of software process improvement model adoption, the management of a software development organization must take a similar position to the automotive manufacturing world of improving process by the formalization and adoption of specific software development practices. The naturally occurring development processes of software developers need to be understood and improved and formalized as a part of the software manufacturing process.

Secondly, in order to implement a formalized software process improvement and obtain successful adoption of these improvements, management must understand the organizational change issues associated with process model adoption. Through use of organizational change steps as identified by John Kotter (1996) and used as a backdrop for this study, management teams may successfully implement a software process improvement adoption model. The steps, suggested by Kotter have previously been discussed and are: (a) create a sense of urgency (b) create a guiding coalition (c) create a vision and strategy (d) communicate the vision and strategy (e) empower the

employees to act (f) generate and acknowledge short term wins (g) consolidating gains and producing more change and (h) anchoring the approaches in the culture.

Although the specific wording of many of these steps were not used by the study participants, there are many supporting statements by the participants that reinforce this conclusion that organizational change issues must be accounted for as part of the overall adoption process. Some of the supporting statements from the participants were: P628, "If the project would have been planned better the requirements would have been managed better; And the whole scenario all the way down to, to the person doing the testing"data element 1104. Where was the vision? Who was empowered? Additionally, from P11637E, "Allow more active participation by all stakeholders, including the people who are going to use the solution, not just management" data element 1214. Once again, a concern for empowerment and a lack of coordinated communication. P3 brings up the immediate concern of needing to have a guiding coalition, "Getting people to step out of their comfort zone or the way they've always done it" in data element 1156. P5150 addressed both the issues of allowing complacency and under-communicating:

> I don't think people see that return on investment; I mean people get preached the return on investment from such a discipline as what you just showed me but often times some valuable piece of information gets lost in the I've got to get something done

in data element 1124. Communication concerns, identified by P147, "I think basic communications - if you are going to do this, if you are going to go through this process

improvement or implement this new process” data element 1167. You must communicate the software process improvement strategy and vision.

Chapter 6 - Conclusions

Study Summary

As discussed in the introduction of this study, in order to prevent or at least mitigate some of the problems of poorly written software, software engineers and IT professionals need to successfully adopt an SPI strategy. When SPI is implemented, changes are being made to the organization and you do not have a complete sense of what is ultimately involved (Kotter, 1996, p.139). Therefore, the guiding research questions for this study can be abbreviated to (a) what will emerge as substantive theories in adopting SPI strategies and (b) what organizational change substantive theories, if any, will also emerge?

Just as in the manufacturing of automobiles and other products, systematic processes and standard methods are utilized to improve and stabilize quality (Fine, St. Clair, Lafrance, & Hillebrand, 1996) For this study, CMMI® was utilized as a software process model example and Kotter's (1996) eight steps discussions on organizational change were used as a setting for the change adoption environment. The grounded theory data analysis then led to a substantive theory, a grounded theory of software process improvement model adoption.

Theory Components

The substantive theory from this research has several components for successful SPI model adoption: (a) concern for quality (b) understanding of development process (c) management commitment to a process model (d) management understanding of organizational change issues and (e) organizational change steps to lead employees to a successful adoption.

Concern for quality. If there is little or no concern for product quality, the remaining issues become a moot point. Management must first have a desire to produce a quality product. This desire must also be held by the software developers and IT professionals performing the work. The data collected showed there is a concern for quality at the developer and IT professional level. Several comments clearly pointed to this concern: Data element 1168 "the most successful project, implementation project I worked on, there was an absolute dedication from the outset to keep the end user in mind at all times" from participant P147 clearly shows concern for quality. This is echoed by participant P44261E with, "The most obvious example, is delivering a solution to the client that actually meets all of their expectations" in data element 1230. Lastly, the concern for quality is shown in data element 1110 "There were complete communication breakdown because once they excluded those two people from the meetings there's no communication happening, so nobody knew what was going on" from participant P628.

Understanding of development process. The development process carries its own professional, methodological, and environmental issues. Unless the management has a grasp on these issues, there is little opportunity for successful collaborative work to proceed. As discussed earlier, software developers will, by the very nature of their work, create process. There are innumerable processes to the development of a computer program, and unless the understanding of this development environment is communicated effectively to the individual developers or development teams, they will rely on their own resourcefulness to complete the project tasks on their own. As participant P3 observed. "I think I would have done more prototyping with more user

feedback” in data element 2156. Participant P629 additionally lamented, “somebody has to mess around changing something back; fixing something because one change was made and somebody wastes half an hour; that person may waste a half an hour then and you keep changing things and different stuff” on his frustration on the clashing of different development processes causing excess work and re-work in data element 1075.

Without an understanding of the natural tendency for development environments to create their own processes, management has no opportunity to put in place a structure to help frame this environment. Understanding this component of the theory is absolutely essential.

Management commitment to a process model. Once the management has understood that process will exist, with or without their direction, it is time for management to make a commitment to a specific process model. For this study, CMMI® was utilized as a SPI Model, specifically the components of Level 2: Managed. While there exists numerous models worldwide, the model itself is not nearly as important as the management commitment to a model. Again, whether an automobile manufacturer uses a robotic arm to mount a wheel on a vehicle or uses trained assembly line workers to mount the wheel, is immaterial – the important factor is the robotic arm or workers perform the task in a standard process fashion the same way every time. In software development environment, without the commitment to a model, it would be as if one day an auto worker decided to mount 15” tires with four lug bolts and the next day decided to mount 13” tires with 6 bolts, on only red cars.

Participant P630 noted, “[you] got to have manager buy-in; that they see the process is important; Not that they are concerned that they have outside people knowing they have process” in data element 1058. Regarding management driving, participant P3 commented, “You got to have a clear strategy and strategies come from leaders.” In data element 1155.

This “buy-in” and “clear strategy” from management are the critical components of driving the adoption of standardized processes and procedures. Whatever the model, the important factor is the commitment from the top level of management. Without the commitment, the process adoption will fail.

Management understanding of organizational change issues. Frequently, the management will see the need for improvement. When customers and employees are both complaining, then it is obvious there must be something that needs to be changed. However, unless the management personnel understand that change in itself creates issues in the organization, the corrective actions will be doomed from the start.

Software developers and IT professionals expressed repeatedly that they had concerns and that they saw a need for change to the way projects were running. In discussing processes and how change was needed, participant P719 observed, “How do you change your requirement? How do you add requirements? How do you delete them? How do you decide how you’re going to it and stuff” in data element 1019. This obviously is a concern for how a process can be changed or adopted. P102757 commented, “Yeah, you know its, its, reluctance to change. Not committing enough resources to support the change” in data element 1208. Management needs to commit enough resources to support this organizational issue and reluctance to change. Their

commitment will confirm their understanding of the needs to change and serve as the first step in addressing the reluctance to change issue.

Application of Kotter's Change Steps

For the purpose of this study, like the selection of the CMMI® as an SPI model, eight steps for organizational change were selected from the work of John P. Kotter's *Leading Change* (1996). Likewise, in the process model discussion, the importance of Kotter's steps (or his *model*) was not nearly as important as the understanding that recognition of organizational change issues must be addressed. Other models of change could adequately address the change issues – it is the recognition that change presents its own critical adoption issues that must be dealt with as a component of making SPI adoption successful. As participant P719 expressed, "a lot of educational issues - trying to get people to understand what a process model is" in data element 1023 is one of the key organizational change issues. Simply stating that we will now follow process is nothing more than simply stating we will now follow process. Requesting a *change* in how a process will be followed or implemented is a much more extensive issue. On educating members of the project, participant P122 discussed, "any new implementation or new CMMI processes need to really be um, the group needs to be educated" in data element 1189. Understanding the organizational impact of this education is critical to the success of the change adoption.

Organizational change steps to lead employees to a successful adoption. As the last portion of the theory states, in order to successfully implement an SPI adoption, there must at least be some organizational change steps identified and addressed.

Kotter's eight steps were utilized in this study as a means of addressing some of the change issues uncovered by the data analysis.

In varying degrees, Kotter's steps provide impact on the potentially successful implementation. In applying some of these steps to the results of this study, the following discussion observes specific steps as they may or may not successfully impact a process adoption concern.

Communicating by a factor of 1,000. In the case of the NASA Mars Polar Lander crash in 1999, this could be said to be under communicating to a factor of $125 million – the cost of the loss of the spacecraft due to lack of communication between contractors on the project. One contractor programmed functions using feet; the other used meters (CNN.COM, 1999). As participant P719 mentioned in a similar situation, "Especially when there's other groups that you just don't communicate with on a day-to-day basis" data element 1009. Participant P629 adds, "I think safe communication between the different teams is extremely important" in data element 1081.

Effective communication between teams and individuals – whether on a small software project or multi-million dollar space project – are essential to its success. If management under communicates the adoption of a process model by a factor of 1,000, what will be the outcome of that adoption? Most likely, it will fail. If the people do no know what they are supposed to be doing, what is the chance of it actually happening correctly or at all? Participant P719 went on to note, "[So things work better] when you know what you are doing" in data element 1031. Participant P123 explained, "We had to have a very clear division between people that weren't able to communicate

constantly in terms of independent pieces of the project that could be done" in data element 1184.

Communication effectiveness is part of the basic process implementation. One of the key areas of CMMI® Level 2: Managed is the Project Monitoring and Control. The specific goals and practices all actually focus on communicative practices such as performance progress, review of issues, and review of data against the plan, to name a few, all concerned with communication amongst the team members on the project. You cannot monitor commitments unless you are communicating effectively with those working to satisfy the commitments.

In conclusion, Kotter's observation on communication by a factor of 1,000 as an important factor of organizational change is definitely supportive of SPI model adoption. If various software developers and IT professionals on the project team are not aware of specific SPI adoption needs, it is up to management to make certain the practices are communicated – thousands of times if necessary. It is never enough for management to state one or two times that the company is adopting a SPI Model and expects the teams to immediately be working effectively under the new model. Communication of the SPI model must be continually communicated - the alternative is crashing space craft into distant planets.

Building a guiding coalition. Without an internal Software Process Engineering Group (SEPG) and Management Steering Group (MSG), who is going to make the adoption happen? Additionally, if these groups have no awareness whatsoever of organizational change issues, how will effective adoption be carried out? Can an organization just happen to be lucky enough to have a structured process just appear?

In order for a SPI model adoption to occur, there absolutely must be a coalition of individuals, both senior and middle managers, software developers and other IT professionals. It is not going to just "happen" spontaneously.

Kotter's discussion of building a coalition can be specifically adapted to the adoption of SPI in an organization. There must be enough people in a position of power, especially so that others may not be able block progress. These people should at least make up a major portion of the Management Steering Group. People with SPI expertise must be on the Software Process Engineering Group, lending their expertise to the specific SPI needs at hand. Do the two groups have enough credibility to be taken seriously by the SPI pronouncements? Enough power for enforcement of changes? Lastly, and most importantly, do the teams have enough combined strong leadership to drive the process adoption for the long run (1996, p. 57)?

Anchoring new approaches in the culture. The SPI CMMI® addresses this as the institutionalization of processes. This is where the adoption has succeeded to a level whereby it has become a part of the company's culture. There is no issue of addressing how a project will be managed; it is just automatically placed under the process model. While on the surface this appears to be the end of both the adoption of SPI and institutionalization of CMMI® processes, specifically in the software development world, there may be no anchoring or institutionalization.

Both of these concepts imply an end or completion of the model adoption or organizational change. The fact of the matter is technology changes and moves so rapidly that no company involved with the production of technology can take a stance of "We've reached the end!" On the contrary, changes made today will be modified or

dropped tomorrow if a technology based organization is to continue. While the approaches and the need for the processes will continue to evolve, regardless of attempts to manage and push change or process, evolution is a natural process that must be recognized as an additional component of the organization.

How many technology companies today are not using the Internet? How many still operate computers using DOS operating systems? What has happened to carbon paper and typewriter sales in technology companies over the last 40 years? Gordon Moore, cofounder of Intel, created what was to be Moore's law that stated in 1965 the number of transistors that were utilized on an integrated circuit would approximately double every year (Moore, 1965). For the last 42 years, this has mostly held true. This phenomenal growth of technology has greatly impacted the change of organizations, especially those based in technology, such as software development.

So can Kotter's anchoring or the CMMI® processes really be the corporate culture? Or does the rapid change of technology prevent an organizational culture to ever be anchored or institutionalized? Based on the data collected in this study, most technology organizations are in constant state of flux and while anchoring changes and institutionalization of processes are noble goals, they just may be unreachable in today's and tomorrow's companies. So as things change so rapidly, will there always be the comments of participant P628, "Processes are not followed in the environment I'm working in at the moment" data element 1093? Or as P630 observed, "My last project was, I always thought ,the biggest issue on the project, the reason that we couldn't seem to get off the ground at all, because we could never nail down what we were expected to be building" date element 1051. While change and process models

may indeed prevent issues of this magnitude, it may be overly optimistic to believe process models can be followed and change will be always accepted.

