Walden University

COLLEGE OF MANAGEMENT AND TECHNOLOGY

This is to certify that the doctoral dissertation by

Lawrence Day

has been found to be complete and satisfactory in all respects,
and that any and all revisions required by
the review committee have been made.

Review Committee
Dr. Raghu Korrapati, Committee Chairperson,
Applied Management and Decision Sciences Faculty

Dr. David Gould, Committee Member,
Applied Management and Decision Sciences Faculty

Dr. Aqueil Ahmad, University Reviewer
Applied Management and Decision Sciences Faculty

Chief Academic Officer

David Clinefelter, Ph.D.

Walden University
2011

www.manaraa.com

Abstract

The Integration of Software Quality into Software Project Management

By

Lawrence E. Day

M.B.A., Technology/Engineering Management, City University, 1988

B.S., Electrical Engineering, Worcester Polytechnic Institute, 1969

Dissertation Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

Applied Management and Decision Science

Walden University

February 2011

www.manaraa.com

Abstract

Billions of dollars have been lost in failed software development projects in the past 40 years. Although there are standard project management processes, the data indicate they are inadequate when it comes to software projects. Standards are needed to produce predictable repeatable results. The problem researched in this study was the lack of understanding about how to measure real progress-to-date of a software project. The purpose was exploring ways to communicate real status of a software development project. The theory of software product quality provided the theoretical framework, the methods of research, and the methods of analysis. The three research questions explored effective ways to measure software project status by analyzing product quality without adding communication barriers for senior management. An ex-post facto exploratory research design was used. Data were collected from inspections of project requirements. Data analysis used statistical process control (c- chart for defects), simulation, input sampling techniques, and parametric analysis. The sample studied constituted the requirements of a completed software project. The results showed that the Quality Performance Index (QPI) method developed in this research did yield a quantitatively significant indication of a project's status. Implications for positive social change included the development of more robust software with better quality and cost management resulting in greater customer satisfaction and savings to stakeholders in industry and taxpayers for government projects.

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

The Integration of Software Quality into Software Project Management

By

Lawrence E. Day

M.B.A., Technology/Engineering Management, City University, 1988

B.S., Electrical Engineering, Worcester Polytechnic Institute, 1969

Dissertation Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

Applied Management and Decision Science

Walden University

February 2011

www.manaraa.com

UMI Number: 3440422

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

UMI 3440422
Copyright 2011 by ProQuest LLC.
All rights reserved. This edition of the work is protected against
unauthorized copying under Title 17, United States Code.

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

www.manaraa.com

Dedication

"Study to shew thyself approved unto God, a workman that needeth not to be ashamed, rightly dividing the word of truth." – II Timothy 2:15

To my wife, Kyung Ae Day, without whose love, support, and encouragement I would not have been able to allocate the time, energy and financial resources to complete this course of study. To my Lord who sustains me daily; who provides my Eternal hope and is the author and finisher of personal change and hence positive social (cumulative personal) change. He worked in the life of John Newton to change him from a slave trader to a preacher and a major figure in the non violent abolishment of slavery within the British Empire. Amazing Grace is a poem he wrote to document the personal change that ultimately resulted in one of the greatest positive social changes in modern history.

Amazing Grace

Amazing Grace, how sweet the sound,
That saved a wretch like me.
I once was lost but now am found,
Was blind, but now I see.

T'was Grace that taught my heart to fear.
And Grace, my fears relieved.
How precious did that Grace appear
The hour I first believed.

Through many dangers, toils and snares
I have already come;
'Tis Grace that brought me safe thus far
and Grace will lead me home.

The Lord has promised good to me.
His word my hope secures.
He will my shield and portion be,
As long as life endures.

www.manaraa.com

Yea, when this flesh and heart shall fail,
And mortal life shall cease,
I shall possess within the veil,
A life of joy and peace.

When we've been here ten thousand years
Bright shining as the sun.
We've no less days to sing God's praise
Than when we've first begun.

www.manaraa.com

Acknowledgments

In chronological order I would like to acknowledge Dr. Larry Beebee who encouraged me to enroll in Walden and was actively concerned about my successful progression through the curriculum. Dr. Raghu Korrapati became my faculty mentor and my dissertation chair. His academic and industry background provided just the mix of viewpoints that fit with my background. He used some of the best research practices of industry to help ground the students in making their academic activities meaningful. His guidance and friendship has been a constant for excellence and professionalism throughout my entire Walden journey. The Walden faculty and administration have taken seriously the challenge of making virtual study and accomplishment successful. They have made distance learning personal for the student and provided a sense of community despite a lack of physicality of the university. My Director, Curtis Banks, provided the real-world social change problem that became the basis for this dissertation study. He and his successor, Susan Gellatly, provided the resources and support necessary for the successful execution and completion of the study. As a member of my dissertation committee, Dr. David Gould provided invaluable personal insight and direction in allowing me to successfully complete this dissertation.

www.manaraa.com

TABLE OF CONTENTS

www.manaraa.com

www.manaraa.com

www.manaraa.com

www.manaraa.com

www.manaraa.com

www.manaraa.com

List of Tables

www.manaraa.com

List of Figures

www.manaraa.com

Chapter 1

Background of the Problem

The problem areas identified in the NATO report in 1969 were similar to ones that appear in current publications almost 40 years later. The NATO report cited the following software problem areas among others (Bauer, 1969):

- Achieving sufficient reliability in the data systems which are becoming increasingly integrated into the central activities of modern society
- The difficulties of meeting schedules and specifications on large software projects
- The education of software (or data systems) engineers

After almost 40 years of trying to solve these problem areas, software developers were still encountering the same problems. It was estimated that in 1995 alone, that the cost of software project cancellations and overruns in the United States was $140 billion, which accounted for about 20% of total U.S. investment in software development and acquisition (Tully, 2002). In the fall of 2004, the U.S. government had to cancel a very high-profile project, a part of the FBI's modernization program to fight the War on Terror, and later admitted that nothing of the investment was salvageable (Ragavan & Hook, 2005).

Software development and project management were examined during those 40 years. Many books and articles were published that describe the intricacies of how to manage a software development project(from modeling to metrics, software and systems analysis, requirements, project management and quality) , by such luminaries as Boehm

www.manaraa.com

(2003, 2000, 1989, 1981), Gilb (2005, 1993, 1988), and DeMarco (2006, 2003, 2002, 1997, 1987, 1986, 1979). Two other major sources of scholarship and practical industry sharing were the IEEE Computer Society and the Association for Computing Machinery Digital Library. In addition, the Project Management Institute (PMI) developed a professional certification (Project Management Professional) in an attempt to develop a standardized profession of project manager. This search for excellence in software development has been systemic to the profession.

Thirty-seven years after the NATO report (Bauer, 1969), a $170 million government project involving the construction of an integrated data networking system was a total failure (Ragavan & Hook, 2005). According to FBI Director Mueller, the Virtual Case File (VCF) system was plagued by a series of management failures at FBI headquarters (Ragavan & Hook, 2005). Software and IT were taking up, on average, 5% of companies' revenues (Charette, 2005). Some examples of the colossal software development failures over the last few years are listed in Table 1 (Charette, 2005).

www.manaraa.com

Table 1.

Recent Large Software Failures

Year	Company	Loss Outcome (Cost in U.S. $)
2005	Hudson Bay Co [Canada]	Inventory system problems - $33 million.
2005	UK Inland Revenue	Software errors - $3.4 billion overpayment.
2004	Avis Europe PLC [UK]	ERP cancelled - $55.4 million.
2004	Ford Motor Co.	Purchasing system cancelled - $400 million.
2004	J. Sainsbury PLC [UK]	SCM cancelled - $527 million.
2004	Hewlett-Packard Co.	ERP system - $160 million loss.
2004	AT&T Wireless	CRM upgrades - revenue loss of $100 million.
2002	McDonalds Corp.	Information-purchasing canceled - $170 million.
2002	Sydney Water Corp. [AU]	Billing system cancelled - $33.2 million.
2002	CIGNA Corp.	CRM problems - $445 million.
2001	Nike Inc	SCM problems - $100 million
2001	Kmart. Corp.	SCM cancelled - $130 million.
2000	Washington, D.C.	City payroll system abandoned - $25 million.
1999	United Way	Admin processing cancelled - $12 million.
1999	State of Mississippi	Tax system cancelled - $11.2 million.
1999	Hershey Foods Corp.	ERP problems - $151 million.
1998	Snap-on Inc	Order-entry system problems - $50 million.
1997	Internal Revenue Service	Tax modernization effort cancelled - $4 billion.
1997	State of Washington	DMV system cancelled - $40 million.
1997	Oxford Health Plans Inc.	Billing/claims problems –stock loss of $3.4 billion.
1996	Arianespace [France]	Software errors – loss of $350 million Ariane 5.
1996	FoxMeyer Drug Co.	$40 million ERP system bankrupts company.
1995	Toronto Stock Exchange	Electronic trading cancelled - $25.5 million.
1994	FAA	AAS cancelled - $2.6 billion.
1994	State of California	DMV system cancelled - $44 million.
1994	Chemical Bank	Software error - $15 million erroneous accounting.
1993	London Stock Exchange	Taurus system cancelled - $600 million.
1993	Allstate Insurance Co.	Office automation cancelled - $130 million.
1993	London Ambulance Service	Dispatch system cancelled twice - $26 million.
1993	Greyhound Lines, Inc	Bus reservation system - $61 million.
1992	AMR (American Airlines)	Reservation system cancelled - $165 million.

www.manaraa.com

Statement of the Problem

Traditionally project status has been measured on budget, scope and time. But there is a lack of academic research on how to accurately measure project status, the problem researched in this study. This case examined why measurements are inaccurate or incomplete and what could be done about it. This research addressed the lack of academic research in communicating true software status to the project sponsors during the system development life cycle. The quality that was researched in this study, while related to product and software development methodology, was not product quality, but quality of the status of the project management process. Product quality was studied to determine its effect on project status, or *project quality*.

www.manaraa.com

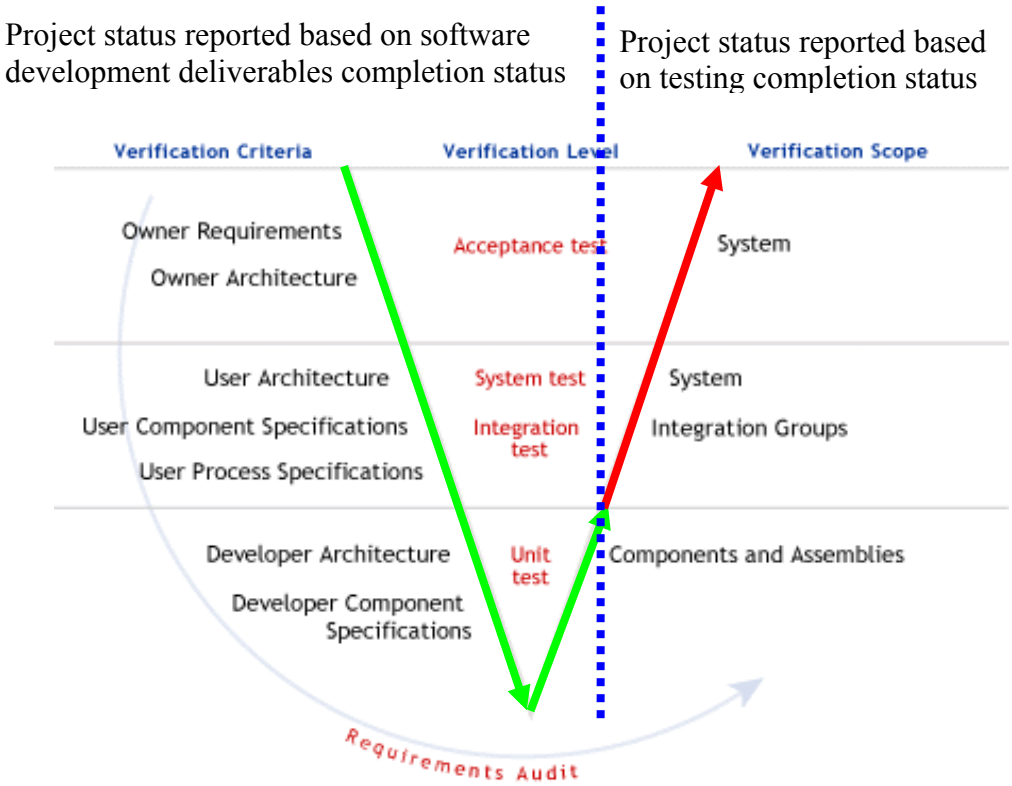

Figure 1. Annotated software development V model. Used with permission.

Prior to the completion of unit test, project status was determined by the status of the software deliverables, as reported by the software development personnel creating the deliverables. This status is shown by the green line in Figure 1. When the product moved into the test region, shown by the red line in Figure 1, the project status was reported based on the progress made in the completion of the testing scheduled. The test group was often different from the development group and the project status changed from the quantitative based subjective status (when was a deliverable complete?) to the quantitative based objective status (the test either passed or failed). The transition in project status accounting could often produce dramatic changes in project status, which were usually not positive producing surprise and consternation among executives and project managers. The study tried to determine if there is additional information that

www.manaraa.com

could be applied to the project status in the development phase (green line) that would make the status transition to the testing phase (red line) more congruent.

Purpose of the Study

In light of the previously identified continuing losses in IT endeavors, the purpose of the study was to make a scholarly contribution to the scientific body of knowledge on how to accurately measure and communicate software development project status. The study determined that a contribution could be made to the existing scientific body of knowledge that would provide new information about how to effectively communicate, in simple terms, the true software project status in development projects. The study analyzed how the professional literature and current body of knowledge supported the research for an integrated solution of quality into the software development project management process. The study developed an enhanced method of software project status that improved the capability of successful software project completion and successfully communicated to senior management.

The study developed an approach to communicating the measurement of project status that executive management could readily incorporate without having to develop any additional software project management understanding. Industry research confirmed that executives should be skeptical of favorable status reports. Major projects with large software components, such as the Airbus 380 2-year delivery slide and the Denver airport baggage handling 16-month-opening slide, continued to show bias in their status reporting (Shore, 2008). The higher risk projects required executives to concentrate on

www.manaraa.com

decreasing bias so that they could more accurately assess the project status (Snow & Keil, 2001). This study provided an early indicator to offset executive bias.

Nature of the Study

This study used a quantitative, post-facto, exploratory design (see chapter 3 for details). The study consisted of a literature search and analysis of software quality measurement methodologies and software project management using earned value management (EVM) to evaluate them with respect to integration or common approach to common solutions. The study was quantitative in nature in that an update, based on the literature search, of the EVM methodology to add a quality component to the already defined modes of cost and schedule was proposed. This updated method was then the subject of research on a completed software project to determine the effectiveness of the methodology against already known project results.

An additional aspect of the study was the specific choice of the EVM methodology as a proposed method to add a quality component as opposed to developing a separate stand-alone quality component. The assumption was made, based on the author's years of experience in this field, that the best way to communicate was for the software developers to change their communication style so that it lined up with senior management rather than having Senior Management try to learn a new technical approach to the management of software projects. In other words, senior managers were already trained in the significance of EVM indications and were able to make correct project deductions from the data presented. By incorporating a quality measure into the Earned Value Management methodology, executives could make informed decisions about a

www.manaraa.com

project based on the quality date without having learned and understood new project management models and terms.

Theoretical Framework

In any project there were three major components: cost, schedule, technical performance or quality, and risk (NASA, 2007). In EVM project status is reported in only two of those components: cost and schedule (NASA, 2007). The projects cited in Table 1 failed because of product quality either in requirements, design or implementation.

The theory of software product quality provided the theoretical framework for the study and the methods of research and analysis chosen for the study. Software quality theory divided into two process phases: engineering and defect removal. The defect removal phase consisted of review and testing. The engineering phase produced new artifacts where defects are introduced (Alstad, 2004). The engineering phases were sometimes called the defect injection phases by quality analysts (Alstad, 2004). The axioms of software product quality included the following (Dromey, 1998):

1. Since software is composed of components, their choice, their tangible intrinsic properties, their contextual properties and the way the components are composed determines the quality of the software.
2. Software exhibits a set of quality attributes; it exhibits observable behaviors and uses that correspond to the quality attributes.
3. Tangible quality-carrying properties of software components contribute to one or more intangible, high level quality attributes of software.

www.manaraa.com

4. Associated with each quality-carrying property of a component is a verifiable empirical statement that links the property either to a software characteristic, a behavior or a use, and then to a high level quality attribute.

In order for projects to be successful, the product they produce must have the quality that meets the customer's requirements: a verifiable empirical statement. The general aims of quality theory are quality planning, quality assurance and quality control (PMBOK, 2004). Upon examining the reasons software projects continue to fail, it appears they do so because of badly defined system requirements and poor reporting of the project's status (Charette, 2005).

The basis for this research was derived from the examination of the essential elements necessary for a successful software development project and how to ensure they are accurately communicated. The project quality indicators that executives relied on for correct assessment of major projects were supplied by project management professionals who relied on the product development professionals' status of completed deliverables. The quality of the deliverables was not an integrated indicator into project status, but was instead, a separate component. The amount of quality control was established by the project. Product quality was measured after completion and the projects status adjusted accordingly.

Measures of software product quality were initially created by Fagan, one of the founders of the IBM Quality Institute, for in-process software development (before testing) in 1976 (Auruml, Petersson, & Wohlin, 2002). Over more than 30 years, the database of literature supporting inspections properties and effectiveness grew to the

www.manaraa.com

point where agile type inspection rules were available (Gilb, 2005). With this basis of data and experience to use as a baseline for examination, the feasibility of quality integrated into project management could be investigated.

The specific aspect of quality being investigated in this study was the aspect of product quality as it applied to project quality while the product was in the development phase of the project. The categories of software development quality were peer reviews, acceptance sampling, product maturity, appraisal cost, internal failure cost, cost of non-quality, cost of conformance, and cost of non-conformance (PMBOK, 2004). The measurement of these areas of quality was not integrated into the cost and schedule aspects of the software development project management. Thus, the input of quality measurements was not directly coupled to project status, unlike cost and schedule, which were in EVM (NASA, 2007).

Scope of the Study

The scope of the study included an IT project in a large aerospace organization. The software development methodology lifecycle that was used by the project in the test case being studied was the waterfall lifecycle with completion of deliverables used to define the extent of activities. Lean, xTreme or Scrum methodology lifecycles were not included. The project researched was one that had already been completed; it was not influenced by any of the results of this study.

Limitations of the Study

This study was limited to software quality and software project management. The sample size to test the research questions was small; more data is always desirable.

www.manaraa.com

However, that was not a factor in validating the study's veracity. The veracity of the method was determined by an analytical logically sound analysis to reach a conclusion. The examination of the data in the samples was non-intrusive to the project personnel. This has no impact on the study, but if determining root causes of some of the sampling data was desired then further effort outside of the study research would be required.

Assumptions

There were three major assumptions that were integral into the validity and value of the study. The first was that the quality system level standards such as ISO-9000 and software development process methodologies such as Capability Maturity Model® Integration (CMMI) would continue to be required and a major organizational objective for the foreseeable future. The second was that project management disciplines and methodologies would continue to have increasing importance placed on them in the business strategies of successful software development companies. The third, as previously stated, was that communication about software from developers to senior management was best accomplished by the developers adopting the communication constructs of senior management rather than attempting to get senior management to understand the intricacies of software development.

Definition of Terms

AAS: Advanced Automation System. A project for the Federal Aviation Authority (FAA) to overhaul the US government's air traffic control computer systems.

Capability Maturity Model® Integration: Capability Maturity Model® Integration (CMMI) was a process improvement approach that provided organizations with the

www.manaraa.com

essential elements of effective processes. It could be used to guide process improvement across a project, a division, or an entire organization. Capability Maturity Model Integration helped integrate traditionally separate organizational functions, set process improvement goals and priorities, provided guidance for quality processes, and provided a point of reference for appraising current processes (CMMI, 2007).

CobiT: A product of the Information Systems Audit and Control Association and the IT Governance Institute. It was a set of guidelines for IT processes, practices and controls that was mainly intended to be used for purposes of audit (ISACA, 2009).

CRM: Customer relations management system. It consisted of all processes a company used to track and organize it contracts with its current and prospective customers.

Descriptive Model: A descriptive model describes the behavior of elements in a system where theory is adequate or nonexistent (Cooper and Schindler, 2003).

Direct metrics: Measurement of a process or product characteristic that does not depend on the measurement of any other characteristic. Examples were the number of faults in a product, number of hours spent during certain process, etc (SML, 2004).

DMV: Department of Motor Vehicles.

Earned Value Management: Earned Value Management (EVM) was a program management technique (NASA, 2007). It integrated technical performance requirements, resource planning, with schedules, and at the same time taking risk into consideration. Earned Value (EV) was a project management methodology which integrated three critical management elements of a project: scope, cost, and time (Anbari, 2003). Earned

www.manaraa.com

Value was a management technique that related resource planning to schedules and to technical performance requirements (Abba, 1997). There were four major steps occurring in the Earned Value process. First all work was planned, budgeted and scheduled in time phased planned value increments. Work was then earned as it was performed. When complete, Planned Value was compared to Earned Value and any difference was called schedule variance, and Earned Value was compared to Actual Cost and any difference was called cost variance (Abba, 1997).

ERP: Enterprise resource planning software. ERP was a company-wide computer software system used to manage and coordinate all the resources, information and functions of a business from shared data stores.

Explicative Model: An explicative model extended the application of a well-developed theory or improved the understanding of the theory's key concepts.

FAA: Federal Aviation Administration.

Information Technology Infrastructure Library (ITIL): A set of IT processes and best practices for IT service management and operations (ITIL, 2006). The ITIL framework consisted of Service Support, Service Request Management, Incident Management, Problem Management, Change Management, Release Management, Configuration Management, Service Delivery, Service Level Management, Capacity Management, Availability Management, Security Management, Software Asset Management and Application Management.

Indirect metrics: Measurement of a process or product characteristic that involved the measurement of one or more other characteristics, such as productivity, fault density,

www.manaraa.com

etc. An indirect metric always contained a calculation of at least two other matrices (SML, 2004).

In-Process: The state or condition of being executed but not completed. For example, usually the quality of a product was measured after the product had been created. Attempts at measurement of quality of a product while it was being created would be in-process. Or if there were measurements of the progress of a phase of software development, such as metrics of status of testing, then these metrics would be labeled as in-process (SML, 2004).

Macroscope®: A software development methodology developed and supported by Fujitsu (Macroscope, 2004).

Objective metrics: Absolute measures of the process or product, and count attributes or characteristics in an objective way with numbers. Examples included number of lines of code, number of defects, etc (SML, 2004).

People CMM (PCMM): The People Capability Maturity Model (People CMM) was a framework that helped organizations successfully address their critical people issues. Based on the best current practices in fields such as human resources, knowledge management, and organizational development, the People CMM guided organizations in improving their processes for managing and developing their workforces (PCMM, 2008).

Process metrics: Measurement of the characteristics of the overall development process, such as number of defects found throughout the process during different kinds of reviews (SML, 2004).

www.manaraa.com

Product metrics: A measurement of an intermediate or final software development product. Examples included size metrics, complexity metrics (SML, 2004).

Project Management Body of Knowledge (PMBOK): A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Third Edition offered a set of processes, generally recognized as good practice, which delivered results across industries and organizations. With over two million copies in circulation, the PMBOK® Guide was renowned as one of the leading tools for the profession and was an essential reference for the library of every project management practitioner. The PMBOK® Guide contained the fundamental, baseline practices that drove business results for any organization – local, regional and global (PMBOK, 2004).

Quality: There were many aspects to quality; here are a few:

1. Software quality is the existence of characteristics of a product which can be assigned to requirements (Petrasch, 1999).
2. Project Quality – consists of Quality Planning, Quality Assurance and Quality Control (PMBOK, 2004).
3. Deming: .What the consumer says.
4. Juran: Fitness for use.
5. Crosby: Conformance to requirements.
6. Conformance to requirements–both stated and implied (McConnell, 2002).

SCM: Supply-chain management system.

www.manaraa.com

Simulation Model: A simulation model clarifies the structural relations of concepts and attempts to reveal the process relations among them (Cooper and Schindler, 2003):

a. Static – A static model represents a system one point at a time.

b. Dynamic – A dynamic model represents the evolution of a system over time.

Subjective metrics: Measurements of a process or product that involve human, subjective judgment & numbers. Examples of subjective metrics are expected complexity and degree of conformance to coding standards (SML, 2004).

Test Phase: The phase of software development when the software developer has completed coding and unit test of the product and it is being tested against functional and performance requirements.

Research Questions

In software development projects that include IT, the project manager gets status on the software deliverables from the development analysts in conjunction with the software development methodology that was being employed. This research study included three research questions on the problem of measuring the software's in-process (before testing) quality impact on the project cost and schedule:

1. What are the requirements for software quality during project execution, prior to the test phase, which will produce reliable predictions of future impacts on the cost and schedule commitments of the project?
2. What software quality measurement technique, currently available, that can be adapted as an in-process (before testing) project quality measure?

www.manaraa.com

3. What is the in-process (before testing) project impact, based on product quality, that requires a communication mechanism or method not currently part of the structure of established project management reporting practices?

Significance of the Study

This study was important because of the ever-increasing uses of large software-based systems for warfighting, financial services, medical advancements, and communications to name a few. As corporate software budgets increase, the need to use corporate resources efficiently will increase. With the implementation of the Capability Maturity Model® Integration (CMMI, 2007), an increase in the predictability of software projects and the quality of their delivered products and services is expected.

There was a financial significance aspect to this study. If a method could be developed that added confidence to software project management, such that software project cancellations could be reduced by just 10%, it would amount to a savings of $14 billion in 1995 dollars alone (Tully, 2002).

There was a public policy significance aspect to this study. When major implementations of public policy projects were delayed and/or overrun, the implementation of public policy was impacted (Schmitt, 2005). Were the failure catastrophic, the will to gather the resources to implement it might have been missing. The FBI Virtual Case File system, for example, had to be started again from scratch. During that time, none of the desired capability was available and the information gathering and coordinating of the U. S. suffered. When the FBI put a new system put in

www.manaraa.com

place, at a cost of $581 million, it was still not fully able to implement public policy at the time it went operational (Schmitt, 2005).

This study had an implication for the software project management community and discipline. The method developed in the study would validate the project status those projects that were integrated in the aspects of cost, schedule and quality. The method would also identify deficiencies in the project status of those projects that were not integrating the three project disciplines.

Implications for Social Change

Governments, corporations and individuals were engaging in cyberspace warfare and economic espionage. Internationally, foreign governments had hacked into U.S. government agencies (Swartz, 2007) and private corporations (Roberts, 2010). The ability of federal officials to respond to technology challenges and threats with on-schedule and within-budget software applications would potentially free up resources for their efficient use in other areas of social government policy implementation (Charette, 2005). The same efficient use of resources in the private sector would free up additional capital that can be invested in the U.S. economy for job growth and sustained competitive advantage (Charette, 2005).

Summary

The problem areas in software development that were identified almost 40 years ago continue to occur today in spite of the effort of governments, industry and institutions of higher education to understand and solve them. Billions of dollars have been lost in failed software development projects. This chapter introduced the problem of lack of

www.manaraa.com

effective software quality project management. It identified a proposed relationship to be developed between quality and project management.

The next chapter contains extensive research in literature on the various approaches to software quality and examples of what were considered successful methods of software project management. The research identified the complexity of the subject and at the same time the lack of interaction in the project management domain.

Chapter 3 addresses the methodology used in this study, which includes the research design and the type of data. In Chapter 4, research results with the use of the Quality Performance Index (QPI) methodology are presented. Chapter 5 summarizes the study and its findings.

www.manaraa.com

Chapter 2: Literature Review

The purpose of the study was to make a scholarly contribution to the scientific body of knowledge on how to accurately measure and communicate software development project status. This chapter is a literature review, and is divided into the following five major areas of the literature review because they constitute the successful development of a software product and address aspects of the research questions:

1. Quality models (research questions 1 and 2)
2. Software development process models/methodologies (research questions 1 and 3)
3. Measurement and metrics (research questions 1 and 2)
4. Software inspections (research questions 2 and 3)
5. Earned value management (research questions 2 and 3).

The five areas of literature review are superimposed on the V model to graphically depict their relationships to software development and each other. Figure 2 is a standard software development generic V model (Macroscope, 2004). The descending part of the V on the left represents the requirements through the code and unit test aspect of software development. The ascending part of the V on the right represents the verification and validation aspect of software development. The software development models are shown across the top; they identify the specific activities and deliverables during the software development life cycle. The quality models are shown on the left side: they identify the processes followed during the software development life cycle. The deliverables within the software development lifecycle are subject to inspections,

www.manaraa.com

especially those in the early part of the cycle. The Earned Value project status was performed as the project progressed through the lifecycle. The measures and metrics were captured and analyzed during the various phases of the lifecycle. This study focused on the left half of the V model, since, when the test phase is entered, the actual status of the project became immediately evident.