Contributions to the Industry

While specific components of this study reveal many areas of concern over SPI adoption and the understanding of change based on Kotter's eight steps, the conclusions definitely add to the body of knowledge concerning both SPI model adoption and the use of organizational change concepts as a component of that adoption. A grounded theory approach has uncovered some strong core categories and themes – management must drive, quality must be a concern, organizational change issues will surface – all of these discussions and observations from the participants have brought new knowledge to the field of software engineering.

Additional Conclusions

Some peripheral understandings and serendipitous knowledge found is that left to their own devices, software developers will create process, regardless of how informal it may be. The nature of the software development environment is such that repeatability in work methods rewards the software developer with the ability to expand on their previous development skills. By the nature of this informal process, then, process adoption might be best implemented not by stating we are now going to have CMMI® processes, but rather a low key, here are some names and a vocabulary for some of the things you are already doing. After all, the CMMI® model is not prescriptive, but descriptive in its nature. Most software developers know best what processes to do in their day by day job – a successful adoption of CMMI® may then be helping those developers learn to place them under the model's umbrella.

Study Limitations

Dr. Barney Glaser warned that using grounded theory for a dissertation has some definite limitations (B. Glaser, personal email, October 18, 2004). After working through this process I more fully understand his concerns, but still feel the methodology served the topic well. However, as indicated by his email, the process takes much more time and effort than some other forms of qualitative or quantitative study. Grounded theory provided numerous insights into the data, especially utilizing the constant comparative analysis component, however, at a great cost of time and effort. For the purposes of a dissertation, it was more like rabbit hunting with an elephant gun. Additionally, the grounded theory process has some areas that are directly incongruent with a typical dissertation format. For instance, literature reviews are supposed to be skipped until the analysis phase, and specific research questions are better described as "guiding ideas" since the discovered data is supposed to be the driving factor of the study. Data analysis happens more immediately and results of that analysis then spurs further research. All of these grounded theory properties do create some additional difficulty in adapting to a standard dissertation format.

This study has barely scratched the surface of what I believe are many more serious Software Process Improvement issues. Even this small data set of 19 participants could provide much more grounded theory analysis time. As with most tools, the more one makes use of grounded theory analysis, the better the understanding the researcher gains of that tool. Additional grounded theory research in software process improvement models and organizational change strategies would provide even greater depth and understanding of the adoption phenomena. Freeing the

research from a dissertation format and following more traditional grounded theory structures would also most likely provide more in-depth results.

Future Research

This study uncovered many concerns and issues in the adoption of Software Process Improvement as discussed and demonstrated by the participants' responses. However, the study did not include discussions with the management teams that were frequently referenced by the software developers and IT professionals. For future study, the concerns that surfaced from the interviews, observations, and literature research should be part of a grounded theory based management perspective study on SPI and change adoption issues. A thorough study utilizing much of the same criteria as this study with managers involved with software development organizations would provide some opportunities for comparative analysis between the worker and manager perspective. Lastly, the combining of the data would allow for even more grounded theory analysis that could potentially lead to a formal theory of SPI adoption.

Additionally, John P. Kotter's work on change adoption issues is just one perspective, as the CMMI® model is only one process improvement model. Future research could focus on alternative or additional organizational change and different process model adoptions. A comparative study could uncover some inherent strengths or deficiencies in either Kotter's approaches or the CMMI® model adoption.

Lastly, only seven of the participants were from an organization appraised with projects at a CMMI® Level 2: Managed. Future research in organizations within the CMMI® Levels 3, 4 or 5 with significantly more participants with broader model adoption experience would also shed more light on some of the successful adoption

methodologies. This type of study could provide a lot more positive data on successful adoption, especially from Level 4 and 5 software development organizations which would be operating at optimum performance.

References

Arnold, D. (1996). Explosion of the Ariane 5. Retrieved March 20, 2005, from http://www.ima.umn.edu/~arnold/455.f96/disasters.html.

Axelrod, R. (2001). Democratic approaches to change make a big difference in turbulent times. *Harvard Management Update, 6*(11), 3-4.

Beer, M., Eisenstat, R. A. & Spector, B. (1990). Why change programs don't produce change. *Harvard Business Review*,*68*(6), 158- 9.

Boisjoly, R. (1995). Memo from Roger Boisjoly on o-ring erosion. Retrieved November 11, 2004, from http://onlineethics.org/moral/boisjoly/MTImemo1.html.

Brooks, F. P. (1995). *The mythical man-month: Essays on software engineering*. (Anniversary ed.). Boston: Addison-Wesley.

Chenitz, W. C. & Swanson, J. M. (1986). *From practice to grounded theory: Qualitative research in nursing.* Menlo Park, CA: Addison-Wesley.

CMMI Product Team. (2002). *Capability maturity model integration version 1.1*. *CMMI for systems engineering and software engineering - staged representation* (CMMI-SE/SW, V1.1 ed. Pittsburgh, PA: Software Engineering Institute.

CNN.COM. (1999). NASA's metric confusion caused Mars orbiter loss. Retrieved Febraury 24, 2007 from http://www.cnn.com/TECH/space/9909/30/mars.metric/

Connell, C. (2002). Introduction to software capability maturity model. Woburn, MA: CHC-3. Retrieved April 11, 2005 from http://www.chc-3.com/talk/sw-cmm.ppt.

Deming, W. E. (1986). *Out of the crisis*. Cambridge, Mass.: Massachusetts Institute of Technology.

Fantina, R. (2005). *Practical software process improvement.* Norwood, MA: Artech House.

Fine, C.H., St. Clair, R., Lafrance, J.C., Hillebrand, D. (1996). *The U.S. automobile manufacturing industry.* Retrieved February 18, 2007 from www.technology.gov/Reports/autos/auto.pdf

Franke, R. H. & Kaul, J. D. (1978). The Hawthorne experiments: first statistical interpretation. *American Sociological Review, 43*(5), 623-643.

Frieden, T. (2005). *Report: FBI wasted millions on 'Virtual Case File'*. Retrieved April 9, 2005, from http://www.cnn.com/2005/US/02/03/fbi.computers.

Gage, D., McCormick, J. (2004, March). Eight fatal software related accidents [Electronic version]. *Baseline,1* (28). Retrieved March 20, 2005, from, http://www.baselinemag.com/article2/0,1397,1543590,00.asp.

Gasson, S. (2004). Rigor in grounded theory research: An interpretive perspective on generating theory from qualitative field studies. In M. Whitman and A.B. Woszczynski (Eds.), *The handbook of information systems research* (pp. 79-24). Hershey, PA: Idea Group Publishing.

Glaser, B. & Strauss, A. (1967). *The discovery of grounded theory.* Mill Valley, CA: Sociology Press.

Glaser, B. G. (1978). *Theoretical sensitivity.* Mill Valley, CA: Sociology Press.

Glaser, B. G. (1992). *Basics of grounded theory analysis*. Mill Valley, CA: Sociology Press.

Glaser, B. G. (1996). *Gerund grounded theory*. Mill Valley, CA: Sociology Press.

Goldenson, D. & Gibson, D. (2003). *Demonstrating the impact and benefits of CMMI ®: an update and preliminary results.* Pittsburgh, PA: Software Engineering Institute.

Hammond, J. S., Keeney, R. L. & Raiffa, H. (1998). The hidden traps in decision making. *Harvard Business Review, 76*(5), 47-55.

Harrington, H. (1991). *Business process improvement: the breakthrough for total quality, productivity, and competitiveness*. New York: McGraw-Hill.

Harry, B., Sturges, K., & Klingner, J. (2005). Mapping the process: An exemplar of process and challenge in grounded theory analysis. *Educational Researcher, 34* (2), 3-13.

Harter, D. E., Krishnan, M. S., & Slaughter, S. A. (2000). Effects of process maturity on quality, cycle time, and effort in software product development. *Management Science,* 46(4), 451-467.

Harvey, K. E. (1986). *Summary of the SEI workshop on software configuration management*. Pittsburgh, PA: Software Engineering Institute.

Herbsled, J. D. & Goldenson, D. R. (1996, 1996/03/27). A systematic survey of CMM experience and results. Paper presented at the ICSE-18, Berlin, Germany.

Herzberg, F. (2003). One more time: how do you motivate employees? *Harvard Business Review, 81*(1), 3-11.

Humphrey, W. (1989). *Managing the software process*. Reading, MA: Addison-Wesley.

Humphrey, W., Kitson, D., & Gale, J. (1990). A comparison of U.S. and Japanese software process maturity. Pittsburgh, PA: Software Engineering Institute.

Humphrey, W. (1995). *A discipline for software engineering*. Reading, MA: Addison-Wesley Publishing Company, Inc.

Ittner, C. D. & Larcker, D. F. (1997). The performance effects of process management techniques. *Management Science, 43*(4), 522-535.

Jalote, P. (1999). *CMM in practice: processes for executing software projects at Infosys*. Reading, MA: Addison-Wesley.

Kautz, K. & Nielsen, P. A. (2004). Understanding the implementation of Software Process Improvement innovations in software organizations. *Information Systems Journal, 14*(1), 3-22.

Kotter, J. P. (1995). Leading change: why transformation efforts fail. *Harvard Business Review, 73*(2), 59-68.

Kotter, J. P. (1996). *Leading Change*. Boston, MA: Harvard Business School Press.

Kotter, J. P. (2002). *The heart of change: Real life stories of how people change their organizations.* Boston, MA: Harvard Business School Press.

Kulpa, Margaret K & Johnson, K. A. (2003). *Interpreting the CMMI: a process improvement approach.* Boca Raton, FL: Auerbach Publications.

Leveson, N., & Turner, C. (1993). An investigation of the Therac-25 accidents. *IEEE Computer, 26*(7), 18-41.

Lincoln, Y. S. & Guba, E. G. (1985). *Naturalistic inquiry*. Beverly Hills, CA: Sage Publications, Inc.

Luecke, R. (2003). *Managing change and transition*. Boston, MA: Harvard Business School Press.

Lynn, G. S. (1998). New product team learning: developing and profiting from your knowledge capital. *California Management Review, 40*(4), 74-93.

Marash, S. (2001). 21st century quality. *Quality Digest*, *21*, 2.

Mathiassen, L., Pries-Heje, J., & Ngwenyama, O. (2002) *Improving software organizations.* Upper Saddle River, NJ: Addison-Wesley.

McFeeley, B. (1996). IDEAL[SM] a users guide for software process improvement. Pittsburgh, PA: Software Engineering Institute.

McFeeley, B. (2004). *The IDEAL model*. Retrieved 12/04/2004, from http://www.sei.cmu.edu/ideal/.

Moore, G. (1965). Cramming more components onto integrated circuits. Retrieved February 25, 2007 from ftp://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf

Nathanial, A. (2003). *A grounded theory of moral reckoning in nursing*. Dissertation, West Virginia University, Morgantown.

Ngwenyama, O. & Nielsen, P.A. (2003). Competing values in software process improvement: an assumption analysis of CMM from an organizational culture perspective. Engineering Management, IEEE Transactions on 50(1 SN - 0018-9391), 100-112.

Niazi, M., Wilson, D., & Zowghi, D. (2005). A maturity model for implementation of software process improvement: an empirical study. *The Journal of Systems and Software, 74*(2), 155-172.

Norman, W. G. (2003). *Implementing CMMI in a previously unstructured environment.* Unpublished master's thesis, West Virginia University, Morgantown, WV.

Orlikowski, W. J. (1996). Improvising organizational transformation over time: a situated change perspective. *Information Systems Research* 7(1), 63-93.

O'Toole, P. (2006, March). *Do's and don'ts of process improvement.* Paper presented at the Software Engineering Process Group conference, Nashville, TN.

Patton, M. Q. (2002). *Qualitative research and evaluation methods* (3rd Ed.). New York: Sage Publications, Inc.

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). *Capability maturity model for software, Version 1.1*. Pittsburgh, PA: Software Engineering Institute.

Paulk, M. & Garcia, S. (1994). The Impact of evolving the capability maturity model to Version 1.1. Retrieved November 20, 2004 from http://www.stsc.hill.af.mil/crosstalk/1994/09/xt94d09d.asp.

Piderit, S. K. (2000). Rethinking resistance and recognizing ambivalence: a multidimensional view of attitudes toward an organizational change. *Academy of Management Review, 25*(4), 783-795.

Rainer, A. & Hall, T. (2001). An analysis of some core studies of software process improvement. *Software Process Improvement and Practice*, *6*(4), 169-187.

Rakitin, S.R. (2001). *Software verification and validation for practitioners and mangers.* Norwood, MA: Artech House, Inc.

Ravichandran, T. & Rai, A. (2000). Quality management in systems development: an organizational system perspective. *MIS Quarterly, 24*(3), 381-416.

Reid, P. C. (1990). *Well made in America: Lessons from Harley-Davidson on being the best*. New York: McGraw-Hill.

Rogers, E. M. (2003). *Diffusion of innovations* (5th Ed.). New York: Free Press.

Saunders, R. M. (1999, 1999/08/01). Communicating change: a dozen tips from the experts. (Cover story). *Harvard Management Communication Letter*, 2, 1-3.