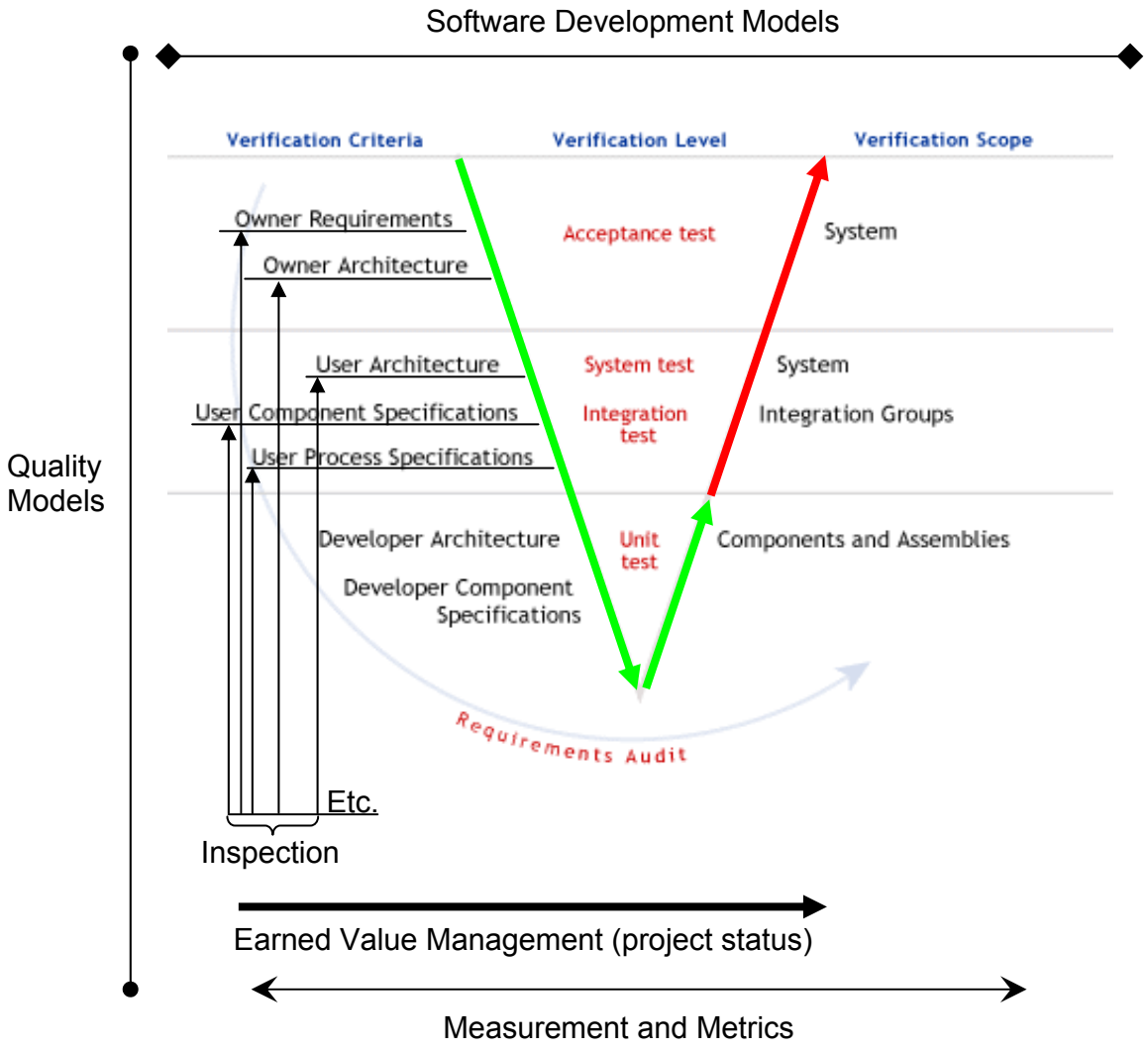

Figure 2. Software development V model with literature research elements added. Used with permission.

www.manaraa.com

A model is the representative of a system to study some aspect of that system and can be put into the following three major categories (Cooper & Schindler, 2003):

1. Descriptive: A descriptive model describes the behavior of elements in a system where theory is inadequate or nonexistent
2. Explicative: An explicative model extends the application of well-developed theories or improves the understanding of key concepts.
3. Simulation: A simulation clarifies the structural relations of concepts and tempts to reveal the process relations among them. There are two kinds of simulations: static and dynamic. A static simulation represents a system at one point in time. A dynamic simulation represents the evolution of the system over time.

This literature review identified existing elements of either a descriptive or explicative model not yet correlated by the profession. Chapter 3 identified the type of model and elements.

Software quality had been, and continues to be, an elusive goal for many information technology (IT) organizations. Military and other government organizations that contracted for large unprecedented software intensive systems had created software development standards that had been regulatory for government contractors. As long ago as 1968, NATO produced a report that identified problems and situations in software development projects (Naur & Randell, 1968). Reading the report today, without knowing the date or source, might have led the reader to believe that the subject was today's software development environment.

www.manaraa.com

Software projects failed most often due to project management quality issues (Jones, 2004). Unsuccessful projects' problems with quality management included not managing changing requirements, not allotting time for detailed requirements analysis, and not allotting sufficient time for verification tasks including inspections, testing, and defect repairs. There were many approaches being practiced with the intent of producing quality in software development projects including the Capability Maturity Model – CMM (CMM, 2007) and Capability Maturity Model Integration - CMMI (CMMI, 2007), IT quality approaches such as Information Technology Infrastructure Library (ITIL), and non-software specific Quality approaches like Total Quality Management – TQM (TQM, 2007) and ISO 9001 (Praxiom, 2007). There were software vendor methodologies and tools such as Macroscope 4.5, Dynamic System Development Method (DSDM) and CMD that attempted to apply consistent software project management disciplines. This study categorized them in their effectiveness against various development scenarios. Each was quantitatively analyzed with the resulting ratings for why they had been chosen. The analysis summary was contained in this chapter and the detailed analysis was documented in Appendix A.

There had been many approaches and methodologies for software quality proposed. This research evaluated them for general patterns of applicability. A partial result of the study provided a general indication of whether a particular type of quality approach was well suited for implementation in a software development project. The breadth of the software quality experience was captured both in quality models and software development process methodologies.

www.manaraa.com

Approaches to quality in general and software quality in particular kept changing with each new one being touted as the approach that would ultimately bring about good quality. Many times the quality approaches were defined independently of the business environment that they were supposed to help. The business executives, who control the financing, had no reference point with which to determine the most cost effective method of quality management to implement with respect to their business model.

The study, as a prerequisite to its findings, analyzed the quality approaches to determine if there was a common way to classify them for easier reference and applicability. Software quality has modeled both in models that deal strictly with the software development process and in models that deal with quality in general, of which software was one aspect of the total quality picture.

Literature Research Statistics

The literature research used the following databases:

1. Academic Search Premier
2. Business Source Premier
3. Computers and Applied Sciences Complete
4. Dissertations and Theses (ProQuest)
5. Military and Government Collection
6. NTIS – National Technical Information Service
7. Regional Business News

Table 2 is a cross-reference table of the type of references in this study and the key words associated with them.

www.manaraa.com

Table 2

Key Words by Literature Type

Key Words	Journal	Professional Publication	Professional Technical Book	Gov't Technical Publication	Article	Technical Conference	Industry Publication	Totals
Metrics		1		2	1	1	5	10
Earned Value	3	3	1	2		1	1	11
Software Quality	2	2		1	1	1	3	10
Software Metrics	2	3	3			2	1	11
Software Project Failure	2	1	2	2			2	9
Software Project Measurement	2		1	1	1	1	3	9
Software process quality	4	1	3	4	1	2	4	19
Software Project Success	3	1	6	3		1	3	17
Quality	1	2	1	3			2	9
Software Inspections	4	1	1		1		2	9
Other	1	3	4	1			0	9
Totals	24	18	22	19	5	9	26	123

www.manaraa.com

The literature that will be reviewed for the study problem area crossed the boundaries that often separate academia, business and government. One of the best examples of this phenomenon was the NASA/GSFC Software Engineering Laboratory (SEL) in Maryland. It was a cooperative venture between NASA, a U.S. government agency, the University of Maryland, an academic institution and CSC Corporation, a for-profit, publicly owned corporation (Basili et al., 2002).

Another example of cooperation and collaboration was the Capability Maturity Model® (CMM) and the Capability Maturity Model® Integration (CMMI) quality model. Carnegie Mellon University, an academic institution, developed this model under contract from the Department of Defense, a government agency. A separate research department of the university was created called the Software Engineering Institute (SEI), It was a federally funded research and development center that conducted software engineering research in the following (SEI, 2009):

1. acquisition
2. Architecture and product lines
3. Process improvement
4. Performance measurement
5. Security
6. System interoperability
7. Dependability

www.manaraa.com

A CMM Level 3, or sometimes Level 2, appraisal had become a minimum requirement for private contractors to be eligible to receive certain government contracts (SBA, 2009). The problem area of this study has been a concern of all three major areas of research and innovation: academia, government and private industry. The literature search into this problem area therefore would include review and knowledge transfer from all three of these areas.

Quality Models

Quality models are included in the literature search as part of answering research questions 1 and 2. There are many quality models in use today. They include process models, software models, and manufacturing models. This study researched the following major quality models:

1. Total Quality Management (TQM)
2. Total Cost of Ownership (TCO)
3. Capability Maturity Model (CMM)
4. Capability Maturity Model Integration (CMMI)
5. Control Objectives for Information and Related Technology (Cobit)
6. Information Technology Infrastructure Library (ITIL)
7. Six Sigma
8. ISO 9001
9. Malcolm Baldrige National Quality Program

Appendix E contains detailed descriptions of each of the major quality models identified.

www.manaraa.com

Comparing and Contrasting Characteristics of Quality Models

Capability Maturity Model had been in use for approximately 15 years and had been formally adopted by the U.S. DoD (SEI, 2009). Every DOD contractor was required to be at Capability Maturity Model Level 3 to qualify for a contract. Capability Maturity Model Integration, recently unveiled by the Software Engineering Institute, was a more comprehensive process-maturity model than Capability Maturity Model. It combined Capability Maturity Model along with other disciplines in systems engineering and product development. SEI would like to phase out Capability Maturity Model and replace it with Capability Maturity Model Integration. It may do so as a research institute, but Capability Maturity Model may already have an embedded market in government contracts and methodologies.

J.P. Morgan Chase had combined Capability Maturity Model, within the Six Sigma framework (Anthes, 2004). Some of the benefits cited for this combination included the following: a 20% to 25% reduction in post implementation defects; reduced efforts to support operational systems because they were more reliable with the result that emergency releases to fix bugs had fallen by 60%; better management of globally distributed projects because terminology and specifications were standardized, and better performance from suppliers because requirements were better specified. One of the challenges that lower maturity organizations had is that they did not appreciate the necessity of taking measurements and performing the analysis on the data. There were many large companies that had some business units or programs at Level 5, such as

www.manaraa.com

Boeing and Motorola Inc. These companies also had business units or programs at Level 1. On the average, the business units or programs ranged between Levels 2 through 4.

Capability Maturity Model/Capability Maturity Model Integration was not alone in this area. CobiT had well defined statements on what to do but they left it up to implementers on how to do it. CobiT, as a quality model, had not been specifically designed to apply to software development or IT services. This illustrated one of the major challenges with quality models. They were very useful if it was understood what they were trying to accomplish. For example, understanding CobiT would allow a software development organization to apply it to a software project. But trying to apply it without understanding could lead to a degraded situation. One of the other things that CobiT lacked that Capability Maturity Model supplied was how to perform the desired CQI (continuous quality/process improvement). CobiT was often used for IT Governance and Audit functions (Anthes, 2004). The positive aspect of CobiT's process statements was that they are general enough to apply to a lot of situations and organizations. The downside then was that it can take a lot of effort to modify them to fit a particular organization's processes. Lockheed Martin used CobiT as the overall quality framework. They then applied Capability Maturity Model Integration (they have four units at Capability Maturity Model Integration Level 5), Six Sigma, and ISO 9000 disciplines in various parts of their IT organization (Anthes, 2004).

ITIL, being process oriented, could also be integrated with some of the other Quality Models such as Capability Maturity Model or Capability Maturity Model Integration. ITIL was limited in that it did not address the development of quality

www.manaraa.com

management systems. ITIL components were specific to IT and the processes associated with delivering and managing services and capabilities; it was not focused on software development. While the intent of ITIL was to standardize the use of IT processes and create a common framework for IT organizations, it appeared that the implementation of ITIL was still greatly individualized by organizations. This might, and did, result in Capability Maturity Model being used solely by an IT software Development organization and ITIL being used solely by IT Service Delivery organization both of which are in the same larger IT organization, yet neither one references the other's quality model.

There is some industry data available that shows ITIL's positive effect on IT organizations. As an example, Capital One had reduced production incidents by 30% and Severity 1 incidents by 92% since they implemented ITIL (Anthes, 2004). As stated previously ITIL developed processes for delivering services in IT areas such as help desk, applications support, software distribution and customer-contact system support. It also had certain areas in its model that other models like Capability Maturity Model/Capability Maturity Model Integration had as Configuration Management (CM). CM was an ITIL process and was found in Capability Maturity Model as a Level 2 KPA. ITIL would lead to a better understanding of the need to perform root-cause analysis. As stated previously, immature organizations shied away from this activity so there might be some benefit in maturity by implementing ITIL. While ITIL identified IT processes, ISO 9000 was more applicable to certification of processes.

www.manaraa.com

Six Sigma was a quality model that incorporated statistical data techniques. To reach a Six Sigma level of quality, data must be analyzed to determine the root causes of business problems and solving them. By taking this approach, an organization could come to an understanding of the complete cost of quality. The most obvious application in IT was for common, repeatable activities such as call centers, help desk operations, or ticketing systems. Six Sigma was a manufacturing model that had been applied to IT and software. As such, the implementation would force organizations to be very specific on their requirements and designs. To accomplish Six Sigma, the IT organizations had to invest in measurement systems, use them, and maintain them. The best fit for Six Sigma was in the testing environment, but it could be used in software development in generating correct requirements. Using Six Sigma in requirements fit in with the Capability Maturity Model construct for process maturity.

ISO was a requirement to do business in the European Union (EU) and it was a diverse enough quality model that it had application enterprise wide. ISO 9001 applied to software development and could also be applicable to IT operations and services. If an IT organization desired to implement ISO, there had to be some tailoring done. ISO's approach was to have organizations implement their construct. It resulted in repeatability and consistency of processes, but, and this is a very important but, it did guarantee the quality of those processes or the products being produces. Unlike other quality models that were useful for process analysis and root cause analysis, ISO was not used for such things. The contrast was that Design for Six Sigma focused on individual projects. It

www.manaraa.com

could fix problems, and it could find ways to improve, ISO 9000s approach was to create more general and organizational level quality improvements.

The Baldrige quality program also did not address process details. It made statements about quality, but did not identify methods to achieve quality. It dealt with the entire organization; it did not specifically address IT at the highest levels. IT issues were addressed, however, by the inclusion of the IT organization within the scope of the Baldrige quality audit. Some organizations, such as Motorola, were using Capability Maturity Model, Six Sigma, which they were credited with inventing, and the Baldrige quality program. One of the uses of the Baldrige quality program was to generate balanced scorecards for executives. Gartner had created a matrix of these quality models; that matrix was shown in Figure 3, (Anthes, 2004). This matrix correlated the Quality Models specifically to IT on one axis, and showed the abstraction, or organizational detail to which the model pertains, on the other axis.

www.manaraa.com

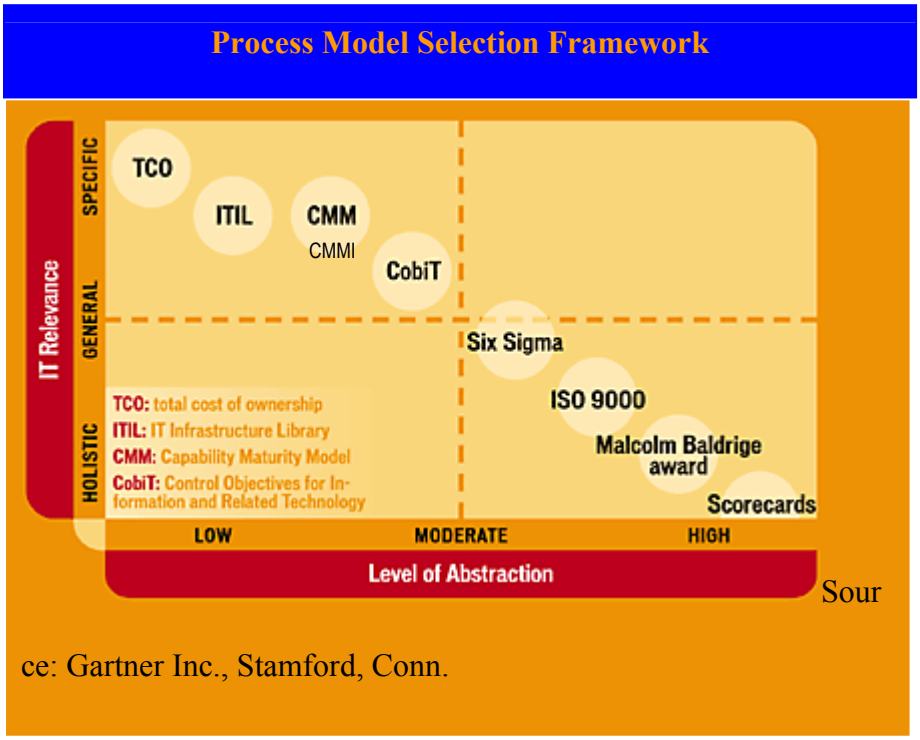

Figure 3. Comparative quality models

The x-axis (Level of Abstraction) in Figure 3, had a direct bearing on the problem statement in this research. Research question 3 addressed the concept of communication of Quality within the context of project execution. The higher the level of abstraction, the less detailed knowledge was needed about a particular instance of quality, and the more applicable it was for communication to senior management and executives. Executive scorecards were a subject of academic dialog on quality and performance, and much of the latest technology for communication within organizations was adapted to facilitate the ability to move data up the abstraction scale so it could be meaningfully communicated to senior management (Lindeman et al., 1999).

www.manaraa.com

Status of a software development project could be tracked using all the models identified in Figure 3; however the level of communication of the details of the status varied by model. The variety of the models was an indication of the attempts by scientists and engineers over the past few decades to come to grips with the issue of poor project performance in the development of software.

Quantitative Analysis of Quality Models

Figure 3 displayed the quality models analyzed in terms of monotonically decreasing level of relevance to IT and monotonically increasing level of abstraction. By converting these qualitative measures to quantitative equivalents and combining the attributes of IT Relevance and Level of Abstraction, a composite index could be produced. One way of perceiving the intersection of relevance and abstraction would be to identify usefulness from the point of view of a quality practitioner. To capture this viewpoint, a composite indicator had been created - the Quality Usability Index (QUI). The generation of a QUI was tabulated in Table A-1.

Software Development Process Models/Methodologies

Software development process models/methodologies are included in the literature search as part of answering research questions 1 and 3. Software development process models/methodologies typically had taken many of the concepts from the quality models and attempted to implement methodologies that capture the critical aspects of the quality models. This study researched the following major software development process models/methodologies:

1. Fujitsu Macroscope

www.manaraa.com

2. CMD Symphony
3. Dynamic Systems Development Model
4. Software Productivity Center (SPC)
5. Rational RUP

Appendix F contained detailed descriptions of each of the major software development process models/methodologies identified.

Comparing and Contrasting Software Development Process Methodologies

Conway asserts that most of the world's population does not have the intellectual capacity to understand process (Conway, 1998). The quality models identified previously many times have the attribute that they tell you what to do, but not how to do it. As such, many attempts to implement the quality models were unsuccessful due to lack of understanding of the sub-processes involved and their significance in the overall process. Toward this end, specific organizations had developed methodologies that had the capability to do the things that the models suggest. The reverse of this capability was that the models, while telling you what to do, did not necessarily contain the information of why it should be done.

Macroscope had a reputation for being very rigid; its whole methodology was based on the process of measuring completion of deliverables as a way of determining completion of a type of activity. When users talked about planning and schedules, the dialog did not include phrases like "business requirements," "system requirements," "design specifications," and so on. Instead the dialog was on 140s, 250s or 490s. This was because every deliverable had a number, so the number became shorthand. This

www.manaraa.com

shorthand was reinforced by the Fujitsu consultants during the methodology training. A request that was made to the researcher by a manager stuck in this methodology, was could the researcher find a way to allow his organization to be more agile and still meet the corporate standards.

RUP was a complete system that was attempting to challenge Macroscope for dominance. Given the influence of IBM, it appears to have been moving in that direction. IBM has used it as its methodology engine making it the core of what it called the IBM Software Development Platform (Rational, 2001). While Macroscope and RUP were systems in themselves, SPC was more of a conglomeration of approaches with consulting being a major part of the package. SPC used consulting, tools and training to assist the customer in a successful project implementation.

CMD focuses on Process Content Management which was specifically targeted to assist organizations in achieving Capability Maturity Model (CMM) Level 5 certification. CMD claimed that its approach would move an organization to Level 5 much quicker than through other methods. CMD was taking a specific quality model and attempting to market to organizations the capability to achieve the Capability Maturity Model goal. CMD's products worked with Microsoft Project Professional® where as RUP was part of IBM, and Macroscope had no equivalent software tool in its offering which did not allow it to offer an automated methodology. In CMD Methodology/Process infrastructure and knowledge bases are added to Task, Project and Portfolio levels of management (QuantumPM, 2004). Gartner created a matrix of these methodologies; that matrix was shown in Figure 4, as cited by Fujitsu (Macroscope, 2002). The four quadrants were

www.manaraa.com

Leaders, Challengers, Visionaries and Niche Players. This helped a project evaluate which methodology they should use depending on the market and the mission of the product under development.

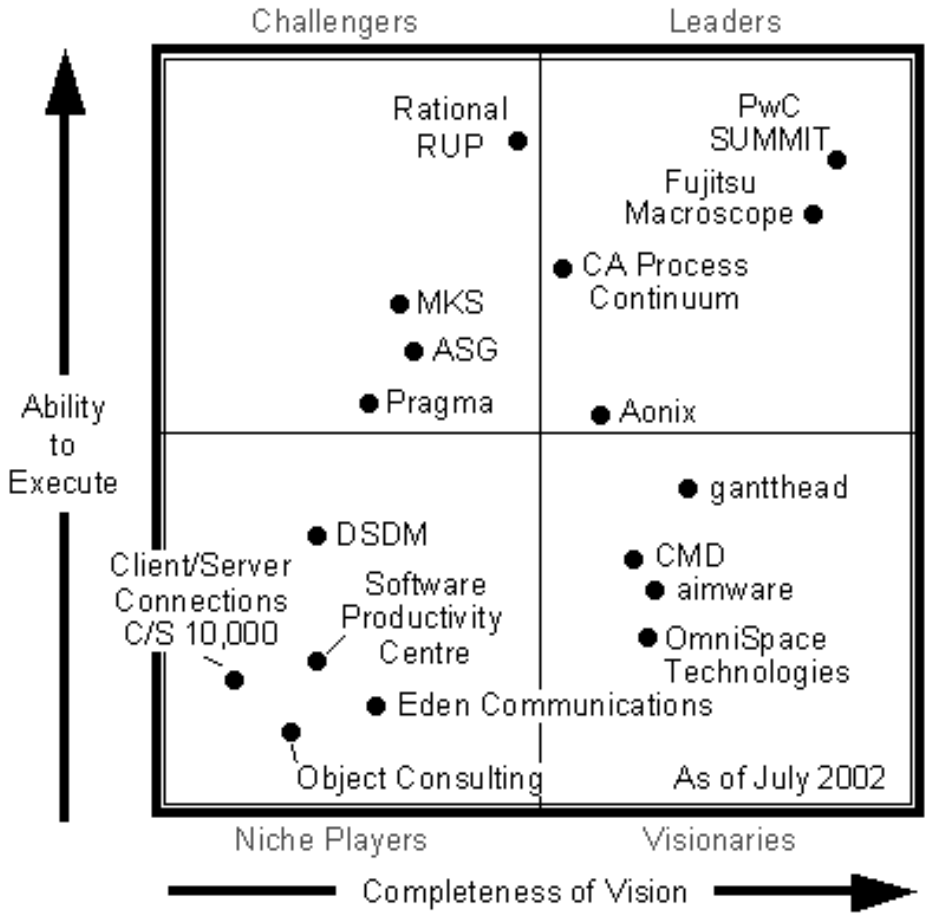

Figure 4. Comparative methodology models. Used with permission.

Quantitative Analysis of Methodology Models

Figure 4 displayed the methodology models analyzed in terms of their Completeness of Vision and their Ability to Execute. By converting these qualitative measures to quantitative equivalents and combining the attributes of Completeness of Vision and Ability to Execute, a composite index could be produced. One way of

www.manaraa.com

perceiving the intersection of vision and execution was to identify usefulness from a methodology practioner's point of view. In this study, the composite indicator that has been created has been identified as the Methodology Usability Index (MUI). The generation of an MUI was tabulated in Table A-2 (see Appendix A). The direct correlation between the Ability to Execute and the execution of project management best practices was part of the subject area of research question 1. The quality aspect of research question 2 referred to quality within the project not a quality model per se and was therefore much more aligned with the Methodology Model Vision. Macroscope had the highest MUI.

Summary of Model Quantification Analysis

The results of the quality model quantification indicated that for practioners' quality models like Capability Maturity Model, Capability Maturity Model Integration and Six Sigma were most applicable to their environment. The results of the methodology model quantification indicated that, for practioners, Methodology models like rational RUP and Macroscope were most applicable to their environment. Both of these models had a direct impact on Project Management implementation. Since the two models were complimentary and not exclusionary, the impact on the ability of an organization to address the three research questions of this study could be evaluated by analyzing the combinations possible for quality and methodology models. A Project Capability Index (PCI) was created by combining the QUI and MUI where PCI = MUI*QUI (see Table A-3). The analysis appeared to indicate that a project using a Capability Maturity Model, Capability Maturity Model Integration, Six Sigma and/or

www.manaraa.com

CobiT Quality Model and a Macroscope Methodology would be best qualified to participate in this research study.

Measurement

Measurement is included in the literature search as part of answering research questions 1 and 2. The University of Magdeburg described measurement as process of assigning numbers or symbols to attributes of entities in the real world that describes them by clearly defined rules (SML, 2004). Some of the most common categories of metrics included Product, Process, Objective, Subjective, Direct, Indirect, Explicit, Derived, Absolute, Relative, Dynamic, Static, Predictive, and Explanatory metrics (SML, 2004). The three major categories of software metrics were product, process and project metrics (Kan, 2002). Software quality metrics dealt with two major categories of concern. The first was intrinsic product quality and the second was customer satisfaction (Kan, 2002). In evaluating project metrics, this study focused on a specific project metric used in software development projects called Earned Value Management (EVM).

Measures and Metrics Common Definition

Software metrics relied on the underlying theory of representational measurement (Orci, 1999), which was a technical discipline of assigning a number, or symbol, to an entity in order to characterize one of its properties. This must have been done within specific rules. In the definition there needed to be an entity, a property, a measurement mapping and rules for the mapping. The measurement mapping and the rules were designated as the metric. The need for a theory of measurement allows for a safe acquiring and reproducibility of measuring characteristics (Hille, 1997).

www.manaraa.com

Often, people used the terms measures and metrics synonymously. A common industry approach defined a measure as a quantity and a metric as a comparison of quantities. For example (ME1, 2005):

- Measure (measurement of quantity): A single item that can be quantified (length, weight, count, volume…) - An attribute of a product or process.
- Metric (one indicator of quantity relationships): A graphic that usually combines two or more measures - Related to something important such as health of a product or process or progress toward a goal or limit. It is designed to yield information and stimulate questions.

An example was provided here:

Measures: Gas Tank Capacity and Current Contents

Metric: Gas Gauge in your car displays the percent full of your gas tank.

University of Southern California (USC). USC defined a metric as a characteristic of a process or product (USC, 2001). Metrics can either be directly observable quantities, or can be derived from one or more directly observable quantities. In USC's case, they equated what we've commonly defined as a measure to a raw metric and what we've commonly defined as a metric to a derived metric.

Planguage. Gilb developed some very detailed concepts that he used in his approach to Planguage (Gilb, 2005, p. 399).

Metric Concept *095

A metric is any kind of numerically expressed system attribute. A metric is defined in terms of a specified scale of measure, and usually one or more numeric

www.manaraa.com

points on that scale. The numeric points can be expressed with defined terms that can be translated into numbers. For example, 'Record +10%.' Normally there will also be other parameters and qualifiers, which add background detail to the metric. For example, Meter and Assumption.

A metric specification encompasses all related elements of specification, not just the Scale of the numeric attribute. A complex specification, with a set of scales of measure, is also a metric expression. There is no implication that it is elementary (has only a single Scale).

The term "indicator" was used to denote a representation of metric data that provided insight into an ongoing software development project or process improvement activity. Indicators were metrics in a form suitable for assessing project behavior or process improvement. For example, an indicator may have been the behavior of a metric over time or the ratio of two metrics. Indicators may have included the comparison of actual values versus the plan, project stability metrics, or quality metrics. Examples of indicators used on a project included actual versus planned task completions, actual versus planned staffing, number of trouble reports written and resolved over time, and number of requirements changes over time. Indicators were used in conjunction with one another to provide a more complete picture of project or organization behavior. For example, a progress indicator is related to requirements and size indicators. All three indicators should be used and interpreted together.

www.manaraa.com

Software Measurement

The purpose of software measurement was to understand and control the process and its products (SML, 2004). This was done by continuously defining, collecting, and analyzing data on the software development processes and products. Software measurement was a key approach to moving up the Capability Maturity Model Integration maturity scale; it supplied the meaningful information needed to allow the improvement of software development processes (SML, 2004).

Components of Software Measurement (SML, 2004)

There were three major components of software measurement: products (see Table 3), process (see Table 4) and resources (see Table 5).

Table 3.

Components of Software Measurement – Products

Entities	Internal Attributes	External Attributes
Specifications	Size, reuse, modularity, redundancy, functionality, syntactic correctness,	Comprehensibility, maintainability
Designs	Size, reuse, modularity, coupling, cohesiveness, functionality, ...	Quality, complexity, maintainability
Code	Size, reuse, modularity, coupling, functionality, algorithmic complexity, control-flow structuredness	Reliability, usability, maintainability
Test data	Size, coverage, level	Quality

www.manaraa.com

Table 4.