Schaffer, R. H. & Thomson, H. A. (1992). Successful change programs begin with results. *Harvard Business Review, 70*(1), 80-90.

Schreiber, R. S., & Stern, P. N. (2001). *Using grounded theory in nursing*. New York: Springer Publishing Company.

Software Engineering and Analysis Team. (2002). Process maturity profile of the software community 2002 mid year update. Retrieved 11/30/2004 from http://www.sei.cmu.edu/sema/pdf/SW-CMM/2002aug.pdf.

Senge, P. M. (1994). *The fifth discipline*. New York: Currency Doubleday.

Tsoukas, H., & Chia, R. (2002). On organizational becoming: rethinking organizational change. *Organization Science: A Journal of the Institute of Management Sciences, 13*(5), 567-583.

Software Productivity Center. (1999). Starbase and software productivity center announce integration of requirements management and project estimation. Retrieved 11/27/2004, 2004, from http://www.spc.ca/about/pr990803-tbi.htm.

Strauss, A. & Corbin, J. (1998). *Basics of qualitative research* (2nd Ed.). Thousand Oaks, CA: Sage Publications.

Streubert, H. J., & Carpenter, D. R. (1999). *Qualitative research in nursing*. Philadelphia, PA: Lippincott.

Urquhart, C. (2001). An encounter with grounded theory: Tackling the practical and philosophical Issues. In E.M. Trauth (Ed.), *Qualitative research in IS: Issues and trends* (pp. 104-40). Hershey, PA: Idea Group Publishing.

Wegerson, P., & Williams, R. (2002). *Mini CMMI*. Pittsburgh, PA: Cooliemon, LLC.

West, M. (2004). *Real process improvement using CMMI®.* Boca Raton, FL: Auerbach Publications.

Whitten, N. (1995). *Managing software development projects.* (2nd Ed.). New York: John Wiley & Sons, Inc.

Wiegers, K. E. (1999). *Software requirements.* Redmond, Washington: Microsoft Press.

Appendix A

Initial Interview Protocol Questions

1. Tell me about how you decided to work in the area of software and systems development.
2. How did you learn your craft?
3. What are the enjoyable aspects of your work in software and systems development? What aspects would you change?
4. Some believe software and systems are about coding and technology. Some believe software and systems are about people and process management. Some believe it is a mix of all four. How do you feel about this?
5. Of the projects you have been involved with, have you worked within a process maturity model? If no, how do you feel the use of a process maturity model could have impacted your work? If yes, explain how you feel the process maturity model impacted your work.
6. When processes are implemented are they followed? Please explain.
7. Of the seven process areas listed in CMMI Level 2 – Managed, which do you believe have or has the most impact on your work? Describe what you believe that impact is and how it affects your work.
8. What issues do you see in attempting to implement process models within an organization?
9. If you could make changes to the way software and systems are developed in a project you have worked on or currently are working on, what would you do?

10. Describe the ideal work environment for your job.
11. What, if anything, additional would you like to comment on regarding this interview.
12. Do you have suggestions on questions that may help this research?

The Seven CMMI® Level 2: Managed Process Areas Card Used in Interviews

Requirements management (REQM). The purpose of requirements management is to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products.

Project planning (PP). The purpose of project planning is to establish and maintain plans that define the project activities.

Project monitoring and control (PMC). The purpose of project monitoring and control is to provide an understanding of the project's progress so that appropriate corrective actions can be taken when the project's performance deviates significantly from the plan.

Supplier agreement management. The purpose of supplier agreement management is to manage the acquisition of products from suppliers for which there exists a formal agreement.

Measurement and analysis. The purpose of measurement and analysis is to develop and sustain a measurement capability that is used to support management information needs.

Process and product quality assurance. The purpose of process and product quality assurance is to provide staff and management with objective insight into processes and associated work products.

Configuration management. The purpose of configuration management is to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits.

Appendix B

Categories and Coding

11/20/06

Participant ID	Participant Data	Researcher
Number	101	
Name:	**Implementation Concerns**	
Coding	Open	
Date	10/12/06	
P719	What the software life cycle was, you know, what the different phases were, what the different reviews are for	Most developers enter the process improvement world with little or no model experience. Schools tend to teach "coding" instead of software process "management' Memo:1004
P719	I think that you'd end up doing more work just to do process stuff than you're getting output.	A common complaint on process improvement is it gets in the way of getting my job done. Why should I spend all this time managing process and get farther behind. Just let me code! This is an issue that must be dealt with regularly and consistently. It
P719	they were assessed level 3 CMM and that was a good level to work at.	CMM Level 3 is higher than this study is scoped. However, it is important to note that this is a positive response to the work environment. Where the goal of CMMI Level 2 is to "just manage" the development activities, at Level 3, the processes are furt
P719	you know the customer and the users have to buy-in too "yeah", that's what the requirements are, too	The concept of buy-in - agreement on the requirements is central to the success of a system and one of the areas that is frequently overlooked.
P630	that's a tough thing there if employees have never worked under process before.; And you've seen	The repeating theme of "How am I going to get my work done if I have to
P630	how you prove that to people without actually implementing it is the tough thing; There's not enough information out there yet to say that it works.	So we are trying to sell an unproved concept, to people unwilling to change, with little or no support from upper management. Is any of this
P629	[Avoid process?] You know, because people don't like paperwork	One area that seems to lack in process implementation is the administrative support of those individuals performing much of the

Participant ID	Participant Data	Researcher
		Process documentation and management of the documentation is critical to the success of projects - it has been shown re
P629	rying to apply that process model to ongoing projects to stuff that's already been started;	What is the impact of attempting to apply a process model to an existing project? Can it even be accomplished? Or, once a project is out of the gate - is it just too late? In a perfect world nothing would start
P629	for one thing, I would definitely recommend the process models;	The focus on issues these people deal with on a day-today basis brings to light the need for a standardization of process in the work being performed. Wild gunslinger development just no longer works - like the
P628	If you would have implemented that same model in a more modern facility I think it would have worked without a hitch	***Some organizations are not yet structured to have the benefits of a formal process - much of the work is piece meal, ad hoc, and not profitable enough to make the investment for the long term - or,***
P5150	The discipline is not followed until two-weeks before the audit; And we're going to crash course it	SPI as an afterthought - or let's just get that box checked on the contract! Memo:1122
P5150	I think that the hardest issues that you run into when you implement such a process is how do you get people to stop trying to fight fires and start with a framework	***The "rush" of being the hero! Running in the burning building and saving the children and appearing on the Six-O'clock News. That fire fighting mode becomes a narcotic for many otherwise very good programmers. Eventually, they work extensive hours, beco***
P5150	[Changes you would make?] I would go back and start with that discipline; I think it needs to be adjusted and tailored for the organization in the project	***Developers like structure, but freedom to do their work - Theory of Software Development process? Memo:1126***
P42	they are all intertwined - its hard to separate any of them	***Theory of software development process - although each of the 7 process areas may appear to be separate, at all times, they are all inexorably linked to each other in some direct or indirect manner Memo:1138***
P147	adequate consequences, adequate, proper consequences are probably something that are missing a lot in some some process improvement	There should be some type clearly understood consequences for those that follow or do not follow the organizations processes Memo:1162
P122	when I wrote the CMMI process for umm, the previous company that I worked for, it was all based around just general process, you know, how do you implement a new project	Process fit under the model and satisfy the specifications within the model Memo:1185
P11		

Participant ID	Participant Data	Researcher
	I believe in all that stuff I think those are good things to practice and good models to follow	Models allow processes to be placed within a development organization
P18432E	Maintaining the processes and procedures for each PA of a particular process model	Implementation strategies - an ongoing process in and of itself - it is never done, must be always vigilant, and always promoting its value to everyone on the team Memo:1226
P18432E	Implement process models in conjunction with people management with buy-in	Plan for success - it doesn't just happen Memo:1227
P44261E	many were resistant to the idea since they felt the paperwork was overwhelming;	Process rejection excuse 101 - come up with something new, please. Theory of Change Adoption Memo:1228
P45987E	I attempted to implement CMMI level 2 practices on two recent ones for which I was the IT project manager	TMC - If attempted without management, probably not very effective Memo:1240
P45987E	Processes are inconsistently followed	If the company does not make people adhere to processes, then who will? People will only work on what they feel the company is directing them to work on. If they set off on their own, it is dangerous waters with sometimes sea monsters and icebergs - Memo
P80327E	technology is changing at such a fast pace that present day	All the more reason to have plans in place to manage it Memo:1260
P78952E	Implement CMMI	***Implement something - CMMI is just one model - any model will work if followed. The answer is not magically hidden in some great model - the answer is in the organization following the management driven implementation of process improvement - the Theory***

Number 102
Name: **Process Hierarchy**
Coding Open
Date 10/12/06

P719	I still think CM and requirements management Process and product quality assurance is another	1) CM 2) REQM 3) PPQA - Perspective from a developer. Project Management and other higher level project process needs are frequently
P719	requirements and CM are just really important.	Restating the hierarchy with requirements first this time. Memo:1013
P719	If you know what the requirements are, then you can work on how do you actually implement.	Developers always are searching for clear requirements. It is difficult to build anything when you are just giving your best guess to what it is.

Participant ID	Participant Data	Researcher
		the uncertainty - iterative and spira
P719	If you don't know what they are, I mean, you're kind of trying things and then you find out, no that	Again, not knowing what the customer wants creates enormous frustration for the developer Memo:1015
P628	The first one [Requirements Management] greatly impacts what we've been doing because	Must understand what the customer wants - requirements to the software developer is only connection many feel they have to the customer. Unfortunately, if not managed to a level of design and architecture BEFORE being passed on to the developers, the req
P628	If the project would have been planned better the requirements would have been managed better ;And the whole scenario all the way down to, to the person doing the testing	***Lack of planning is a certain way to make sure there are complexities in the project. The planning portion is the key to success. Each individual process component, requirements, etc. are very important, but the overall plan is what really brings it all***
P5150	Requirements management; what are we doing? Configuration Management: What have we got?	The two key software developer concerns Memo:1120
P42	requirements management and configuration management is very important as well	Software developer perspective - the two CMMI Level 2 process areas that they are most familiar with. Memo:1136
P42	measurement and analysis	***Something that is frequently overlooked by the developer - whether they know it or not, there must always be some kind of measurement and analysis happening during the software development life cycle. Each time a section of code is tested, the***
P3	Project Monitoring and Control.; Especially in the context of new processes trying to be implemented amongst a software development team	All of the 7 CMMI Level 2 process areas are vital to management of a project. That is the first step to trying to get the runaway train - ad hoc development - slowed and under control again. Memo:1150
P3	But this was definitely a case where, I think those two; Managing Requirements and Project Monitoring	Any pick for any reason is always correct - since managing all is the only true option Memo:1154
P147	requirements management - right now I'm doing project scheduling so full requirements management to identify those	The different process areas support and feed other process areas - nothing just stands alone or on its own. A project is a huge interaction of people, components, and time - all using all the process areas at one
P147	Project Planning , yeah, that's pretty high on the list; Monitoring and control ; that goes right along with Project Planning	Can we really get there without a plan? Memo:1164
P147	Process Product Quality Assurance, I think	PPQA really is the glue to hold everything else together. IN CMMI, it

Participant ID	Participant Data	Researcher
	that's a high impact	insures the quality of every process, every deliverable. It is the reality check for the process universe Memo:1165
P123	I would say project planning is probably important; Requirements management sounds really important in a sense of getting your requirements nailed down	Project planning is the absolute overview of everything that is and will be happening - without this highest level view of everything, where would one start? As discovered under the Theory of Software Development
P122	[Ranking of process] I think they all have impact	Yes, they do Memo:1190
P11	of course requirements are important, because without requirements you don't know the boundaries of what it is you are being asked to do; obviously a certain amount of planning has to take place for anything to be successful	Yes Memo:1196
P102757	The very, very first one obviously because it is close to my heart is requirements management; process and product quality assurance, and configuration management;	A common developer perspective. Memo:1206
P11637E	as a general rule, I believe good requirements produce good solutions	One piece of the puzzle - defining what makes "good" requirements and what vision these requirements are attempting to address are also
P16238E	Project Planning, configuration management and Measurement & Analysis	The list Memo:1217
P18432E	Requirements Management managing requirements, and change to the aforementioned list, Project Planning - an idea of when things should progress, Project Monitoring and Control this provides project monitoring f)Configuration Management	The list Memo:1224
P44261E	As a project manager it is the most critical process area that forces you to think through the fundamental elements of a project; As part of this process you are defining schedules, resources	Part of the process areas listing form the project manager perspective Memo:1234
P45987E	As a project manager, most do; REQM, PP, and PMC standout the most; Requirements feed planning, and planning is an essential task of any project manager	The list Memo:1250
P80327E	Project Planning Less surprises for Staff; Process and Product Quality Assurance Customer	Working from a plan - we'd never think of skipping a plan on a $200 million space flight (except maybe we should've made a requirement