Components of Software Measurement – Process

Entities	Internal Attributes	External Attributes
Constructing Specification	Time, effort, number of coding faults found, …	Quality, cost, stability
Detailed design	Time, effort, number of specification faults found...	Cost, cost-effectiveness
Testing	Time, effort, number of coding faults found...	Cost, cost-effectiveness, stability, etc…

Table 5.

Components of Software Measurement - Resources

Entities	Internal Attributes	External Attributes
Personnel	Age, price	Productivity, experience, intelligence
Teams	Size, communication level, structuredness ...	Productivity, quality ...
Software	Price, size ...	Usability, reliability
Hardware	Price, speed, memory size ...	Reliability
Offices	Size, temperature, light, ...	Comfort, quality

Examples of software measures included the number of source lines of code, number of defects, number of test cases, number of documentation pages, number of staff-hours, number of tests run, number of requirements, etc. All these could become metrics if they were compared over time. Examples of software metrics included defects per error category, source lines of code per staff-hour, defects per thousand lines of code, a cost performance index (BCWP, ACWP, BCWS, ACWS) or Earned Value Indices (CPI and SPI). This study researched the additional following metrics:

1. Macroscope Metrics

www.manaraa.com

2. ITIL Metrics
3. Progress Metrics
4. Effort Metrics
5. Cost Metrics
6. Results Metrics
7. Trouble Reports (TR) Metrics
8. Requirements Stability Metrics
9. Size Stability Metrics
10. Computer Resource Utilization
11. Training Metrics

Appendix G contains detailed descriptions of these metrics listed.

Goal/Question/Metric (GQM)

The GQM model for metrics development was a bottom up approach where metrics were used for process improvement. This was in contrast to a top down approach of process improvement such as Capability Maturity Model Integration. Goal-oriented measurement focused on the explicitly stated goal as the highest importance for improvement programs. Basili from the University of Maryland in conjunction with NASA developed a software productivity laboratory in the 1980s timeframe. This resulted in developing the GQM approach. According to Basili, as cited by Software Measurement Laboratory (SML, 2004), the GQM methodology is a systematic approach for integrating goals to models of the software processes, products and quality

www.manaraa.com

perspectives of interest. This is based upon the specific needs of the project and the organization (Basili et al, 1994).

According to NASA, GQM defined a measurement model on three levels (NASA, 2006): Conceptual level (goal) - A goal was defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view, and relative to a particular environment; Operational level (question) - A set of questions was used to define models of the object of study and then focused on that object to characterize the assessment or achievement of a specific goal, and Quantitative level (metric) - A set of metrics, based on the models, was associated with every question in order to answer it in a measurable way. To improve processes by GQM, the measurement goals needed to be defined. Then from the goals questions were generated that if answered would show the progress towards the goals. Finally metrics were identified that would supply all the necessary information for answering those questions. The GQM approach provided a framework involving three steps as shown in Figure 5.

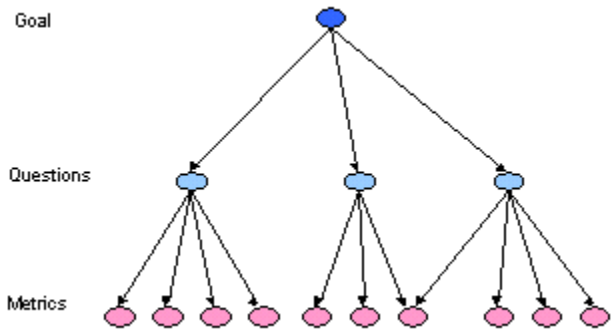

Figure 5. Goal/questions/metrics methodology

www.manaraa.com

Software Development Process Metrics

Another approach at software development metrics involved the identification of metrics with processes. Software metrics as numerical data related to software development, were required to support software project management activities. Metrics could be classified with respect to the functions of management as follows: Planning - metrics served as a basis of cost estimating, training planning, resource planning, scheduling, and budgeting; Organizing - Size and schedule metrics influenced a project's organization; Controlling - metrics were used to provide status and track software development activities for compliance to plans; Improving - metrics were used as a tool for process improvement and to identify where improvement efforts should be concentrated and measure the effects of process improvement efforts. A set of representative Project Metrics includes the following indicator categories: progress, effort, cost, review results, trouble reports, requirements stability, size stability, computer resource utilization, and training (USC, 2001).

SEI Goal Driven Software (DOD)

The Software Engineering Institute developed a metrics approach at taxpayer expense that integrated with its Capability Maturity Model Integration maturity model and supported government procurement activities. There were four reasons to measure software processes, products, and resources (Park, Goethert, & Florac, 1996): to characterize, to evaluate, to predict, and to improve. Characterization gained understanding of processes, products, resources, and environments. It established baselines for comparisons with future assessments. Evaluation determined status with

www.manaraa.com

respect to plans. Measures were like sensors that let software professionals keep projects and processes under control. With evaluation, achievement of quality goals, the impacts of technology and process improvements on products and processes could be assessed. Prediction allowed planning. Measuring for prediction involved gaining understandings of relationships among processes and products, building models of these relationships, observing values of attributes, and using values to predict others.

This was done to establish achievable goals for cost, schedule, and quality—so that appropriate resources could be applied. Predictive measures were also the basis for extrapolating trends, so estimates for cost, time, and quality could be updated based on current evidence. Projections and estimates based on historical data also helped risks to be analyzed and design/cost tradeoffs to be made. Improvement occurred when quantitative information was gathered. This allowed the identification of roadblocks, root causes, inefficiencies, and other product quality and process performance improvement opportunities. Measures allowed planning and tracking of improvement efforts. Measures of current performance created comparison baselines and improved communications for product and process improvement.

Software Inspections

Overview

Software inspections are included in the literature search as part of answering research questions 2 and 3. The formalized development of software inspections was attributed to Fagan's efforts while working at IBM (Gilb, 1988). Software inspections became an IBM best practice, but also met their first resistance. Later, Bell Labs started

www.manaraa.com

using the process and reported a productivity improvement of 14% (Gilb & Graham, 1993). A software inspection is a formal static analysis technique that uses well-defined methods to find defects in software design and development documents. The basic phases of an inspection include (BDS, 2001):

1. Select reviewers from the authoring group, peers, stakeholders, and subject matter experts.
2. Reviewers examine the product using an inspection checklist and list errors.
3. Errors are reported and recorded in error logging meetings.
4. The error logs are used to correct products before delivery to stakeholders.
5. Metrics are collected about the inspection process itself and the errors logged.

The Inspection Process

The inspection process consisted of 10 steps broken up into the following three major sections (Gilb & Graham, 1993): Initiation and Documents – included request for inspection, the planning process, documents needed for the inspection of the product, the entry process and the kickoff meeting; Checking – included individual checking and the logging meeting, and Completion – included edit, follow-up and exit. Other terms for the logging meeting included inspection meeting (Houdek, Schwinn & Ernst 2002), and error logging meeting (BDS, 2001). The inspection process did not need to be restricted to just software artifacts. Other industries were also adopting the process. Hewlett Packard, which had institutionalized software inspections, had also transitioned the process to some of their hardware products (Gilray, 1996). The inspection method for drawings was institutionalized at Boeing with Gilb's assistance. It was called PEP (Process Error

www.manaraa.com

Prevention) and was so successful, that other software groups picked it up within Boeing and reconverted the process to apply to the software development process (B-SEPG, 2004).

Defect Detection

The effectiveness of defect detection was due to a combination of factors. The Software Inspection Process was perceived by some as a process that once put in place would run smoothly producing consistent results. The effects of the qualifications of the inspectors and the characteristics of the document were still not well understood among the general population that was using inspections (Rus, Halling, & Biffl, 2003).

Defect Density

The defect density was used in two different frameworks. The first deals with the found defects. These types of defect densities were gathered during the Software Inspection process, and could be measured in the following terms (Gilb & Graham, 1993): total number of defects per document size (pages); defect density (defects/page) per document size (pages); total defects per inspection duration (minutes), and defect density (defects/page) per inspection rate (pages/hour). Observations made about defect density in detected defects included the following (Gilb & Graham, 1993): inspectors reviewed large documents at a higher number of pages per hour than a corresponding document with fewer pages; the number of defects per minute when compared to inspection durations appeared to be linear up to four hours; the defect density dropped sharply as the scope of the material being reviewed increased.

www.manaraa.com

The second approach addressed the number of defects in an artifact, whether a document or the code. The results of a Software Inspection may be used to attempt to determine how many total defects there were originally in the product and how many still remained. The approach to determining remaining defects was called RDET (remaining defect estimation technique) (Houdek, Schwinn, & Ernst, 2002). One of the approaches suggested was taken from what initially developed in the science of Biology. Originally scientists used this technique to estimate total animal populations (Houdek, Schwinn, & Ernst, 2002). The RDET Based on Animal Sampling Technique equation is $n_{total} = (n_1 * n_2)/n_{both}$ where n_i = number of animal observed on day i, n_{both} = number of animals seen on both days and n_{total} = the estimate of the total population.

This RDET was particularly attractive for its innovation, but may not have been able to be readily adapted by Software Development organizations. There did not appear to be any support in the literature for re-reviewing an artifact and comparing the two results. The ability of Inspections to detect defects against the total available would be another definition of Defect Density. Gilb claimed in 1995 that 95% of all defects can be removed by using inspections (Software Testing/Quality Conference, 1995). There seemed to be some industry support for this assertion, at least for large unprecedented software intensive systems like the Space Shuttle where human life was at stake and the system options in case of failure were few. Table 6 identified the software development sub-processes where defects were found in the Space Shuttle software (Billings & Clifton, 1994).

www.manaraa.com

Table 6.

Error Source in On-Board Space Shuttle Software

Software Development Sub-Phase	Percentage of Defects of Total Found
Pre-Build Inspections	85.4%
Other Inspections	7.3%
Testing	7.2%
In Production	0%

Agile reviews/extreme inspections have been developed to obtain results more quickly with fewer resources needed than with traditional software inspections (Gilb & Gilb, 2004). A method of quickly determining an engineering estimate of defects has been anecdotally devised and tested in industry. The process was for knowledgeable managers to gather in a meeting where a couple of pages from one of the project requirements or specifications were inspected. The inspection lasted no more than 30 minutes, and no definition of major defect was given. Each knowledgeable person determined themselves what was major and what was not. After the time was up, the number of major and total defects was to be tabulated. Table 7 showed an example using the algorithm to calculate the total number of defects per page. Major IT and Embedded software corporations had used this method with results matching those predicted by the method at Microsoft (Finn, 2001) and Intel (Simmons, 2002).

www.manaraa.com

Table 7.

Agile Review/Extreme Inspection Sample Error Logging Results

	Total Defects Found	Major Defects Found	Design Items not Requirements
Person A	24	15	5
Person B	44	15	19
Person C	55	20	4
Person D	22	4	2

Process Improvement

While Software Inspections were still commonly thought of in terms of defect detection and correction, there was some significant work being done in the area of using software inspection data for process improvement. At the beginning of the Space Shuttle program, peer reviews and inspections were implemented (Billings & Clifton, 1994). Based on the defect data gathered in the 1970s, IBM was able to use the severe and critical error data to develop a Defect Prevention Process based on audits and data analysis (Billings & Clifton, 1994). As shown in Table 6, there were no software defects discovered in flight. Many of the articles cited have alluded to process improvement as a result of performing inspections. Most, however, did not specifically identify the processes involved. Those who did usually alluded to a database of errors, performing root cause analysis and somehow improving the process.

Additional Innovations in Inspections

Electronic Meeting Systems (EMS). One of the logical fallouts to the use of inspections in the 21st century with a mobile and virtual workforce was the use of

www.manaraa.com

Electronic Meeting Systems (EMS) to hold the error logging meeting. Phillips Medical Systems and the Baan Company used an EMS for 14 electronic inspections. The initial results indicated that using EMS gave much more significance to the error logging meeting than the traditional face-to-face meeting (Van Genuchten, Cornelissen, & Van Dijk, 1998). They concluded that using electronic support for the logging meeting may improve the effectiveness and efficiency of the inspections

Six Sigma, SEI Capability Maturity Model and Software Inspections. The Motorola Company initiated a drive to get to a quality level of Six Sigma and CMM Level 3. The specific goals were as follows (Major, Pellegrin, & Pittler, 1998): Quality - 10-times improvement in quality every two years; Customer Satisfaction - exceeding customer expectations and all competition; Cycle Time - 10-times improvement in five years; Software Technology Roadmap - all organizations created, qualified and executed their own Software Engineering Technology Roadmap process, and Process Capability - all organizations achieved SEI Maturity Level 3. The results were that 75% of the software organizations were successful in achieving SEI Capability Maturity Model Level 3, and listed first in the list of Major Initiatives responsible for achieving the goal was rigorous inspections (Major, Pellegrin, & Pittler, 1998).

High Quality Low Cost Software Inspections. Hedger, in reviewing Radice's book *High Quality Low Cost Software Inspections* (Hedger, R., 2003), includes a history of software inspections and details of how to efficiently execute the software inspection process. The innovation in this approach was the re-emphasis on the basics underlying the process and how they could be more effectively used to reduce the cost of ensuring

www.manaraa.com

quality in the software product. The innovations and efficiencies were limited to the inspection process itself.

Summary

Software inspections have been around for more than 30 years. They were still thought of as a method to find and fix defects. This was the old style of quality. It was basically reject and rework. It did produce a better product, but the rework was done when the product was finished. The reason that even this was advantageous was that while the inspection of the software artifact might be at the end of its process, if the software artifact was at or near the beginning of the software development process, then finding errors from the software product at the beginning of the process ($90-$120) could save hundreds or thousands of hours of rework ($10,000) in the testing or customer delivery phases of the project (Bush, 1990). The literature and industry practice indicated, however, that there was additional activity and energy being focused on the data that was captured when defect metrics from the inspections are kept and analyzed. The data had the great potential to lead software organizations into process improvement and defect prevention which was the most cost effective type of quality.

Earned Value Management (EVM)

EVM Overview

EVM is included in the literature search as part of answering research questions 2 and 3. A project was composed of four elements: schedule time, cost resources, technical performance and risk (NASA, 2007). EVM had a schedule variance component, which involved a time sample of the variance between the work planned and work

www.manaraa.com

accomplished, was described in terms of cost (NASA, 2007). EVM also had a cost variance component, which involved a time sample of the variance between the actual cost and work accomplished, was described in terms of cost (NASA, 2007).

EVM Background

The implicit assumption was that the technical aspect of the work being accomplished was being accurately measured, so that if a deliverable was done within cost and schedule, it had been a successful effort for the project manager. The current implementation of EVM recognized that there was a technical performance element to project management, but did not allocate an earned value variance metric for technical performance. The use of earned value analysis (EVA) or EVM was first practiced by industrial engineers in factories in the late 1800s (Anbari, 2003). Starting in the 1960's, with the advancement of continually more complex projects the US Department of Defense introduced the concept of PERT networks (Abba, 1997). This was followed by the cost/schedule planning and control specification or C/SPEC which then became the basis for the cost/schedule control systems criteria or C/SCSC.

In the 1960s, due to increasing complexity of projects, the military initiated a spend plan approach. This led to the development of the PERT network. A resource loaded PERT network evolved into the first attempt at earned value. The US Air Force first used earned value on the Minuteman program in 1963. EVM was defined using 35 criteria which were organized into five basic management principles. The 1970s and 1980s brought a consolidation of approaches within the government that standardized the EVM concepts and management styles. The concept of a planning horizon was

www.manaraa.com

introduced. A C/SCSC Joint Implementation guide was published. Out of this evolved the concept of total quality management (TQM). In the 1990s project solution environment EVM became a part of the regulatory requirements for government acquisition. EVM facilitated the advance of integrated baseline reviews (IBR) and was essential in the development of international project management principles (Abba, 1997).

C/SCSC identified 35 criteria which were organized into the following five areas: organization and integration of people and work; planning and budgeting; accounting; analysis, and revisions (Abba, 1997). The relationships of the elements of EVM are shown in Table 8 (Blanco, 2003).

Table 8.

Earned Value Element Relationships

	Scope	Schedule	Budget
Work Planned	What work was scheduled?	When is it scheduled?	How much is budgeted?
Work Completed	What work was done? How much was actually spent?	When is it done?	How much was it budgeted for?

EVM Key Components

EVM was a quantitative approach to project management. It purported to measure the true performance of a project by calculating cost and schedule deviation as well as predicting actual completion times and costs (Brandon Jr., 1998). The EVM key components (Anbari, 2003) were itemized in Table A-5 (see Appendix A).

www.manaraa.com

Challenges Associated With Implementing EVM. The following were the major challenges to the successful commercial implementation of EVM (Brandon Jr., 1998): purely commercial enterprises had a minimal awareness of the technique; getting data, especially percent complete and actual cost, if done according to the strict methodology was not cost effective; reporting was not easily implemented, and project resistance to implementing EVM.

How Data Acquisition Challenges Can Be Overcome. The less intrusive the process, the easier data acquisition will be (Brandon Jr., 1998). If the work packages were not sized correctly, then the smaller they get, the more reporting had to be done which takes up more personnel time. Calculating percent complete was very time consuming. A shortcut that would give a close approximation, assuming that the work packages were properly sized, would be to make close estimates rather than stringent calculations (Brandon Jr., 1998). Typically projects had to report on a weekly basis. If the average work packet was also sized at a week, and assuming that the current tasks were estimated at 50% in error, then the maximum error would be about 1%. This was derived using the formula of [(average packets per week*average cost per packet*0.5)/ (total cost)] (Brandon Jr., 1998).

How Reporting Challenges Can Be Overcome. The barriers to getting correct reports out of EV fell into three major areas (Brandon Jr., 1998): getting actual costs; setting up automatic interfaces between the project and corporate systems, and EVM mechanisms were usually not straightforward and simple. If the project was not in

www.manaraa.com

serious trouble, then reporting the actual costs wasn't necessarily required. Only when a project was in trouble would the management need to drill down to the cost details.

How Employee/Contractor Resistance Challenges Can Be Overcome. Some people viewed EV as a way of measuring their productivity and thus impacted their rewards instead of viewing it as a project management tool. Brandon Jr. recommended separating EV from quality programs, thus making it clear to the project personnel that EV was not part of the quality evaluations.

Graphics

The earned value graphics were usually depicted with one axis being time in increments of reporting periods, and the other axis being either money or resource hours. NASA managed to money at their level. Other Earned Value management techniques had the vertical axis in labor hours; that was what an IT project Manager often managed to. The display of Earned Value information could produce data rich graphics. A typical Cumulative Earned Value Report from created by Primavera had the following information displayed simultaneously in graphical form with the y-axis being hours and the x-axis being time units, typically weeks or months (Primavera, 2005): Planned Value (PV) – a cumulative line that started at zero on the left side of the graph and continued to accumulate in value until the last week is reached on the right hand side of the graph; it was planned at the beginning of the period under examination; actual value (AV) - a cumulative line that started at zero on the left side of the graph and continued to accumulate in value until the current status week; current or estimate to complete (EAC) – a cumulative line that started where the AV ends and projected the completion status

www.manaraa.com

given current trends; earned value (EV) - a cumulative line that started at zero on the left side of the graph and continued to accumulate in value until the current status week; it was the amount of hours budgeted for the work that got performed; percent complete – a cumulative line that started at zero on the left side of the graph and continued to accumulate in value until the current status week; it was the percent of work completed; budget at completion (BAC) – A total line that started from the left and continued to be calculated each time period; it was the total budget baseline, and estimate at completion (EAC) - A total line that started from the left and continued to be calculated each time period; it was the Actual to date plus the estimate to complete the remaining work.

A summary of the EVM status was displayed graphically with the use of performance indexes. When displayed, the y-axis was centered at 1.0 and performance was tracked during the time intervals from left to right. Both CPI and SPI could be graphed simultaneously. A value of 1.0 was desired. FigureB-3, (see Appendix B) displayed the various EV parameters in graphical form. The cost variance (CV) and the schedule variance (SV) were displayed graphically. As with any cumulative chart or worm chart, besides just seeing the current values, the trend also was displayed. The trend could be used to predict the future to the extent that a trend would not change quickly without adding additional resources or de-scoping the activity. The S-curve had the limitations that project managers often will not see the problems starting to occur because the deviations were so small in absolute terms that they did not seem significant although if they were displayed in SPI or CPI numbers, they would be seen to be significant. Companies that presently use the cost/schedule control systems criteria

www.manaraa.com

(C/SCSC) according to exact regulations are not able to institute early (and therefore meaningful) corrective actions on possible cost overruns. This also concurs with government published data that recommend the graphing of earned value via an S curve plotted against monthly durations, or calendar month designations (Cass, 2000).

Further Refinements of CPI and SPI

With modern technological advances, EV could be further utilized below the project level. It was possible to apply EV at the sub-project or milestone level. The overall EV of a project examined the cumulative cost over the lifetime of the project. Within each project were many milestones. There were CPIs and SPIs for both cases. The following C/SPIs have been identified (Chang, 2001): S1: measures period schedule performance – Project Level; S2: measures total schedule performance – Project Level; C1: measures period cost – Project Level; C2: measures total cost – Project Level; S3: measures inception to-date schedule performance – Milestone Level; S4: measures milestone schedule performance – Milestone Level; C3: measures inception to-date milestone cost – Milestone Level, and C4: measures total milestone cost – Milestone Level. The characteristics of these C/SPI combinations were summarized in Table A-6, (see Appendix A). Using these lower level CPIs and SPIs, a project manager could track the status of blockpoint releases within a project as well as track to overall status of the project.

Some experts preferred the use of CR instead of CPI. They identified five strategies that a project manager may adopt to deal with a bad project situation (Evensmo & Karlsen, 2004): laissez-faire strategy – do not change anything; Santa Claus strategy -

www.manaraa.com

try to maintain original schedule and budget by becoming more efficient; tail between the legs strategy - try to maintain original schedule and budget by reducing scope; age before beauty strategy - try to maintain original schedule and budget by adding resources, and late sunset strategy - try to maintain original planned costs by extending the schedule. Evensmo and Karlsen's emphasis was on the use of CR as opposed to just CPI or SPI. They acknowledged that their use of CR was not based on firm theory, but they remained of the opinion that it was an important tool.

Defect Performance Index (DPI)

DPI (Olson, 2008) is a method of measuring project performance currently being used by Olson, President of the Lean Solutions Institute. The method was proprietary, but the author was authorized to generically describe it. A model of projected defects in the software development phases was compared against actual defects found in software development phases using the formal software Inspections methodology (Gilb & Graham, 1993). This resulted in a DPI, the application of which to project management remained proprietary.

Performance-Based- Earned Value Management

Another variation on a theme was an approach to Earned Value Management called Performance Based-Earned Management (Solomon & Young, 2006). A standard Earned Value Management System (EVMS) had limitations with respect to both standards and models for systems engineering, software engineering, and project management (Solomon, 2006). The GAO reported on deficiencies in the application of

www.manaraa.com

EVMS (GAO, 2006) in the Ballistic Missile Defense System (BMDS) program of the Missile Defense Agency (MDA). Four areas of deficiency were identified:

1. Deferred Functionality - Contractor did not track the cost of work that was deferred from one block to another. As a result, the cost of the first block was understated and the cost of the second block was overstated.
2. LOE - Prime contractors incorrectly planned discrete work as level of effort (LOE). The program lost is ability to gauge performance and to make adjustments that might prevent cost growth.
3. Re-baseline - The cumulative performance of one contractor was distorted because it re-baselined part of its work. When variances are set to zero, the cumulative performance of the contractor appeared more positive than it was.
4. Award Fee - MDA rated a contractor's cost management as outstanding and awarded 100%of the related fee although earned value data indicated that the contractor overran its budget.

Performance based EVM as documented in the book *Performance-Based Earned Value* asserted that it provided guidelines and examples that will ensure that the EVM information was accurate and reliable; the EV was based on technical performance or quality, and a program using Performance-Based EVM would not have EVM deficiencies reported by the GAO (Solomon, 2006).

Comparison and Contrast of Quality Project Management Approaches

This research identified some major categories of approaches to ensuring that a software development project is successful. These major categories included but are not

www.manaraa.com

limited to the following: quality models, software development process methodologies, measurement/metrics, software inspections, and earned value management. In reviewing these categories for application to our study research questions, it was imperative to consider the component effect of the executive in the communication status with the purpose of effective and successful management of the project. When people in technical disciplines are communicating with executives, it is critical to keep the following constraints foremost in presentation of information (Tripathy, 2007*)*:

1. Executive communication should have a sales orientation. The communicator should think like a sales person when developing the communication. It should have been reviewed by the communicator as if he were an editor.
2. The communication should contain high level summaries of key ideas or solutions you are proposing. They should be presented in a logical flow. Each idea should be present in at the most two or three sentences or bullet points.
3. If necessary, the communicator can give links or references to topics where the ideas that are presented may benefit from more detailed explanations.
4. Use the right word, right statement, and right flavor in your communications. They should reflect clarity of thought and logic, and be without redundancy of unnecessary content.
5. Use heading and subheadings judiciously. Try presenting a long list of information in bulleted list.

www.manaraa.com

NOTE: The author of this study found from many years of experience in communication, use of numbered lists enhances communication efficiencies to a greater degree than bulleted lists. In communication, it is much easier and less ambiguous to refer to Item 9 rather than the Ninth bullet down from the top.

6. Avoid repetition of ideas, jargons, phrases, ambiguity.
7. Read, re-read and if possible rewrite the executive communication before sending it for review by others.
8. Try creating at least two or three different executive communications and choose the best one among them.
9. Avoid using graphics or information that requires graphics to understand.
10. Review the communication from the executive's viewpoint. Do you see it addressing all of your concerns? If yes, you have written the right stuff. If no, rework.
11. Check the following: grammatical and spelling errors, consistency in tense, consistency in content (data and facts), and stick to one version of English language.

As previously noted, this study had three research questions. The first research question focused on Project Management. The second research question focused on Quality. The third research question focused on Executive Communication. In Table A-4, (see Appendix A), each approach was assigned a value from 0 to 5 in each of the three research question emphasis areas with respect to how thoroughly they cover the area.

www.manaraa.com

This coverage number was based on the research performed in that area. Then an Effectiveness Penetration Index was calculated by averaging the coverage values. A Pareto analysis of EPI approaches was shown in Figure 6. It appeared that Defect Performance Index, Performance based-Earned Value Management, Earned Value Management, Software Inspections, Macroscope, Malcolm Baldrige and Requirements Stability Metrics had the most affinity towards the goals of this study.

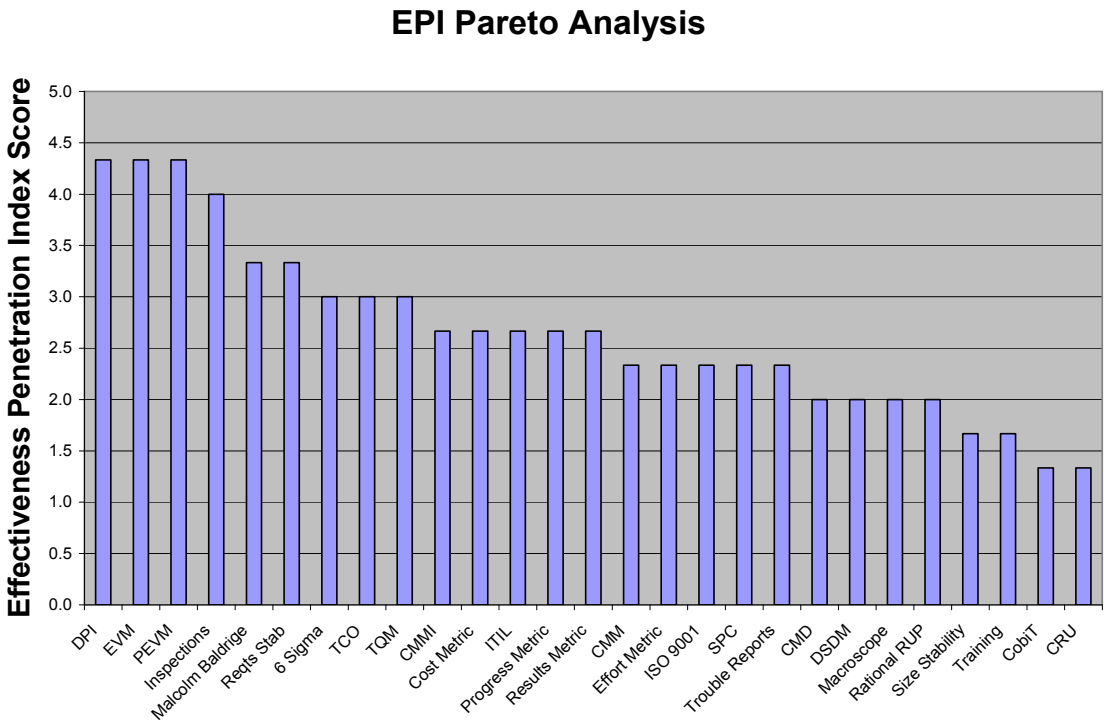

Figure 6. EPI Pareto analysis.