Participant ID	Participant Data	Researcher
	Confidence; Configuration Management – This helps maintain release schedules and maintain software	that all measurements were in meters and not feet). When things go wrong, start by looking at the plan (if you can f
P78952E	I believe that Process and Product Quality Assurance is the most important process area. If a project adheres to the process and product quality assurance plan then all other process areas will be maintained accordingly	Quality - If this is truly satisfied, what else is there? Memo:1263

Number 103
Name: **Process Acceptance**
Coding Open
Date 10/12/06

P719	Many organizations that I worked for that had right, is CM - configuration management.	Almost by necessity, if a software development organization does not at least have good configuration management, they quickly see how
P719	I've seen yes and I've seen no. I'm going to leave	Process implementation is difficult to attain in an organization that is not fully committed to an implementation. Many times processes are implemented, but the troops do not have the training to follow, the desire
P719	how do you change your requirement? How do you add requirements? How do you delete them? how do you decide how you're going to it and	Lack of planning for the development environment will always become evident at the software developer level. Memo:1019
P719	you just can't code it because it doesn't work....you actually have to test it and make sure it's doing what you're expecting it to do.	Guessing at code, then running without testing...the code and fix development cycle that rarely produces good code and usually costs a lot in development time and dollars Memo:1021
P630	Because once I had been through it several times, I knew we were doing the same things over	***Whether anyone wants to admit it or not, processes are always followed - just not in a formalized manner. Each individual, each team, each project, each organization, all have "ways of doing things" which could be noted as a "theory of process" However,***
P630	because it wasn't formal, no one else could have taken over those projects and had the same success as easily	When processes are passed among people without any formal management, there is always the risk that the process will eventually be lost. In the event that the knowledge leaves the company, which
P630	So we were, you know, all executing to the	***Theory of team process - it happens whether or not management***

Participant ID	Participant Data	Researcher
	same informal process.	***pushes for implementation - so why not help those executing the processes learn the vocabulary and syntax of those processes, document for all to know and understand. Memo:1039***
P630	Well we always did the same thing.; grabbed this template we had for an application; developed all the databases very similarly; reusable login components and things like that; So you'd always start at the same place	***Process exists without the model. As a development organization matures, processes are created, regardless. There is no need to have a model to have process - it will naturally occur in the development environment. Applying a formal "process model" is***
P630	it was definitely a process that we followed and enabled our team to repeat the same kind of work	Obviously there was a fairly well-developed process in place.
P630	We had common pieces we always pulled together; I didn't quite realize until I'd been there a couple of years that we really did have a process	***Does a process need to be formalized in order to exist or be followed? Apparently not. This organization was already on the road to process maturity even though it was not officially documented. Memo:1046***
P630	when we did like ISO 9000 over there.; you know, were happy to see something like that kind of	***People may fear change, but they can appreciate structure - especially if you have been working any length of time in an unstructured environment. Like anyone else, software developers feel more secure and a better sense of direction when there is a***
P629	I think that somebody that been following the CMMI model over a long period of time, no matter what projects you're going to throw them on, it's going to be more apt to jump in and just follow	***When implementation has been successful and the team follows standard processes, the work becomes more quickly coordinated and responsive to the needs of the project. Regardless of Maturity***
P629	if you have people that are, you know, used to CMMI, I think they'll be more apt to follow the model process knowing that was actually aiding them in the end	Institutionalization of process and model - it is now part of our corporate
P629	configuration management was really, I thought one of the most important ones -- just because as you're working on stuff, if stuff changes in the middle of your project, it throws everything off	Sees the value of managing the work one is doing. Amazingly, a lot of software is lost due to power failures, accidental overwrites, changes that are made and not saved, and so on. Many of these things occur because there is not good management of config
P629	By gathering those requirements you actually know where to start with your entire project	***Established process - setting up steps to good development – gather requirements first. Regardless of the correctness of the statement, this shows the process thinking is happening even if they are not***
P629	You ought to try to actually get stuff under	If you are not following a process, then you are destined for problems.

Participant ID	Participant Data	Researcher
	configuration management and make sure	Memo:1083
P629	I like the idea, to answer how we were trying to do CM on the changes that we were doing as developers patches and so forth and printing out	Once people work together with good communication and configuration of their work, they find that their jobs become that much better and even
P628	something like twelve millions lines of Cobol code; there was a definite procedure in place for every aspect of every deliverable	Apparently, for the large scale project, this organization understood the need for structure Memo:1086
P628	The point is there is a procedure and we were following the procedure	***When it is all said and done - in any team environment, the dynamics of the individuals make up the results - Theory of team interactions? Memo:1091***
P628	And when it's followed correctly it greatly helps.	***Process followed makes for a good work environment - Theory of software development process Memo:1092***
P628	the large-scale scheduling system, everybody there DID follow the process; and it worked well	People in a development environment know when things are working well - and when they are not Memo:1094
P628	They had, uh, they had met with not only just the management level but they also met with the end-users	Success in requirements elicitation Memo:1099
P628	But they were locked down and they never did change the whole time; I think the project was about 5 years	Absolutely no change may be drastic - after all, as projects progress, the focus and view changes. Change is not bad, it just needs to be planned for and managed effectively - Lock down 5-year-old requirements may prevent some development confusion, but
P5150	same way as some people think about diets if you take, you know one crash diet, then go back to your old ways, know its not going to work – it's a discipline – a way of life	***Good analogy - people start and stop diets - they also start and stop adhering to process models; if you do not teach good eating habits initially to children, they eventually go back to what ever eating***
P5150	I've also never seen anyone arrested for J-walking if its not enforced, you know, at least its not	Simply stating process is in place will not make it happen - must be managed continuously at first until such time it becomes corporate culture Memo:1119
P42	we had this process we kind of made up; it wasn't based on the CMMI Level 1 or 2, we just made up	***Theory of software development process - there must be a process*** ***when any one begins developing software, especially if there is a team. It is a naturally occurring phenomenon. For software***
P42	we'd have formal documentation of everything for the customer, as well, for any changes that were	***Documentation - part of their own process Memo:1130***
P42		

Participant ID	Participant Data	Researcher
	That was the first place I'd been where they used the government's MIL-STD 498; when I got there the procedures had been well established and I would say, that you know, it seemed, at the time I complained; of course you complain, but you get used to it	Government has pushed for standards in SW development for years. A typical reaction when entering a process or structured environment is to react with complaints, but once acclimated tot he environment, it just
P42	you have to learn it, its basically something that really helped, I mean I can look back now and see how much it really streamlined everything	After a period of time in working under process structured environment, people can see the benefits of not having the same kinds of work hassles they dealt with in an unstructured environment. Memo:1132
P42	[Processes followed] From my experience um, yeah, well I work in the government sector so, a, we have to follow	One good thing about some government environments is that if you want to work there, you will follow their processes. It is structure not unlike
P3	often the process maturity model was an afterthought to most of the people on the software team; We had decided to adopt a process maturity model, and we sat together and white	***In motorcycle racing, the word is that if you have to think about passing the guy in front of you, it's already too late; if you have to think about process implementation because a project is having problems, it's also probably too late***
		The team knows
P3	takes a very meticulous, very well-organized, and very well-respected manager; especially if it is new processes and you've got people who have a comfort zone	Almost drill instructor level drive is needed to make the troops adhere to process
P147	the folks that were operating that company that I reported to at the time were familiar with that and they kind of adapted my goals to how that was going to work out for them	***The first one to step up to the process leadership plate will either get pushed into the arena (without a sword) or just immediately shot Memo:1157***
P147	[Impact]it was a much more effective result at the end	Those that truly adopt process improvement see unbelievable results after awhile - plus they actually get to meet their kids again each evening Memo:1158
P147	but if it is not explained to individuals, if the goals aren't, if the expectations aren't laid out, um, it's a bad thing	***You have to mean it - people quickly see through smokescreens and big talk - if you do not back up your process demands with training, money, and support, it is obvious it was never really meant to be Memo:1160***
P147	at the same time you have, um, if you have policies and procedures that are part of processes that are there, an integral part that has to be consequences	When people know their jobs, raises, reviews, etc. are dependent upon their following the processes in place, they take a different view of it rather quickly; good process improvement leaders must stand up to the customer, management, and the employees -
P123		Has model some model experience Memo:1172

Participant ID	Participant Data	Researcher
	{Involved with process models] So I guess I would say that [yes]up until this most current project	
P123	I think I've never seen processes that are 100% followed	While 100% would be a goal to achieve, what organization could possibly achieve it? But an organization of 90% following everything 90% of the time would be deemed very successful. It is the 0% and total lack of any process (which is impossible - even de
P123	First place I worked, we worked producing software that was actually, we actually had customers and so we had at least to do releases on a regular	***Regular releases and bug fixes - Theory of Software Development process - it existed in some fashion, but apparently not under a formal model Memo:1174***
P123	I would say project planning and I think I would stick with that; I would characterize myself as a person who likes to know that things are well planned and thought out and I don't particularly like to have things re-planned a lot	We all like to know where we are headed Memo:1178
P122	Then companies where the process is already written and you just came into it and you you learn the process, umm, but in other companies there's been no process	Best case - coming in new to a company with established process. "This is the way we do things around here!" Memo:1187
P122	so you had to create, create the process and its always more helpful if you have the people who are actually going to be doing it involved in the creation of it	***TSDP - the software development team decides to create the process in the organization that is missing one. Memo:1188***
P11	Mostly, I've found, that they are loosely followed	Processes will be followed at the level of importance an organization communicates to the follower. Theory of Management Commitment Memo:1195
P11	But you can control that by monitoring the project and controlling the project to make sure people, even though you don't have a deadline to reach, they're still making progress toward satisfying the goal	***Setting realistic goals on tasks is ultimately the responsibility of the project manager. If they have the technical expertise to do it themselves, they should; if they do not, they need to have the***
P11	I'm a strong believer in data driven software development	Yes - spoken like a true database analyst! Memo:1200
P102757	not a very mature one; I think significantly.I think significantly.like um, like with the first major system I did it was really the advantage was in	At least there was an identified process - the first steps in conquering ad hoc development craziness! Memo:1201

Participant ID	Participant Data	Researcher
	the military we have SOPs for everything	
P102757	you have to sell that to the people who are going to be affected by the change	***Theory of change adoption*** ***What is in that sell? Memo:1203***
P102757	Requirements driven scenario based modeling all	A methodology that constantly looks at the application through the lens of scenarios - useful in clarifying requirements and managing the project
P11637E	yes;In order to establish implementation consistency and predictability	Understanding the necessity of process Memo:1211
P11637E	Allow more active participation by all stakeholders, including the people who are going to use the solution, not just management	The more a team approach is fostered, where each project member - contractor or customer - feels some ownership and some responsibility for the success of the project, the more like it will succeed. When people feel out of the loop and ineffective, they
P16238E	I have not formally worked within a process maturity model on projects, but worked successfully in many informal ways on projects	***Theory of Software Development Process - had to be doing something (some kind of process) but did not know the specific name Memo:1215***
P18432E	I feel the process maturity model brings a desired order to the work job; People know how and when to carry out their job functions, what the expected	Isn't your job much better when you know what you are supposed to do? Memo:1222
P44261E	I've discovered that they are not always followed; often times little components of the process are forgotten or never meet actual implementation	***There are those that will always try to circumvent - if it is planned for, then their circumvention becomes actually part of the process,*** ***and therefore, no longer NOT an issue, just part of the way the process works*** ***Theory of Change Adoption***
P45987E	implementing aspects of CMMI particularly PP, PMC, and MA helped keep the effort organized and this was noted by the staff	***Another success story -*** ***TSDP Memo:1242***
P45987E	It really is crucial to have a good handle on requirements; both the client and our own senior management, will continuously come up with requests	***Might as well have a good handle on all that as being done - someone will ask anyway - good reason to adopt process.*** ***Theory of Change Adoption*** ***Memo:1251***
P45987E	Put greater emphasis on requirements	All possibly good fixes - it is the implementation of these processes

Participant ID	Participant Data	Researcher
	management and requirements development; Use prototyping tools where called for in the design process and conduct quick verification sessions	the constant application of them in a change resistant environment that is the real trick Memo:1254
P80327E	I feel that the weekly meetings have had one of the greater impacts on keeping the project moving smoothly	Communication - project glue Memo:1256
P80327E	Yes, At least on the projects I have been on; Weekly meetings and status reports have been met; Plans are maintained and followed; Software issues are recorded in a tracking system	Lets talk about it! Memo:1257
P78952E	A process maturity model would ensure consistency in developing and maintaining the software, and make modifications easier and cheaper to implement over the lifecycle of the product	Yes, another convert! Memo:1261