Conclusion

The amount of data presented in the research was an attempt to demonstrate that the areas of needed emphasis indentified in the NATO report almost 40 years ago have not gone unnoticed. Five major areas of endeavor were identified: quality models,

www.manaraa.com

development methodologies, metrics, inspections and Earned Value Management. This was not an exhaustive list, nor were the unique approaches identified in each area complete. They were a small but representative sample. If the areas and approaches were further examined, it could be determined that the audience they were communicating to was not the same and the viewpoint of the importance of the information presented by each specific approach could vary. The research contained herein then, did in fact reinforce the following assertions/conclusions:

1. The problem areas identified in the NATO report are still occurring.
2. An entire support industry has sprung up to attempt to solve the problems identified by the NATO report.
3. There was no unified theory or implementation approach across the various areas of emphasis.
4. There was still a communication gap between the technology experts/implementers and the managers and executives with organizational fiduciary responsibility. The gap was demonstrated in the various viewpoints of the information presentation in the identified areas.
5. The literature research analysis performed in this study as summarized and depicted in Figure 6, provided a viable framework for development of an integrated methodology that addressed the research questions of this study.

These conclusions formed the framework for the chapter 3 dialog which examined the possibilities of developing a methodology for measuring project status based on the analysis and downsizing selection of the data from the literature search in

www.manaraa.com

Chapter 2. Chapter 3 introduced the research concepts of using agile inspections to implement a quality component in earned value project management at the beginning of a project before testing occurs.

Summary

This research has just barely skimmed the surface on the subject of software development metrics. The GQM approach was included to show the level of research and empirical application that has been conducted with academia and governmental cooperation. The ITIL and Macroscope areas were included to specifically address the Information Technology approach to metrics management and usage. It is different from embedded systems and governmental acquisitions. General descriptions were also included from industry and academic class work to show what some of the more advanced practical applications of metrics are evolving to.

Earned Value is a set of metrics that show the status of the process of executing a project. The parameters that are examined are cost and schedule status over time. Earned Value has both a graphical and mathematical component. Part of the ability to use Earned Value to best advantage is the careful selection of the graphical displays. Selecting the correct parameter to focus on is critical to understanding the predictive elements of Earned Value. Earned Value is generally accepted as a project must for large programs and DOD/Government funded programs. It is making some head way into the commercial marketplace, and will be more commonly accepted as new technology and tools to ease the overhead burden are introduced into the marketplace.

www.manaraa.com

In many of the technical, professional and management forums, there is a never-ending dialog concerning what is the best approach for a quality model or a process methodology. This research seemed to indicate that there is not common agreement on a best approach. The mission of the product needs to be taken into consideration. The mission of the organization needs to be considered also. In the case of supplier provided Process Methodologies, supplier viability and methodology cost need to be considered. A general observation is that the quality models are more technically based where as the process methodologies are more business oriented. The businesses attempt to develop methodologies that implement what is technically recommended but in a practical way.

Chapter 3 identifies the method that is used in the research. It further develops the application of some of the software model aspects found in this chapter, both process and project.

www.manaraa.com

Chapter 3: Methodology

As previously stated in chapter 2, software projects continue to fail most often due to project management quality issues. Since the NATO report was published in 1968, software quality has been an elusive goal for many IT, military, and government organizations. This chapter contained the research design which includes the subjects of quantitative approach, population, reliability, validity, data collection and analysis, sampling and sample validity. There is also an examination of EVM as a project management methodology with adaptations for the research in this study followed by a chapter summary.

Research Design

Quantitative Approach

There are three major categories of research approaches: quantitative, qualitative and mixed methods. This study used the quantitative approach to project status; it did not look at the qualitative aspects of presenting project status since that was what oftentimes put a project manager in conflict between the advertised qualitative status and the actual quantitative status. The logical approach was deductive as opposed to inductive. The quantitative effect that was observed was the impact of defects on schedule and cost factors within a software project. Consistent with the quantitative approach, this research presupposed that the results garnered from the specifics of the project under study could be successfully generalized to the broader community of software projects (Ross, 1999).

More specifically, this study used a quantitative, post-facto, exploratory, research design. Statistical tools and techniques were used to examine the project data after it was

www.manaraa.com

completed. Post-facto studies investigate possible cause and effect relationships by observing an existing condition or state of affairs and searching back in time for plausible causal factors (Sinks, 2007). Because the events and independent variables had already occurred, the study was non-experimental (KovacsBurns, 2005).

The data that was analyzed post-facto were requirements documents that were generated during the process of producing a software product. The research descriptive in nature although an argument could be made that it was explicative; the determination would be to what extent an existing method has been modified or whether the new element added was significant enough to stand alone.

The experimental design could not be employed. The experimental design required control populations and this option was not available. Most software projects were under budget and time constraints and the project data made available to the researcher was no exception. Management was willing to allow evaluation of project data already completed, but was not willing to invest in experiments on projects that already had cost and schedule commitments that were thought to be adequate with the current processes that were in place. Thus an experimental approach was not possible; as stated previously, the non-experimental design was used.

The qualitative approach and the mixed methods approach were not taken for the following reasons:

1. Since the research approach was quantitative, there was no requirement for information from participants in the initial phase of the research. Follow-on studies might require participant input, but this initial phase did not.

www.manaraa.com

2. The research was noninvasive into the processes of the projects doing development. This was intentional in that it was the closest to a control environment with meeting the strict requirements of single-blind controls.
3. Part of the research findings presented project status analysis results to senior management as a comparison to the project status results from the conventional methods employed. It was not desired that the people conducting the projects participate in any way in the parallel research.

Population

The population for this study came from the IT organization of a large aerospace company performing software development. The IT organization had been appraised by an SEI-approved appraiser at Capability Maturity Model Integration Level 3. A project was sampled within one of the IT software development organizations. The project dealt with web applications interfacing with production databases. The project that was the subject of the research was a company wide application. It was an automated tracking system used throughout the enterprise to request acquire, change, relocate, and dispose services for Information Technology (IT) assets including computing hardware and software products. It had visibility to the top executives, had to be operational 24 hours a day, seven days a week. It was an intricate part of the IT internal operations and was critical to the successful operation of the rest of the entire organization.

The IT supervisor, project manager and part of her staff were located in the state of Washington. Other parts of the organization were located virtually throughout the U. S. as well as Indian subcontractors. The populations sampled were requirements

www.manaraa.com

documents from the project and the defects that were discovered in them. The samples were part of a large major upgrade to an existing product. Expected quality data was provided by the application supervisor and project manager.

The research question about project management focused on determining requirements for project status quality during project execution prior to the test phase, which would produce reliable predictions of future impacts on the cost and schedule commitments of the project. The research question for quality focused on examining if there was a software quality measurement technique currently available that could be adapted as an in-process (before testing) project quality measure. The research question for communication focused on determining if the process project impact based on product quality required a new and as yet, undocumented, communication mechanism or method not currently part of the structure of established project management reporting practices.

Reliability

Reliability in this study had two aspects. The first aspect was the process that the project followed in producing the software deliverables that make up the components of the software development process in the Macroscope methodology. The project sampled was operating in an industry standard reliability framework: Capability Maturity Model Integration. As each level of Capability Maturity Model Integration maturity was achieved, the reliability of the processes to produce a product increases and the products themselves were more consistent within the project and from one project to the next within the same organization. The project, with all 16 documents, was appraised at Capability Maturity Model Integration level 3 – Defined. This level has all projects

www.manaraa.com

operating at a common organizational level set of processes including documentation, software development processes and metrics data collection. The documents and processes for developing them should have had a consistent reliability in documentation data gathered.

The second aspect of reliability in this study was the reliability of the inspection results when the documents created using the Macroscope methodology in a Capability Maturity Model Integration Level 3 process environment were examined. The subject matter experts that reviewed the documentation were software architects in the organization that the project was in but not directly related to the project. This was to attempt to ensure the reliability of their inspections in being objective and unbiased.

Validity

The project sampled was representative of types of projects in the company IT department and IT departments in general. It was a large project covering more than a year's time span that was a major upgrade to an existing system. The documents sampled were completed documents of a completed project with multiple releases of the application in production. Customers were using the application and customer identified defects were being tracked by the enterprise help desk. The documentation was stored in production level configuration management tools.

The methodology employed for determining the validity of defects detected was based on industry norms of generic definitions of requirements defects. The rules applied to the unique total defects detected and defects remaining calculations were based on

www.manaraa.com

industry experience from recognized experts in the field of software inspections as applied both to commercial and government projects.

Data Collection

For this study, the data that was used was existing archival data. Gathering this research data did not involve answering questionnaires by human subjects. The data used derived from already completed production documentation and was collected completely independently of any activity within the projects themselves. The documents were made available for analysis, and any subsequent findings had no impact on the ongoing projects. The data collected fell into three categories:

1. Project Data: The specific data concerning the project that fed into the parametric equations for performance calculations like schedule(s), documentation, resources, etc.
2. Project Management Estimates: Project related personnel were requested to determine estimates on the quality of their deliverables, something not done in standard project management.
3. Major Defects found: The number of defects in the project deliverables that was determined by subject matter experts using the Agile Inspection Method.

Data Analysis

The analysis of the data collected following the process outlined for analysis of the documents' quality and defect density was done through the use of standard data

www.manaraa.com

analysis tools such as statistical process control, simulations, input sampling and parametric analysis. The three research questions previously stated are as follows:

1. What are the requirements for software quality during project execution, prior to the test phase that will produce reliable predictions of future impacts on the cost and schedule commitments of the project?
2. What is a software quality measurement technique currently available that can be adapted as an in-process (before testing) project quality measure?
3. What is the In-Process (before testing) project impact based on product quality that requires a communication mechanism or method not currently part of the structure of established project management reporting practices.

Testing of research question 1 was accomplished by developing a parametric simulation of the project that responds to defects detected and comparing the results of the projected QPI method impacts with actual project performance. Testing of research question 2 was accomplished by evaluating the effectiveness of the agile software inspection technique selected from the literature review analysis to produce a measurement of software project quality. Testing of research question 3 was accomplished using the QPI method of project communication that incorporated the additional elements identified in the parametric simulation of the project that do not currently exist in project management communication constructs to communicate the project simulation results to senior management.

www.manaraa.com

Sampling

Sample method. The sampling approach chosen for this research was purposive sampling. Within purposive sampling there were two major categories: judgmental and quota (Cooper and Schindler, 2003). Quota sampling was used to improve representativeness. Judgmental sampling was used to ensure that the samples conformed to specific criteria. In this study the judgmental sample method was used. The specific criteria were that the sampled documents were all requirements documents; they were of Macroscope design, and they occurred in a project that used the Macroscope methodology. The project was also Capability Maturity Model Integration Level 3 appraised. The documents selected were what the project had available and were used in the execution of their project. No attempt was made to alter the samples to fit the prescribed methodology. Instead both the documents and their state of currency were used exactly as they had been created and subsequently modified (or not) by the project.

The other judgmental aspect of the sample was the selection of the subject matter experts to review the requirements documents. The subject matter experts were selected based on the following criteria:

1. They had no connection either technically or organizationally with the projects.
2. They were well respected by their peers and management (who had to approve their involvement in this study).
3. They had a broad bandwidth of knowledge and experience in the IT environment, both in applications development and in providing services.

www.manaraa.com

Sample size. In this research, the sample size was the number of requirements documents which were sampled individually and independently. For the project, the total number of documents examined was 16. The total number of unique content pages for this combined set of requirements documents was 378. With the word count per page ranging around 120 per page (conservative), the sampling size opportunity for unique defects was in the order of 45,360.

Sample size justification. As previously itemized, the various sample size groupings were, in increasing order, 1, 16, 178 and 45,360. When requirements were being reviewed for defects, each word that failed to meet the reviewing criteria for a valid requirement was considered a defect. A defect was identified under the checklist conditions shown in Appendix I which was the actual checklist the reviewers used to determine major defects.

When defects were counted, if one sentence contained eight different words or phrases that were defective, then the number of major defects for that sentence was eight, not one. The opportunity then for defects depended on the word total with only one defect being allowed per word. If a word and its context were such that two possible major defects could be realized, the Agile inspection counting rules only allowed one defect to be counted.

Example: For the following requirements statement, the screen should respond to the input quickly, there were at least two major defects:

www.manaraa.com

1. The word should is not a requirement. The application could be coded without any attempt made to meet this requirement and it would still be legally correct. The statement needs to say that the screen "shall" respond…
2. The word quickly is not testable. There is no quantification to verify that performance is met or not. If the statement was modified add a two second response to an input then it would be valid.

Thus, a reviewer would log two defects even though there is only one requirements statement.

Sample reliability. The project that was sampled was part of an organization that was appraised at a Capability Maturity Model Integration Level 3. One of the Key Process Areas at Capability Maturity Model Integration Level 2 is Requirements Management. At Capability Maturity Model Integration Level 3 there is another requirements oriented Key Process Area titled Requirements Development. The requirements documents sampled in this study were created in a Capability Maturity Model Integration Level 3 process. The stability and reliability of the requirements were clearly established by the project's status in Capability Maturity Model Integration.

This sample reliability and process stability was necessary to allow the correlation of the parametric simulation to the actual project results. If the quality of the documents or the process of producing them varied from document to document within a project or across projects, then the ability to make a consistent correlation of requirements defects on the project process would have been diminished.

www.manaraa.com

Sample validity

Sample validity has two components: accuracy and precision (Cooper & Schindler, 2003). The sample was stable with no systematic variation. The requirements documents conformed without deviation to the Macroscope methodology that was implemented at Capability Maturity Model Integration Level 3. The precision of the estimate was determined by the congruity of the documents developed to those that the methodology prescribes.

Precision is expressed in terms of validity and reliability (Hopkins, 2000). Validity is determined by how well it measures what it's supposed to. In this study, all defects were valid defects. Reliability deals with the ability to repeat the results. The methodology involved was designed to specifically negate the impact of different subjective appraisals thus reinforcing the reliability of the methodology.

EVM Methodology

The EVM Methodology is a well established methodology with two major components: cost and schedule. However, all projects contain a third project component of technical performance/quality and a fourth of risk. In the Earned Value Management Methodology currently, there is neither a direct quality or risk methodology component. This study incorporated quality into the EVM methodology. The attributes of EVM components are composed of predicted values and actual values. This study analyzed the various approaches within the software profession for measuring quality to determine if one or more of the quality measure could be included in the EVM methodology to create a quality component to EVM.

www.manaraa.com

Additional Factor Inputs, Equations and Quality Component of EVM

There were additional factor inputs with resulting equations and quality components. These factors were all identified and itemized in detail in Appendix C. Factors included content pages, defects identified, expected defect density, and impact multipliers.

Summary

This chapter provided an overview of the research approach used including the research methodology. The research approach is quantitative specifically using a quantitative, post-facto, exploratory, research design. Statistical tools and techniques were used to examine the project data after it was completed. It is non-experimental since all activity on the project being researched has already been completed. The QPI method was developed from the literature search; it integrates product quality with EVM to produce project quality.

In chapter 4, the QPI method that addressed research question 1 is developed based on the literature research performed in chapter 2. Agile inspections that addressed research question 2 were initiated on completed project documents with the simulation results documented. Project metrics were part of the analysis and the results were captured. The data gathered from the project was analyzed using the QPI method and conclusions were developed including presentation of results that addressed research question 3.

www.manaraa.com

Chapter 4: Results

As previously stated in chapter 3, software projects continue to fail most often due to project management quality issues. Since the NATO report was published in 1968, software quality has been an elusive goal for many IT, military, and government organizations. Software projects fail most often due to project management quality issues (Jones, 2004)—which include not managing changing requirements, not allotting time for detailed requirements analysis, and not allotting sufficient time for verification tasks (including inspections, testing, and defect repairs).