Number 104
Name: **Change Adoption**
Coding Core
Date 10/12/06

P719	you know people always have ways they like to do things.	***This is a huge hurdle in the implementation of process model. People have worked certain ways, sometimes for many years, and have usually figured out some way to achieve success. Approaching them with a new way of doing things - even if it is not***
P629	when we learned, when we were doing that, starting it initially was kind of a headache; You got to learn all that stuff. This isn't going to do me any good	At the beginning of new things (change) it is always slowest and most confusing. Memo:1070
P629	if you're not strict about making people follow the process, I don't think they will	In instituting new change, it must be reinforced repeatedly if you wish to people to take it seriously. If they feel it is just lip service - then they
P629	I think that the model needs to be followed, even though the people may kind of balk at it and	***The theory of process in software development - it is obvious that people will have to concoct some type of process to complete***

Participant ID	Participant Data	Researcher
	especially being something new	***work, why not simplify it and follow one that already exists? Memo:1082***
P628	here there's a real, there wasn't one until we ended up trying to implement one ourselves; once everybody found out that we had implemented one, they did everything in their power to	***Self implementation of process - Theory of process Resistance to change even on internal development of process - personalities came into play - quoting participant:*** ***"there was once a conversation where there was something that was broken, that I knew***
P628	[Process]was working good until people found out that we actually had a procedure; They tried to	***People will resist change to the point of sabotage - project suicide - in some environments, it is the continuation of the project for the contractor that is the goal of the management - long after they***
P628	our immediate supervisor and the manager above him were involved in all the meetings and once it was found out that we had our own procedure in	If you intentionally break the communications, you can show to your manager the failure of process - sabotage and keep your job! A hideous
P5150	rarely someone says "If you had the last, you know, if you could do this all over again, starting here, how would you have done it? I think this might fall into the issues you run into with implementing a process; nobody tends to think in that mentality	How many times can you get paid for doing the same project? Some contracting companies are experts at this. Ask the FAA and DoD. Memo:1128
P3	then everybody went back to their desks and it seems like they all just wrote code and programmed and, and delivered the way they always had	It is so easy to start eating the wrong foods again, stop exercising, have just one more cigarette - the world of changing is filled with fallbacks
P3	my experience with a process maturity model was disappointing because I didn't see people change	If no one else is going to do it, why should I? Memo:1146
P3	you need a project manager who can ensure compliance from the project team and that means	Until it becomes part of the corporate culture, the battle for change is always "online" - there is no rest from pushing for improvement. Memo:1148
P3	when one, who from almost day one could not establish credibility with the project team –and, people refused to follow the process he was trying to establish	The road is littered with failed project managers - frequently they hold signs saying "Will code for food" on the side of the road Memo:1149
P3	Getting people to step out of their comfort zone	Never underestimate a person's willingness to fight for their comfort

Participant ID	Participant Data	Researcher
	the way they've always done it	zone - nothing is more feared than the unknown Memo:1156
P147	Recognition of impact of existing processes and changing those processes on employees is huge, and then, and then, working that issue in the right, appropriate manner, um, I think that a	Change management - not an easy task Memo:1166
P123	Everyone knows it's a pain in the neck and everyone tries to circumvent it because it is difficult, but a lot of the problem is, too, some education can help in terms of people realizing that it is important	The extra steps of notating what is being done or following certain steps in doing it always presents and opportunity for conflict or rejection by
P102757	when a new piece of equipment comes in, they feel threatened because their expertise; they're kind of on the same level now so implementing these processes could be the same way; get people	***Theory of Change Adoption*** ***Change is frightful to many and they will do most anything to avoid it. Memo:1205***
P102757	Yeah, you know its, its, reluctance to change; Not committing enough resources to support the change	***Theory of Change Adoption*** ***Management wants things to be better, but they too, do not want to*** ***change the way they operate. Process improvement takes no prisoners - change happens to all, everywhere, and that change***
P3	getting them to step out of that constant frame of reference they have had and look at things from a	***Adapting to change in process Memo:2155***
P16238E	There is typically less motivation to follow process initiatives if presented as a second job in lieu of presenting it as part of a practitioners job	***Theory of Change Adoption - Rather than presenting someone with*** ***an extra job, why not help them understand the vocabulary and steps they are already using under the Theory of Software***
P18432E	Processes are implemented to the extent that those involved in their implementation have buy-in	***Theory of Change Adoption Memo:1223***
P44261E	has to be an organizational commitment by all stakeholders to really get the commitment level necessary to fully implement processes	***As Lowe's advertises, "Let's build something together!"*** ***Theory of Change Adoption*** ***Memo:1233***
P44261E	Staff feeling overburden;unless you first hand experience the advantages of process	In an ad hoc environment, most everyone feels very overwhelmed much

Participant ID	Participant Data	Researcher
	improvement chance are you will be resistant to the burden of documentation that process models	doing creates a lot of resistance - maybe rightfully so - process implementation needs to be seen as a solution to th
Number	105	
Name:	**Education**	
Coding	Open	
Date	10/12/06	
P719	We had some quick corporate training about what the process was,	Companies seem to take education as something to quickly throw at an employee so they can check off the "trained" box. Process improvement training tends to be reading the book to employees or
P719	If you don't know what you have, you don't know how to follow it	Unfortunately, because training frequently is just minimal and performed
P719	a lot of educational issues - Trying to get people to understand what a process model is.	Can't expect people to understand without being educated Memo:1023
P719	I've seen organizations and they have team building exercises and things like that. Sometimes it works and sometimes it doesn't.	Taking communications seriously in a company is sometimes looked upon as just nonsense; it is the smart organization that does take it seriously and continually attempts to improve corporate and employee communication. Memo:1033
P719	that's probably where it's having the right people in the right places that's really important; People that can, that are good at facilitating communications.	Full time working on communications never hurt any organization - if we can not work together, then the company becomes a series of stove
P630	something I learned a couple years later, that there really was some process in place.	***People are surprised to learn that they already are performing processes; they just hadn't formally named them yet. It is difficult to perform without a process of some sort - so by tweaking and tuning to fit what an organization is attempting to move to***
P630	It wasn't transferable to anyone else and a new person coming in couldn't get up to speed as quick because of that.	When process is not communicated 1000 fold - how can individuals in the organization be expected to understand what is to be accomplished? Memo:1040
P630	But it definitely made the work that we did over there possible and the turn around time that we were given.	If it were not for the informal process in the development of their projects, the work would probably never be accomplished in a reasonable timeframe Memo:1041
P630	first thought of the employees at the bottom are	When people performing process already are approached with the idea

Participant ID	Participant Data	Researcher
	"this is going to be a waste of my time." ;if they don't see management taking seriously	of process must be followed, the immediate reaction is this will take more time - even though they already may be following process - it just is not documented as process. Is this manage
P630	It seems like there's a place there where there should be some mechanism for training	Training of personnel on the project tends to be frequently overlooked - it
P630	I've heard other people say that DoD does this kind of thing to educate ahead of time	Training everyone involved is a component of the CMMI generic practices Memo:1055
P630	Some of us who've gone through courses and stuff where we've been sold on the idea that it's important. We're willing to buy-in a lot quicker.	***Education appears to be critical in the development of process implementation. If people do not understand the vocabulary and structure of a Software Process Improvement strategy, how can they***
P630	I think as the projects get bigger, documented process becomes more important and training for new people; You can't just have documents sitting out there and say "this is our process;	Smaller groups have been able to frequently work closely with a customer and communicate effectively among themselves and this is a real world experience that many developers have had. Therefore, they believe they can continue to develop the same way - u
P629	nitially I think that it was, when learning the model is the slowest, most aggravating part; But once you learn the model and how things work through that model, I think that it actually does aid in the actual development process from start to end	Education or learning of process is difficult for those not trained as part of the software development skills. While the theory of team process may account for all development following some coordinate process effort, the lack of formal process that dev
P629	if you had someone new on a project or new to CMMI, I think that's where you're going to encounter your slow downs	Lack of experience - needing education - will slow down that portion of the team Memo:1069
P628	[Changes I would make] Make sure everybody really and truly understood what a prototype meant	Here is a a famous trick of customers and managers: We are not quite sure of what we want to build, so please code us a prototype (some times also known as Demo is suggested) and we promise we will not use it for our production system. Developers take m
P123	people would have the right education as far as what's the importance of it guess I would make the analogy to say security and passwords, and things like that.	Mandating change without providing training is just nothing more than a waste of ink, electrons, or hot air....people either have to come to the project with the right education and understanding or it has to be provided for them as part of the project ramp up
P122	understanding the life cycle of a software project is more, I think, more, even more important than understanding the capability maturity model	These are one of the same - if the model is understood and followed, one will naturally be lead through the lifecycle of the project. Conversely, if one knows the lifecycle, there is no requirement to step through it in
P122	any new implementation or new CMMI	Training, training, and more training. If you want people to be able to

Participant ID	Participant Data	Researcher
	need to really be um, the group needs to be educated	perform at certain levels in certain ways, you need to either a) hire them fully trained that way (if possible?) b) train them - continually - in the
P16238E	Educate customer as appropriate on process improvement	Part of the job - always Memo:1219
Number	106	
Name:	**Management Commitment**	
Coding	Core	
Date	10/13/06	
P719	Management, I mean, there has to be management buy-in, they're not free.	Improvement costs $$$ Memo:1025
P719	they take time and money to implement and so that's where management buy-in becomes very important.	Initiating process improvement in an organization is a very time consuming and costly endeavor. It is not to be taken on lightly if a company wishes to have any chance of success. In fact, the greatest waste of money is making a half-hearted attempt to
P719	They [management] have to be willing to put up the time and money.	Management commitment is much, much more that the managers saying they would like something to happen. When any of us commit to something, we provide the opportunity for to succeed - which means people, company resources, money, time, support, and an ent
P630	But if they didn't 'have confidence that management thought it was more than just trying to get a plaque on the wall. ;It wasn't seen as something very useful to the underlings	Many companies approach process improvement as a means to get more business and a plaque on the wall. While going through the implementation of process improvement definitely helps the software development environment, if it is perceived by management as
P630	gotta have manager buy-in; that they see the process is important; Not that they are concerned that they have outside people knowing they have	***In any organization it is extremely difficult to push improvement up to management; it is always a matter of needing to be pushed down***
P630	employee buy in comes after that; I think is a lot more achievable if the managers were really sold on it	***Without the direction of the management the employees will not have the need or desire to establish formal process, only informal to the level that it will help them achieve their work goals. Memo:1059***
P628	[People followed process] Honestly it was management; management positions that put their foot down and said, Okay you're doing this way	***Only management can enforce the following of processes - if it is your job or follow a process, most individuals (no matter how grudgingly) will follow the process. After a period of time, the "following" becomes nearly transparent as it becomes more cor***

Participant ID	Participant Data	Researcher
P628	everything worked out well and down here if anyone ever, if our immediate supervisor wanted to do it, the people above them tried to circumvent it and so it just didn't work	Management actively interceded and prevented the development of process - the active destruction of a project Memo:1096
P628	management excluded the developers; one guy who was the big man, he was excluded from the meetings; kind of hard to understand why a supervisor was excluded from the meetings there was some head knocking there	If you intentionally break the communications, you can show to your manager the failure of process - sabotage and keep your job! A hideous perspective of some of those in power (as above) Memo:1108
P628	hard to understand why a supervisor was excluded from the meetings	If management does not have full buy-in and does not have their act together, there is no way that the developers will be able to deliver a successful product in a reasonable time. Management in-fighting completely destroys a project - any hope of proces
P5150	I see a lot things, I see a lot of overhead in the discipline that, yeah, if you got that , it would be a good thing, but what's the, the return on investment of building such a discipline	***From a purely business perspective, implementation of software process improvement looks to be very costly, and for most senior managers the feeling is there is little return on investment (ROI). Process improvement costs are frequently short-changed; org***
P5150	I don't think people see that return on investment; I mean people get preached the return on investment from such a discipline as what you just showed me but often times some valuable piece of information gets lost in the I've got to get	The effort to better track, better record your efforts, monitor your code for errors, follow a structured coding standard, spend that little extra time on reviewing and understanding specific requests - all of this is rarely seen as increasing value and p
P42	In order to do it, its got to come from the top down;ts got to be something enacted from the highest levels down and enforced	***Theory of Management buy-in - for something to truly happen within an organization, management "has to" buy-in on the concept and enforce it. Process lip service is nothing more than process lip service - commitment to the implementation of software***
P42	it would be a lot more efficient if you could use specific tools in my experience and there's never any budget seems like that's always the short	$$$ - Not providing adequate dollars for certain work functions is one of
P42	but they were too stingy to buy that that's like 8 hours of developer's time and yet we spent 100's of hours doing these bug fixes and manually checking the bugs for bugs	The old saying penny wise - pound foolish - plenty of opportunity for the
P3	You got to have a clear strategy and strategies	***Strategy - plans - processes - they all overlap at one point***

Participant ID	Participant Data	Researcher
	come from leaders	***- aren't we really just saying we need to know where we're going, how we are going to get there, and what we are going to do when we arrive? Memo:1155***
P147	we were able to get management's buy-in quicker, we were able to um, better explain the business case for the work we had to do which got the appropriate amount of funding we had to have in a timely manner	When things are working well, it is easier for everyone - too bad there is so much fear fighting against some simple process improvement
P147	when I've been involved with projects that had buy-in from senior management ;they saw the problem they were willing to work the problem ;they were willing to take ht expense hit	***Management that is responsive to the needs of employees and the project will always have more success than management that wishes to stay a few feet away from all the issues. Somewhere a long the way, someone has to basically hang it out there in front***
P123	president would come down and say, this is the thing that you're going to do because that guy wants it done or something like that; cusomter driven circumvention	***Management has not fully "bought in" to TSDP if they cancel process as a means of immediate acquiescing to the customer demands Memo:1176***
P123	I think you need your management buy-in, you need, not just management, but you need your whole team to agree that that's important	***Management must drive TSDP Memo:1179***
P122	Management's always a problem; Management has to buy-in to it	***No TSDP if management doesn't write the check - the best demonstration of management buy-in Memo:1191***
P11	Anytime you're going to do something like that you have to make sure that you have buy-in top to bottom	
P102757	what determines whether they're going to be [processes adopted] is a series of factors, and, one of them is management buy-in	Back to the Theory of Management Commitment (TMC) Memo:1202
P102757	there's a cost to changing a process, so people become uncomfortable and the more tightly bound they are to the old way of doing things	***Theory of change adoption*** ***Cost in emotional ties to the comfort zone, cost in the training of new ways of doing things, cost in the loss of profit to the company*** ***while worker methods are re-tooled, cost in time on deliverable***
P102757	they don't end up doing grunt work just because	***Theory of Change Adoption***

Participant ID	Participant Data	Researcher
	there's grunt work to be done; buy-in from management down to the people that are affected and then committing enough resources to	***Theory of Management Commitment*** ***Theory of Software Development Process Memo:1209***
P16238E	Sr. management commitment to enforce process improvement.; Sr. management inexperience to manage high mature organizations; Sr. management commitment to lead by example	***Theory of Management Commitment - again, and again, where is management on this issue? The underlings want to be shown the way! Memo:1218***
P16238E	Obtain Sr. management commitment	***Theory of Management Commitment Memo:1220***
P16238E	gather measurement data(estimation vs. demonstrate Return on Investment	***Process improvement movements tend to focus on the improved practices and development environments and leave out the manger's thinking on ROI. Specific studies of cost savings or increased profits from use of process improvement strategies are slim. The***
P18432E	Buy-in from the project stakeholders and others that will be needed to implemented. If there is not sufficient buy-in from those that will implement the process, the effort will fail	We all must paddle in the same direction if we are ever going to cross the river before we hit the rapids (sideways) Memo:1225
P44261E	Get buy-in from those whose commitment is most necessary	***Management controls the money - sad, but if they are not on board,*** ***you're only wasting your time***
P45987E	In both cases having insufficient resources to manage the requirements hurt the projects	***TMC - Show me the money! Memo:1241***
P45987E	There is insufficient oversight from senior management to ensure more consistent implementation of processes;[Management] they only get involved when there's a client complaint	***Three non-process enforcement monkeys - Hear no, see no, and speak no*** ***TMC Memo:1249***
P45987E	Commitment from senior management, project management, and staff; commitment from senior management to provide adequate resources to develop the necessary processes, and to consistently implement those processes	***Senior management commitment seems to consistently come back*** ***in most every discussion on what need to happen to make things***
P45987E	Make the whole team work in the same physical location most of the time	There is much to be said about keeping a working team together to foster better working relationships - however, today's remote office work is becoming more of a reality and for many reasons, physical
P80327E		

Participant ID	Participant Data	Researcher
	Corporate Management Buy in and lack of money	TMC - Must be low enough to win- high enough to actually do it - picking the right business is the first right step in the project vision Memo:1259
P78952E	Cost barriers;	ROI - Again, doing it the right way looks expensive, that is until you see how much it costs to do it the wrong way several times! Memo:1264
P78952E	lack of upper management support, lack of client support	***TMC - It must be there and the client must also understand that when you contract someone to do the work, you are relieving yourself of much of the day-to-day decisions - part of the reason for***

Number 107
Name: **Communication**
Coding Open
Date 10/13/06

P719	[Smaller groups communication]It's easier because you don't have as many communication paths.	Small groups tend to communicate better than those spread out on large projects - this can also tend to be a level of resistance to process in that work is performed relatively well in small groups and they do not see the need for large process functions
P719	You know? Leave it to know somebody better and work with them better.	As people get to know each other better they tend to be able to work as
P719	Especially when there's other groups that you just don't communicate with on a day-to-day basis.	Remote communications to strangers does not make for good project understanding Memo:1009
P719	[In large development groups] It's harder to keep in sync.	More cooks in the kitchen Memo:1010
P719	[So things work better] when you know what you are doing	The developer feels much better about the work he or she is doing when they know it is what they are supposed to be doing. Working with vagueness in your job is a difficult task for anyone and is especially
P719	I guess just different people just communicate differently.	People may look right at each other say the same words and still not even be close to understanding what has just happened. Having requirements or design documents means very little unless they are written in such a fashion that people can precisely unde
P630	think that's sort of the starting point of where we ended up with requirements that weren't useful	Without training in advance on how to meet the customer's expectations, many are just guessing on how to go about it. There are

Participant ID	Participant Data	Researcher
	because they didn't know what was expected.	many steps in gaining agreement and confirmation, restating, a whole host of items that should be performed. Memo:1057
P630	that would be clear communication of project status all the time to employees.; No hiding of information to the extent practical; you should never be hidden from what your customer thinks about what you're doing and things like that.	***Communication has been identified several times for the past several decades as key to successful software team development. It has also been discussed thoroughly in some of the best known IT***
P629	a lot of stuff in the environment is changing and the communication between the different, between people that are gathering requirements and then actually coding the software and so	The project can only be as good as the communication between al the people on it. Memo:1080
P629	I think really probably it safe, communication between the different teams I think is extremely important.	It is vital that the teams have a fulltime, regular communication link between each other. If there is not a good means of communication then much work will be lost - either because something was never done
P628	[Using prototype for production]That's where the breakdown there between the client and management was	There is no shortcut to good software. Just because a prototype looks like it is the application does not mean that it has the capabilities it needs behind the scenes - which it most definitely wouldn't or it would not be a prototype Memo:1115
P147	I think basic communications - if you are going to do this, if you are going to go through this process improvement or implement this new process,	How little we actually do communicate with each other on a regular basis. Dependent upn the specific tasks, in a team environment, communication may need to be almost constant to at least every day or so - this needs to be part of the project plan - real
P123	We had to have a very clear division between people that weren't able to communicate constantly in terms of independent pieces of the project that could be done	Project communications are always a point to consider - whether across the country or across the cube - it is the project manager's responsibility to make certain communications are happening in an effective way. Frequently, the first breakdown in communication
P3	end of the first 6 months of that project and delivered version 1 and by, and by 3 weeks after go live not one single end user was using the product	Applications that were never used - what an expensive proposition - the
P45987E	Regular staff status meetings would also help by controlling the schedule and giving the staff a feeling that there is predictability and order	Regular reinforcement helps guide the team members to the final goal. Memo:1247

Participant ID	Participant Data	Researcher

Number	110
Name:	**Development quality**
Coding	Core
Date	10/14/06

P719	sometimes your developing new, new ways of doing things and maybe your end-user and your customer don't really know either	A frequent issue in the development of software is the customer is not clear on what they want to accomplish. The customer, in a perfect world, should be responsible for adequate business requirements and business rules. Frequently, the development proc
P719	you need to have some way to manage the change	Change control is vital to software development. Without managed change, things go haywire very quickly. Especially when the development is being handled through multiple developers - how can one
P719	[Lack of Requirements] Well that's a way you can get stuck in a rut. If you don't know what you're, what the end result.	Development quickly grinds to a halt when people do not know what to do. Requirements alone do not define development - they must be translated into design that the developer understands what to build. By saying a building shall have a front door (a req
P719	I mean you need to be, you know, using good CM	If developers are not educated to use the practices needed for good management of their configurations, source code, etc., how can they move forward? Memo:1020
P719	[Quality issue] To get what you want out. To get the product that your users and your customers need.	A need to thoroughly test code before it is declared to be finished. Memo:1022
P719	trying to come to an agreement on what we're going to do	Development should be dictated by the design of the system and not negotiated by various developers. Under a negotiation method, the developer with the strongest personality or negotiation skills wins...which means the customer may or may not win, depend
P719	what you're trying to do with requirements is capture what your customer and your users	There is a sense here of attempting to really build what the client wants. This is a consistent theme throughout the study that is clearly vocalized here. While it seems a very basic concept, it is the goal that
P630	And we always did a kind of informal requirements thing with whoever we were building	Standard requirements procedure to help with the overall quality of the work. Process in place, would have been evaluated as satisfying part of the Level 2 REQM Memo:1045
P630	on how you purchase their software so they know	Training in advance so customer expectations are met Memo:1056

Participant ID	Participant Data	Researcher
P628	the requirements were locked down and then they locked them down and then we came in and I	If allowed to be changed continually, software requirements will be changed continually Memo:1100
P628	It was rolled out in phases, and uh, it worked	You have to love it when a plan comes together - phased roll outs have proven over the years to be extremely successful with software. Memo:1102
P628	There were complete communication breakdown because once they excluded those two people from the meetings there's no communication happening, so nobody knew what was going on	By refusal to communicate to the first level managers, the passive resistance to following a process causes any hope of good work to fail; it also allows those performing the exclusion of certain mangers to remain one step removed from the train wreck and
P5150	You start with the framework and build everything	***Like the simplest of computer programs, a framework (or model) provides the basis for your coding work - a microcosm of SPI? Is this part of the Theory of Software Development process? Memo:1123***
P42	he developed this process of regression testing and everything; it increased productivity to one point where one person could now do the job of three people	Utilizing automation in the development of automated systems - a good idea that generally is not so well embraced by management - Memo:1135
P147	the most successful project, implementation project I worked on, there was an absolute dedication from the outset to keep the end user in	Losing focus of the customer is so easy to do in the day to day project hassles. When the project gets its "own life" we tend to treat "it" as the customer and not the person, people, or organization that is really backing and/or funding the development.
P147	but were willing to go out and find people that did to take care of people that they could trust, they put the right team around them, they almost never failed	***Surrounding yourself with "yes people" is the first step to destroying any project. Senior management that completely fails is management that immediately rids themselves of any malcontents or naysayer and always promotes and listens to the "yes people."***
P123	there were some processes in terms of how things got elevated up to decide okay what's going to be fixed, what's not going to be fixed	***Maybe not formal, but process was being followed -TSDP Memo:1175***
P123	The area that I'm seeing on this project, that's the issue, is the very front-end requirements	On a road trip from Chicago to New York, you may need to stop and reconsider the direction you are heading when you see the signs for Denver. Memo:1182
P123	standards and things like that, all that's really important, but it doesn't really get you very far if you don't have where you're going	Lack of project vision and scope is the first mistake most people make; they assume everyone "knows" what is supposed to be done and jump in gathering requirements. The first vision and scoping of the overall project is the most vital part of a project;
P122	[Way to change things] Automate. I'd use every	Automation of work is great EXCEPT when it is used as a means of

Participant ID	Participant Data	Researcher
	automation tool known to man	bypassing the "human touch" or as an attempt to redirect responsibility from self to a "program." Ultimately, a good mixture of automated tools and some plain roll up the sleeves and get th
P11	There's also the risk of over-utilizing techniques like that as well and again; you end up focusing more on the process of you're using instead to do software development instead of the actual	First line of defense in "I don't wnat to change!" Always sounds so altruistic - but really, when examined closely, it is just whining and excuses. Obviously, no management in their right mind wants a worker
P3	I think I would have done more prototyping with more user feedback	Prototyping is another methodology for helping understand the customer requirements - unfortunately, it can become very costly if the customer continually looks at prototypes and say, "No, that's not what I wanted."
P3	7 figure salary people on this team, not too many, but one or 2 and at the end of 6 months everybody kind of slapped their cheek and said , Did anybody think to ask the end-users if they like this? then found out that it wasn't anything	Investing in expensive or smart people without process structure ends up with same kinds of messes, just a lot more expensive ones Memo:4156
P44261E	However, the long term benefits of implementing process improvements, makes the front end effort very worth it; The longer a project exists under an advanced maturity model, the more the benefits	A true convert! It must be repeatedly learned and shown to the organization in order to gain cultural adoption - institutionalization Memo:1229
P44261E	The most obvious example, is delivering a solution to the client that actually meets all of their expectations	***When it works, it REALLY works. It is a good feeling to successfully meet the client's needs AND make a profit doing so. Especially after not burning out your people with 60+ hour weeks for 2 years.*** ***Theory of Management Commitment Memo:1230***
P44261E	thoroughly tested meaning less time wasted addressing coding and other system errors	***ROI - When things are developed and tested properly, the expense of re-work is greatly minimized. If you do not want someone to follow stand processes, then expect them to fall back to the programmers bread and butter methodology - code, fix, repeat T***
P44261E	It forces the project manager to think through the total scope of the project	Project planning - one of the key 7 CMMI Level 2 process areas, provides the basic project view for its successful completion
P44261E		Cheapest is never the best. Neither is most expensive. Memo:1239

Participant ID	Participant Data	Researcher
	Reasonable work estimates and commitment to those estimates;often times as an organization we tend to be the cheapest option	
P45987E	Critical applications are being developed without sufficient requirements control and development	How were they ever tested if there were never any requirements? Best kind of software business - don't have to prove anything to the customer
P45987E	members of the staff are currently putting in excessive overtime, and there is a general sense of chaos	If we don't know where we are going, we really don't need to worry about ever getting there, because after we arrive, we wouldn't know the
P45987E	Tools to facilitate REQM and CM	Developers always think latest and greatest technology tools will fix most anything - processes are basic, frequently manually oriented, and not subject to automation in some cases. They still are essential and can fix things that are broken - such as wo

Number 112
Name: **Lack of process**
Coding Open
Date 10/14/06

P630	I was one of the people complaining about lack of	There are those that are the torchbearers or champions of process within an organization that attempt to persuade management and others to take it seriously. Unfortunately, it is often a difficult sell - managers
P630	Like I said, I don't think that because it was undocumented, I don't think it definitely wouldn't scale.	Understanding that the process may need to be actually documented Memo:1044
P630	I would say they [processes] never are [followed] unless management buys in first.	Is this a contradiction? Although not explicitly stated earlier, this organization seemed to have fairly established processes amongst the development staff -no earlier mention that management had to buy-in first. Memo:1047
P630	My last project was, I always thought ,the biggest issue on the project know, the reason that we couldn't seem to get off the ground at all, ecause we could never nail down what we were	As mentioned throughout this study, one of the most frequent issues regarding the failure of a project was the lack of understanding of just what is to be built. It begins when the customer is not clear on their own business requirements, they can then n
P630	supplier agreement management; I felt like the	The world of COTS - Commercial Off The Shelf - software - supplier

Participant ID	Participant Data	Researcher
	customer didn't have a good understanding of how to purchase, or procure software.	agreement management is much more than just the procurement of software, however, it is definitely one component. Unfortunately, when projects start to stall and have problems, people do
P630	Didn't know what to expect or what they should be expecting from their contractor, didn't know how to communicate what they wanted and I didn't feel like the contractor did a good job at educating them either on what they should expect.	When expectations are not met, it is the problem of both the client and the contractor; the client maybe did not present their expectations clearly; the contractor maybe did not elicit the expectations properly. In
P629	somebody has to mess around changing something back ;fixing something because I change was made and somebody wastes half an hour; that person may waste a half an hour then and you keep changing things and different stuff	The only thing worse than one person trying to manage hundreds of lines of their own code is more than one person trying to work together on multiple parts of programs without good configuration management practices. Things get changed without one person
P629	we have this problem with this guy, you know, with getting different requirements in and they're continuing to change	Without planning for change and managing those changes, development cannot proceed effectively and efficiently - running into blind corners and
P629	You don't realize, you don't think about how much that actually affects people until you see something blow up on somebody	When things start going wrong, it is usually too late. Customers are upset, projects are delayed, there are turnovers in personnel, there is just an entire series of bad events when a project is not performed in a structured manner. Memo:1085
P628	there were several smaller projects that were worked on; Here's what we need; Go and do it however you want	Do small projects have no value? Or does their value mean little compared to the overhead to enforce process throughout. Either case, it is not uncommon for even the people supporting process on a major project, to shortcut processes for their own work.
P628	And now it's this running joke, Oh Carl won't tell anybody when something's broke	***Personal attacks on those that attempt to follow process or procedure are not uncommon in the developer environment. Probably, for any, it goes along with not have "buy-in" to the concept of process - change resistance - just sidestepping what really nee***
P628	Process are not followed in the environment I'm working in at the moment	***Conversely - when process is not followed, it adversely affects the project - in fact, failed process implementation is probably more destructive than allowing natural Theory of Software Development Process flourish on a project Memo:1093***
P628	I don't think it was just developers either because	God project planning by management - they successfully planned for

Participant ID	Participant Data	Researcher
	the whole project team, database team, they had the exact same issues that we were having	the failure an obtained it Memo:1111
P628	So it was just a communication breakdown from project management level to anybody else involved with the project	Break communications and quickly sabotage the project Memo:1112
P628	You can throw real data into a prototype to see what comes out of it, but you don't turn the prototype on for real	Testing the function of data in a specific prototype may be useful Memo:1114
P628	hey from my experience it's bad to turn the prototype on for real; "oh we're not going to do that Three months later it's turned on for real	Customers or manager with prototypes do not seem to clearly understand that they are generally just something thrown together for proof of concept - Memo:1117
P42	I can see how, even if they had our process, which wasn't maybe perfect but it would light years from where they are now; And they wouldn't	Even companies known for have strong process management have areas where the troops are fighting tooth and nail to keep them out.. When organizations are broken into little autonomous empires - sometimes the local king just doesn't want to go along. Me
P3	I think that because we were a young organization, um, from the time that I started working there it was a very young organization	New organizations or old - sometimes it does not matter - each project is approached as "New" and if you've worked on them for months or even years, you're apt to make the same mistakes over and over until a
P147	hold people accountable for not just for the folks implementing the processes but for the folks that are going to be affected by the process when it is done	Who takes ownership of the issues? Implementation of a process is the first step; continuation of the process is the more difficult step. When you are messing with the way people do things, generally people
P102757	You had a better defined process, and the very beginning of that is having, you know the requirements and the scope and a model of the system, before you ever start the development process	Must have a clear starting point Memo:1207
P44261E	We neglect to truly reflect the actual work effort of work, and budget to that work estimate; project managers struggling to cut cost, and over burden	ROI needs to be tightly coupled with the project management and processes for correctly estimating level of efforts - low enough to win the
P45987E	A third, more recent, project is operating in a nearly complete ad hoc fashion; It could greatly benefit from strong requirements management	***Another non- success story -*** ***TSDP to the -1 Memo:1243***
P45987E	Most applications have no requirements documents	How were they ever tested if there were never any requirements? Best

Participant ID	Participant Data	Researcher
		- just hand it to them and say, "Trust me, this will do what you need!" Memo:1244
P78952E	Not necessarily; here needs to be an internal quality assurance process to ensure that the project is following there own set of rules	Implementation of process requires its own processes to manage the implementation. At one of the great process institutions in the world, I was once told "We do a terrible job of following processes here as part of our own work! Memo:1262
P78952E	lack of an org PPQA plan, lack of an org process group to audit projects	Why things go wrong sometimes - with these missing, things will go wrong most of the time Memo:1266

Number	113
Name:	**Process failure**
Coding	Open
Date	10/15/06

P630	I've seen a lot of process stuff at both places I've worked over the years, the process will be sitting out there in a Word document and for a week or two somebody will go gung ho "let's apply the process then it fizzles out and you go back and do things	***Failed process only creates a larger barrier for improvement. The scenario is, some mid-level manager preaches process improvement and catches the ear of a senior manager - the senior manager is mildly interested in it and asks for process improvement, b***
P630	gotta be technical improvements there that can happen, but still this doing all your software documentation in Word documents, to me, is absolutely crazy; its not a manageable media at all ; you can't do anything with it; you end up trying to revise	***Stating that documentation must be written and maintained is the easy part. While the initial document may frequently be put in place, the continued management of the document and updating based on changes in the project is another story. This is an are***
P628	Well have you logged it? I said, No, I'm not going	Break down in the effective use of process through personality interaction Memo:1089
P628	the customer keeps changing their mind as to what they want so the requirements keep changing; they're still re-hashing and that needed to be something that was locked down a year ago	One of the very hardest actions needed to learn by software analysts and requirements analyst is being touch on customer. You almost have to literally grab the customer by their tie and look them in the eye and
P628	if our representatives are uninvited from the meetings how do we know what's going on so therefore, the entire procedure broke down	It is easy to make process failure occur by excluding the key people from meetings and events Memo:1107
P628	the last three big projects I worked on all had prototypes involved;Two of them ended up	Everyone seems to think shortcuts will work Memo:1116

Participant ID	Participant Data	Researcher
	on the prototype in production	
P5150	just like the software application I was telling you about we've got the application now we are trying to cram the framework into it things are squishing	In trying to go back and fix process in a previously unstructured project is like trying to stuff watermelons into a pop bottle Memo:1127
P42	Human nature doesn't, we're lazy, humans are somewhat lazy by nature; If given the opportunity to do nothing or to do extra work, people , will pick the do nothing	***The extra effort of managing processes for those that have dodged the bullet in the past always seems like a hassle and a reason to resist. - Change resistance Memo:1139***
P42	normally when I come into a project its seem like they've either been under bid or, which means somebody screwed up	When problems start occurring - start throwing bodies at it - typical management knee-jerk reaction - possibly analysis and evaluation and seeking out root causes of problems would be a better tactic? Memo:1141
P3	every 6 months we were kind of re-inventing ourselves again, so requirements management was a constant struggle	Change and growth are always so difficult - if it just wasn't for the people, we could probably get something done Memo:1152
P3	It was a power struggle, it was and organization struggle, um, it was, it would just constantly at the forefront, getting reasonable requirements and sticking to them	Without clear vision, you cannot gather good requirements. There are actually several layers of requirements, and each need to be defined through proper vision of what the client is trying to accomplish. Unfortunately, requirements are frequently handed
P11	I also feel that sometimes the way in which these types of things are implemented can be of a hindrance if they're too rigorous	Expecting process to be managed in a specific way is the right of the manager. No matter how great a single worker is, that is all they are, a single worker. And there are plenty of "single worker" jobs out there. An organization needing to have several
P11637E	Expensive, cumbersome, may slow down implementation	Reasons (excuses) to not implement process - unfortunately, this is followed by the next question, "Please give me a reasonable alternative?" and the answer is.... Memo:1213

Appendix C

Curriculum Vitae

William Grant Norman, MSSE

Education:

Current Doctoral Candidate, Ed.D. Dec 2006 (Projected) Technology Ed – Software Process Models	West Virginia University Morgantown WV 26506
Master of Science Software Engineering, Aug 2003 Systems Management and Process Models	West Virginia University Morgantown WV 26506
Bachelor of Arts, Humanities, May 1974 English Literature and Speech	Alderson-Broaddus Philippi WV 26416

Professional Experience:

A total of 20 plus years automated systems and management experience in the areas of program manager ($1.8 Million budget DOJ project), project manager, program coordinator (DoD project), project leader, software developer, software testing and evaluation, software instructor, business development manager, market analyst, financial software support and development, editor and publisher of technical newsletters and software products. Designed, interfaced, programmed, and supported biometric access control systems as early as 1994.

I have managed project resources in the areas of software design, requirements management, configuration management, project planning and software testing. I have extensive background in working with a variety of individuals and teams, acting as manager, lead and team participant. I have prepared and presented numerous software development, training, and other reporting presentations for a variety of audiences including project customers, senior management, and congressional and military personnel.

I have extensive experience in full lifecycle system design, from medium to large complex projects, including development of project charter and analysis, user requirements analysis, system architecture and design, coding and development, testing and evaluation, implementation and training, support and maintenance.

I have designed and managed software development and database development systems applications for a wide variety of situations from areas of commercial software products utilizing identification technologies, such as barcode data collection and biometrics, sales and marketing database systems, personal information managers, state government Medicaid data tracking systems, electronic data interface (EDI) tax filing systems, petroleum industry multi-billion dollar bulk oil invoicing and tracking system, and manufacturing work in process and time and attendance tracking systems..

Created complete project management documentation for management review and developer implementation, including specific information on risk management issues, change control, regression testing, and employee development.

Managed staff for multiple types of system development projects and several levels of testing in interfacing biometrics, bar code data collection systems, access control systems, manufacturing systems, unit testing, system testing, regression testing, and CMMI© implementation and measurement and analysis process areas.

In both one on one and within large group meetings, led customers through the process requirements analysis, design and development in order to facilitate their project needs for a specific application. I have additionally organized large seminars and have spoken several times before small groups and as many as several hundred people.

As a writer, editor, and publisher I have produced a large variety of works, including project documentation reports, training manuals, tests, fiction and poetry. I have edited a wide variety of writing from other authors and have managed the publication of several creative and technical publications. Additionally, I designed and created multiple websites for different organizations including West Virginia InfraGard and Rotary Club of Cheat Lake.

Employer: TEK Systems on contract at Lockheed Martin on FBI Project
Period: 04/08/2004 – 05/30/2005
Job Title: Software Design Lead ; Requirements Management Lead, Documentation Lead; Process Implementation

I supported the management and development of software with Lockheed Martin for the FBI. I worked in the areas of software design team lead, requirements management lead, and document lead. I assisted in the implementation of CMMI© process areas in requirements management, configuration management, and overall software process improvement. I worked with IBM's Rational Requisite Pro and managed the Caliber RM requirements management software environment.

I have managed, on average, between 4 to 8 software designers, technical writers, and interns, and have supported and assisted both the project manager and the chief project engineer in the areas of scheduling, process creation and implementation, requirements review and analysis, and documentation of deliverables for review.

Additional project details and information are classified.

Employer: Analytic Services Inc.
Period: 12/09/02- 01/08/04
Job Title: Program Manager

I managed the development of a biometrics laboratory software and device evaluation systems for law enforcement agencies. Funded by a grant from the National Institute of Justice, setup the first independent biometrics laboratory testing organization within the State of West Virginia. Managed and signed-off on funded purchases for biometrics devices, software development kits, lab test equipment, systems, software products, research materials, and a wide variety of other related lab materials, such as tools, furniture, and vendor support agreements.

Held accountable for laboratory, including forecasting costs, business management, scope and vision for the laboratory, and managed both the quality of final laboratory output and the business requirements for testing and evaluation of a wide variety of biometric devices and software products.

I was responsible for the overall management of both Analytic Services Inc. employees and contractors. This included the hiring of new employees and contractors, dismissal of employees and contractors, employee and contractor evaluations, assignment of duties to all employees and contractors, and regular reporting of contractor, employee, and other human resource information to the district manager and director.

I initiated CMMI© Implementation, beginning with the introduction of the process model, CMMI© V1.1, from Carnegie Mellon's Software Engineering Institute, taking this previously unstructured software development shop to a Level 2 Managed organization in less than 1 year. Instituted multiple process area management team leaders to oversee and manage initiating, planning, and controlling of software requirements, planning, controlling, and executing process areas of project control management, risk management, process quality assurance, supplier agreement management, and other specific areas of process control and management as required to produce deliverables within the business and schedule requirements.

In depth documentation of this model introduction at Analytic Services Inc., August 2003, in Masters Thesis, "Implementing CMMI© V1.1 in a Previously Unstructured Environment."

As a selected speaker, the results of this implementation, specifically in the biometrics laboratory area, were presented at the National Defense Industry Association CMMI© Conference in Denver, Colorado, November 2003

Managed the contact and development of proposals for development of biometric pilot projects with a wide variety of law enforcement and educational institutions, including Broward County Sheriff's Office, West Virginia State Police, Morgantown Police Department, Harrison County Schools, and the US Army National Guard. The proposals included projected costs, development time table, customer requirements, testing procedures, and delivery and installation of biometric devices, software and support to meet the needs of complex problems and meet the customer's expectations.

Employer: Alderson-Broaddus College – Mollohan Training Center
Period: 10/15/2003 – 12/15/2003
Job Title: Part-Time Instructor

I served as an instructor for the Mollohan Training Center teaching two courses, Business Communications and Introduction to FrontPage 2002. I was responsible for the course design and curriculum, choice of learning objectives, course outline and syllabus, and execution of all course lectures, assignments, tests, and labs.

Employer: Self- Employed Consultant Contract: West Virginia University.
Period: 09/30/01- 12/09/02 11/15/01– 12/09/02
Job Title: Program Coordinator

Coordinated and promoted software engineering and biometrics programs at West Virginia University under the direction of the Chairman of the Computer Science and Electrical Engineering Department. Duties included coordinating biometrics class development, logistics and planning for 5 Day Introduction to Information Assurance and Biometrics courses with DoD contractors and personnel. This included the development of new procedures and enhancement of project management in order to meet the tasks of the DoD funded program, including, but not limited to, the development of a project tracking system within Microsoft Project, the creation of IPR's for quarterly reporting to the contracting company, presentation of project tasks levels of completion and tracking to DoD contractors and personnel, the development of curriculum tracks of graduate level, biometric related courses, interface and work with faculty in the development of these courses, timelines, milestones, and coordination of faculty effort. This frequently involved the analysis of existing university procedures and processes and the recommendation of new or alternated processes and procedures in order to meet contract deadlines.

Financial management and purchasing of biometrics and computer related equipment to setup the West Virginia University biometrics laboratory. Recommended and pursued the successful implementation of biometrics access to the laboratory itself, coordinated the installation of the Sagem Morpho IAFIS system, which had been donated to the university, served as the manager for the Sagem Morpho system, coordinated the training of faculty, staff, and students on the system, Coordinated and ordered specific biometrics and computer products, scheduled and planned for implementation and installation, reviewed costs and forecasts with chairman and other financial administrative individuals, and managed a graduate assistant in the development of these specific tasks: Biometrics 5 day Course Implementation, processing of course evaluations and materials, course planning and structure of biometrics graduate level track, training and management of the Sagem Morpho system.

Marketed and promoted both the software engineering and biometrics programs. Researched and developed marketing databases for potential students and program donors. Developed contacts for scholarships and secured the first $48,000 biometrics student scholarship for West Virginia University. I visited extension campuses of West Virginia University and presented information on software engineering and biometrics as recruitment information to prospective students.

Employer: CDI Consultant
Contract: West Virginia Dept Health Human Resource
Job Title: Project Manager
Period: 01/17/00 – 09/30/01

I served as Project Manager on a Medicaid database system design and normalization. I evaluated existing design and recommended migration to SQL Server 2000 to manage an estimated total of 5 million records by year 2015. I managed the project, which included two other contractors and as many as 7 state employees. Designed, developed, and employed several systems using MS Visual Studio products. Managed and served as DBA on database systems, managed the database normalization, design, backups, installation and recovery on SQL Server 2000

Developed and taught several classes from basic computer operation procedures, through object oriented programming concepts, and SQL Server 2000 DBA and development. Documented both new and existing systems and taught classes concerning software documentation techniques and practices. I assisted state employees in preparation for state examinations in order that they may be promoted to programmer jobs.

Instituted source control Configuration Management system utilizing Visual SourceSafe and instructed state employees on proper usage of the source control program features and functions, plus initiated data backup procedures and database replication procedures to protect from data loss.

Employer: Buchanan Associates
Contract: ISC
Job Title: Software Consultant
Period: 08/16/99 – 12/15/99

For ISC I developed an interface between an ORACLE 8 based Point of Sale data system and SQL Server 7 Great Plains Accounting system. Created interface to pass data between systems and update files on both systems using Visual Basic 6.0 with ADO 2.1 Work included management of both a SQL Server 7 and Oracle 8 test server. I met with various other developers and support personnel from Great Plains Software. This required the understanding of various accounting functions such as accounts receivable, point of sale, inventory, and general ledger. I designed a custom interface which was responsible for managing significant sales revenue from Kansas amongst

more than half a dozen remote stores in New York state, dialing in and downloading daily sales information.

Employer: Access, MIS Inc
Contract: Koch Industries
Job Title: Software Consultant
Period: 12/04/95 – 06/30/99

For Koch Industries, served as a Team Leader, designed, coded, and supported a multi-million dollar motor fuel tax filing system MF Tax, written in Visual FoxPro 5.0 using OOP concepts and class objects. This EDI and paper tax filing system pulls information from the mainframe, formats, and submits data to motor fuel taxing authorities of 40 different states within the USA – approximately 12 of which were filed via EDI, the remaining were filed via paper reports.

Managed, designed, coded, and supported this project, which involved coordinating with up to half a dozen tax analysts, accounts, and managers. Required extensive research for use of ANSI tax filing standards.

Served as a Team Leader, designed, coded, and supported: a multi-billion dollar AR/AP: Ship Windows written in FoxPro 2.6 Windows. I worked directly with accounting supervisor and system administrator. The completed system tracks purchases and sales of petroleum products; produces invoices, updates mainframe with accounts payable for wires/check processing, provides a variety of paper reports, DDE interfaces to Excel, international invoicing provided through interface to Word 97.

Designed, coded, supported data feeds to multi-million dollar Risk Management system from Ship Windows system above, to SQL Server 6.5 based system. Written in Visual FoxPro 5.0. Performed all ODBC connections between systems, launched stored procedures on the SQL Server using Visual FoxPro, updated server with risk information for decision support.

Served as a Team Leader, designed, coded and supported Deal reconciliation system in Visual FoxPro 5.0. Set up multiple data connections via ODBC with remote views and updates from SQL Server 6.5 and ORACLE. System gathers data from multiple data sources and presents a variety of data views and formats for reconciliation of sales of petroleum product trades. Design based on OOP concepts and class objects. Use of ActiveX controls.

Employer: Self-Employed
Contract: ATM/Canterbury Corp
Job Title: Software Consultant
Period: 09/15/93 – 10/01/95

Designed and wrote Master Trak in FoxPro 2.6 Windows. This product is a commercially sold file-folder tracking system using barcode data collection to track the movement of file folders in large companies. I programmed a wide variety of handheld portable barcode scanning devices, printers, and network communications controllers. Numerous Fortune 500 companies including Exxon, Kellogg, American Express, IBM and others purchased systems.

Designed and wrote Macs in FoxPro 2.6 DOS access control system for monitoring access to health clubs and swimming pool areas. This MACS system used both barcode triggered scanners and a biometric hand recognition system. I integrated all the scanning from all devices into FoxPro 2.6 DOS in both standalone and networked environments. [

Employer: Self-Employed
Variety of Customers
Job Title: Software/Hardware
Consultant

Period: 04/01/84 – 09/15/93

I performed a wide variety of automated systems work for a variety of organizations. I won a bid for Epson printers with Houston Independent School District. I managed the installation of over 400 printers by my own contracted staff in a period of 3 weeks at over 100 different schools in the Houston metropolitan area.

I managed a variety of Novell network installations at several companies in the Houston area, including CPA firms, Country Clubs, and manufacturing companies. Maintained various barcode data collection systems at manufacturing organizations and supported and trained various administrative staff at these customer locations in the use of Word Perfect, Lotus, and other pre-Windows era DOS based personal computer programs.

Completely designed, coded, implemented, and maintained software for managing a work in process and time and attendance assembly line tracking system in both a DOS and Unix based system. System provided information for the actual assembly times of various valve actuator devices being manufactured for large utility and ship industry. Prior to the use of this system, actual build times for these custom controls were only estimates.

Complete auction management system for use in auctions, provided barcode tagging for items and real time data entry of bids. Prior to implementation of this system, customers had to wait for paper transactions to reach cashier in order to pay and exit auction. The real time auction management allowed the customer invoice to be generated at the end of bidding, allowing for immediate checkout of the clients.

Completely designed, coded, supported, and maintained wholesale decorative products tracking systems for carpet, fabric, and rug companies. This product tracked the activity of samples, rugs, or fabrics, utilizing barcode technology and scanners. Systems were

sold and installed nationally at a variety of design centers in several major cities. System was written about in *Flooring Magazine* as a way to manage and control the problem of missing samples, rugs, or fabrics. The product was still in use as recently as 1999 in several locations.

Special Qualifications

Completed several graduate courses at West Virginia University as part of Masters Degree from 2002 - 2003:

SENG691K – Software Requirements Engineering – Extensive examination of the requirements gathering, management, testing, customer expectations, project vision and scope, and overall management of software project requirements.

SENG691L – Data Warehousing – Concepts of managing, developing, and coordinating large data warehousing systems and projects; understanding the various components of large data access and warehousing procedures, designs, and practical applications.

SENG 591: Introduction to Object-Oriented Design
Developing a software system from an object oriented perspective.

SENG591A – Personal Software Process – Development of software management skills for utilization in project estimating, defect tracking, time tracking, with a goal of overall software process improvement.

SENG 520: Software System Analysis and Design
Defining the software requirements of a large and complex software product and the principles and concepts of designing the software that will implement the product are discussed.

SENG 510: Software Development Project Management
Topics include: project management process, measures and metrics, project planning and estimation, risk analysis, scheduling, tracking and control.

SENG 530: Verification and Validation
Processes and methods for evaluating the correctness and quality of the software product throughout the software life cycle are discussed.

SENG 540: System Lifecycle and Configuration Management
Topics include: Software process and CMM and CMMI, software maintenance and evolution, program understanding, reengineering software, configuration management and software tools related to these issues.

All the above course were completed between January, 2002 and August 2003 with a 4.0 grade point average.

Doctoral work has included the investigation of the diffusion of software process models, privacy and legal issues in the implementation of biometric and other automated identification systems, and the study of various technological impacts on society of various automated systems and technology. Dissertation topics include the implementation and issues of software process models in software development and biometrics research environments.

Memberships

ACM – Association for Computing Machinery
AIS - Association for Information Systems
PMI – Project Management Association
SEI – Software Engineering Institute

Publications

Norman, W.G. (1993, January). Managing activity with a computer. *Flooring Magazine.*

Norman, W.G. (2003, August). Implementing CMMI in a previously unstructured environment. Masters thesis. West Virginia University, Morgantown, WV.

Norman, W.G. (2003, November). Implementing CMMI in a biometrics testing laboratory. NDIA 3rd Annual CMMI Technology Conference & User Group, 18-20 November 2003, Denver, CO.

Norman, W.G. (2005, August). Implementing the capability maturity model CMM: praise and criticism. AoM/IAoM 2005 Conference Presentation August 12-14, Norfolk, VA.

Norman, W.G. (2006, March). *Implementing CMMI in a small organization.* Poster session presented at the Software Engineering Process Group conference, Nashville, TN.