This chapter reports the results of the data analysis on the use of the new project quality methodology based on the literature review and the study research. The results of data analysis consist of (a) the analysis of the methodologies found in the literature search and the resulting new methodology that addresses the research questions, (b) the results of running a simulation using the new methodology with actual project data from a completed project, and (c) comparing simulated performance with actual project performance.

The findings document a significant improvement in early project management detection of poor project status. It provides answers to the following research questions:

1. What are the requirements for software quality during project execution, prior to the test phase that will produce reliable predictions of future affects on the cost and schedule commitments of the project?
2. What is a software quality measurement technique currently available that can be adapted as an in-process (before testing) project quality measure?

www.manaraa.com

3. What is the in-process (before testing) project affect based on product quality that requires a communication mechanism or method not currently part of the structure of established project management reporting practices?

Results of Data Analysis

On February 22, 2005 the project researched in this study completed releasing its requirements documents. The status reported by the project was that it was on schedule and within budget according to the parameters set by the organization. The application was scheduled to be delivered in 3 months, at the end of May. At this point in the project, if the QPI method had been available to executive management, it would have provided them with the following information:

1. The QPI for the project was 0.29. In EVM, no project can recover from a performance index that low. To continue, the project must be re-baselined.
2. A schedule slide of 3.6 months was projected.
3. An additional resource impact of 14,764 hours was projected.

The executive team reviewing the project status would have received a green status from the standard project management disciplines, while the QPI method projected a slide to September. In the rest of chapter 4, the details of how these results were obtained are explained along with detailed findings.

Analysis of Literature Search

The first set of results comes from the literature search and the EPI analysis. Out of the EPI analysis came the combining of elements developed separately to produce something that had not existed previously (based on available literature). The creation of

www.manaraa.com

the Quality Performance Index (QPI) and the use of Agile inspections to achieve the data used in the QPI are the QPI method and were derived from the literature search.

EPI Analysis

According to the EPI analysis in Figure 6, the literature research areas that had the most affinity for the study research questions were DPI, EVM, PEVM, and inspections. DPI is a proprietary methodology and therefore not much further analysis is achievable. PEVM is a special case of EVM and the two will therefore be considered as the same with EVM being the methodology analyzed.

EVM analysis. A project has four major components (NASA, 2007): cost, schedule, product technical/performance/quality, and risk. EVM's coverage of those project components includes cost and schedule. While Earned Value has the ingredient of work accomplished, the measuring of the work accomplished is not a part of the EVM methodology. This can result in work been credited as accomplished when it will yield inferior performance. This inferior performance is often masked until testing occurs. This can cause the project to go from green to red as the project enters the test phase and the more advanced test cases start to fail.

The EVM performance indexes are constructed similarly such that if the project is performing better than expected by accomplishing all its tasks and using less budget (i.e., being underrun), then the CPI is >1.0; and if the project is performing better than expected by completing its task ahead of schedule then the SPI is >1.0.

Quality Performance Index (QPI). If the EVM methodology were to have a product technical/performance/quality performance index consistent with the other two

www.manaraa.com

performance indices it would be in the form where performance better than expected would be > 1.0 and adverse performance would be < 1.0. Software inspections produce an indication of quality on software development products prior to the software entering the test phase. If an element of expected quality in the software development products is introduced, then there would be the minimum number of elements necessary to produce a quality performance index within the EVM methodology.

When measuring quality in the amount of defects in a product, the fewer the number of defects, the better the quality. To produce a performance index where better than expected produced a result of > 1.0, the QPI equation would be as follows:

QPI = Expected Quality/Actual Quality (**Equation 1**)

Inspections for defects. There are two aspects to inspections. The first is what type of inspections and the second is where in the software development process the inspections occur. The problem identified in this study is a lack of accurate status on the project status before the product enters the project testing phase. Earned Value Management was previously cited in this study as being most vulnerable to inaccurate status when the project was in its early phases. Therefore, the literature suggests that the best documents to start inspecting would be the requirements documents.

As previously identified in this study, there are multiple types of inspections. The formal Fagan inspection is the most rigorous but also requires the most resources and time, which is something that most projects which are in trouble usually do not have. In addition, the objective of the Fagan inspection is to fix the product whereas what this study is interested in are ways to improve the reporting of project status. The Agile

www.manaraa.com

inspection process gives defect counts but uses very few resources and is not as rigorous in process as Fagan inspections.

In summary, we concluded that, since we need correct status as early as possible because the early phases is when EVM is least accurate, we would inspect the project application requirements documents. Further, since resources needed to support inspections are usually tight, the study decided to use agile inspections. Along with low resource usage, agile inspections produce the defect count and the defect location if the reviewer annotates the document. Agile inspections have the disadvantage of not producing the specific defect type or improving the product under inspection.

EPI Results - A QPI Method. The results of the research analysis from the literature search led to the following attributes of a QPI integrated into the EVM methodology:

1. QPI value computed by dividing Expected Defects by Actual Defects.
2. Agile inspection process used for determining actual defects.
3. Project requirements documents inspected.
4. Effort leverage between finding and fixing defects in requirements vs. finding and fixing defects in test applied to unexpected defects to obtain schedule impact.

Items 1 and 2 are the elements used in the implementation of the QPI method, which is a method for determining project status that did not exist prior to this study. Items 3 and 4 give the user of the QPI method direction on where is the place to incorporate it in the

www.manaraa.com

project that provides the most prediction leverage with the minimal use of resources at the earliest stage in the project where it is possible to use the QPI method.

Simulation Parametric Data

The second set of results was found by creating a parametric simulation of the QPI method's prediction of the project performance. The data produced by the simulation was compared to the actual project performance. The simulation was created parametrically so that an organization attempting to use the QPI method could adjust their variables to meet the unique conditions existing in their own projects.

Independent Variables (IV)

The IVs in this study were the parametric variables that are inputs to the QPI method simulation. The following IVs were of primary interest in this study:

CP = Content pages of each document reviewed. Content was defined as that part of the document that adds value to the product.

D = Major Defects found by a single reviewer. Requirements documentation is the only class of documentation that was examined; architecture, design, and code were purposely excluded. An example of a customer requirement that had at least one major defect is the application will respect the users time. This requirement was un-testable and thus a major defect. There was no test case that could determine respect. Instead the statement needed to have a quantified aspect such as the application shall display results within two seconds of a user's request. Thus, a test case

www.manaraa.com

could be developed that had a pass / fail criterion of two seconds response time for the application.

ED = Expected defects per content page. This was the value of defects per content page of documentation that the project managers expect the project developers to produce during the software development process. This was a normalized predictive quality indicator. In the methodology, this value was provided by the project manager at the start of the software development project.

H = Hours per day that a person spends on a project.

P = Number of people on the software development project.

TCP =Total number of content pages reviewed. This is a cumulative number of pages reviewed by the reviewers as more documents are reviewed (see Equation C-2 in Appendix C).

Impact Multipliers (ME1, 2005):

There were well established leverage factors for defects found in the various phases of software development compared to the cost in time and resources that it would take to fix the same defect when it was detected in the test phase. For pre-coding defects, the leverage can range typically from 100-to-1 to 10-to-1 (Rothman, 2000). If there were requirements defects that were caught and corrected in the specification phase of a project, then the resources required to fix them there was typically 10 times instead of 100 times (ME1, 2005).

www.manaraa.com

PI = 1000. The Impact multiplier of defects detected, after the product was released to production that were generated in the Requirements phase.

TI = 100. The Impact multiplier of defects that were detected in the Test phase but were generated in the Requirements phase.

DI = 10. The Impact multiplier of defects that were detected in the Design Specification phase but were generated in the Requirements phase.

Dependent Variables (DV)

The DVs of interest were related to quality. These were newly identified DVs based on the methodology developed from the literature research that identifies Project Management information that was produced by a Quality Component of EVM. There were two areas of project management information that could be created by an earned value management process that contained a quality component.

RD = Total Defects found by the reviewers. The rule of thumb for estimation was that the total unique defects found was twice the number that was that maximum of the largest number of defects found by each reviewer (see Fig. C-1 in Appendix C).

TD = Total Defects in the document. The rule of thumb for estimation was that the total number of defects in the document was three times the number detected (see Equation C-3 in Appendix C).

TDD = Summation of Total Defects in each document. This was a cumulative number of defects detected by the reviewers as more documents were reviewed (see Equation C-4 in Appendix C).

www.manaraa.com

DD = Defect Density - Detected Defects per Content Page. This was the total number of defects detected divided by the total number of content pages (see Equation C-6 in Appendix C). This was the actual value of the quality index as compared to ED.

QPI = Quality Performance Index (QPI). Earned Value Management had two performance indexes in its current methodology. Schedule Performance Index (SPI) was described in Table A-5 (see Appendix A). It had both a predicted schedule component (Planned Value) and an actual schedule component (Earned Value). Cost Performance Index (CPI) was described in Table A-5 (see Appendix A). It had both a predicted cost component (Earned Value) and an actual cost component (Actual Cost). To develop a QPI, therefore, there had to be a predictive component and an actual component. QPI was the ratio of the expected quality (ED) to the detected quality (DD) as described by Equation C-8 in Appendix C.

The QPI status to Executive management could be represented in the same scale as the status of the Cost and Schedule performance indicators:

Green = >90%

Yellow= 80% - 90%

Red = <80%

The second piece of project information was potential cost and schedule impact in terms of projected hours of resources and days of schedule. Management had identified an expected defect density, which when multiplied by the number of content pages or lines

www.manaraa.com

of code generated in a development phase produced the number of anticipated defects in that phase. When the number of real defects found in that phase was subtracted from the number of anticipated defects, this left the number of unanticipated defects. This was the number of defects per content page that were not expected in the planning of the project and was shown in Equation.

UDD = Unanticipated Defects per Content Page. The inspected defects per content page minus the expected defects per content page (see Equation C-7 in Appendix C).

Status Reporting – Project vs. QPI Method

A comparison of the classical project EVM status reporting to the enhanced QPI method status reporting is shown in. If the QPI method had been in place on the project, as early as February 22, 2005 the QPI predictions could have been available to the executive management. The QPI predictions present a picture of impending disaster to the executives since the project was scheduled to be completed in May.

Table 9.

Contrast of EVM and QPI method

Status Elements	EVM Project Reporting	QI Methodology Predictions
Cost	Within Budget	Additional Budget Required • 15,093 hours • $1,660,230
Schedule	On Schedule	Project will slide 3.6 months
Quality	None Available – Project assumes that following development methodology will produce good quality product	QPI = 0.29 • 1.0 is desired • Project is not recoverable when an EVM PI is that low.

www.manaraa.com

Research Tools and Data Gathering

Project Document Deliverable Selection Criteria

The documents selected for inspection were the requirements documents. An example of a completed agile inspection Requirements Inspection Checklist was found in Appendix D. When the project was completed, it used an earlier version of Macroscope called P+. The entire suite of P+ documents was mapped to their equivalent documents in the current version of Macroscope being used (Macroscope Version 4.5).

In the methodology description, there were a set of core deliverables and an additional set of optional deliverables. The project produced 57 documentation deliverables out of a possible 88 of the expanded methodology for a deliverable coverage of 65% of the total that could be produced. If only the basic suite of deliverables was considered, that reduced the number that could be produced by 21. In that case the project produced 48 documentation deliverables out of a possible 67 of the expanded methodology for a deliverable coverage of 72% of the total that could be produced.

Macroscope Requirements Construct

The study was concerned with those deliverables that were considered requirements documents. As seen in Figure B-2 in Appendix B, the project completed the Owner Requirements and Developer Requirements. The documents that had the light red overlay on them were the ones that the project created. The project also appeared to have not identified any specific unique user requirements as standalone deliverables. The project manager stated that the project captured these requirements in design documents

www.manaraa.com

that were created farther downstream in the process. There were 16 project requirements documents, listed in Table A-7 in Appendix A that were inspected for the study.

Data Collection Process

Three architects volunteered to perform the agile inspections of the requirements documents. Each reviewer received a copy of each document along with an inspection sheet. The only thing required of the reviewer was to fill out the times and defect numbers at the bottom and return them to the researcher.

Reviewer C was a highly regarded technical analyst and was initially quite interested in the research. Reviewer A was the most skeptical at the start. Paradoxically, Reviewer C never submitted an inspection result; his participation was voluntary and he could not be coaxed into putting in the additional effort when it was finally needed. Reviewer A ended up being the only Reviewer to inspect all the documents and submit reports. Reviewer B reviewed five documents.

In the technique of agile inspections, the number of defects in a document was twice the maximum number found by a single reviewer. While it would have been nice to have more than one reviewer's numbers, this turns out to be very beneficial for the study since the proposed methodology was designed to add information on project status in real world conditions and having a lack of resources is definitely a real world condition. The data from the agile inspections was listed in Table A-8 in appendix A.

There were two reviewers for five documents and one for the rest; if more reviewers had participated, then possibly more defects could have been discovered. This would have made the already negative predictions of project status even more negative. It

www.manaraa.com

would have changed the result in magnitude but not direction. We could logically conclude the following:

1. The findings based on the reviewers' inspections were suggestive and more reviewers could only make the projection of project performance more negative.
2. If the reviewer was the most technically excellent of those who would be chosen if multiple reviewers were chosen, all other things being equal, then his numbers would generally be the highest and therefore give an accurate result.
3. If the reviewer is not the most technically excellent of those who would be chosen if multiple reviewers were chosen, all other things being equal, then his numbers would generally be lower than the highest.
4. If reviewer C were used, then his defect counts would logically be a minimum going on the assumption that if reviewer A also participated, he would have found more defects.
5. If the management proceeded with the analysis using reviewer C and it showed a negative disparity to the status presented by the project, we could assume that the disparity was real and probably greater than indicated.
6. The QPI method was designed to allow management to assess the project status using the management-by-exception approach. In other words, if the project showed Green and the QPI showed Red, then management would delve further into the project status. If the project showed Green and the QPI

www.manaraa.com

showed Green, then management would not delve further into the project status. Any management action based on Yellow status would depend on how the particular program was using the QPI method.

7. While a Green status using only reviewer C may have indicated the project status may be in worse shape than the defect data indicates, the fact that management was using only one reviewer and he was not their best demonstrates the amount of weight they wanted to place on the QPI method.
8. Therefore we concluded that the use of one reviewer is logically technically valid to present a negative impact and achieved the minimalist management goals of the organization using the QPI method.

The defects per content page data points were analyzed for stability using the statistical process control C Chart equations for defects. The results were shown in Figure 7. The first observation that could be made was that the project's process for creating requirements defects was stable. While there was a peak of 4.0 defects/content page for the P130 deliverable, that was still within the Upper Control Limit (UCL) of 4.6 defects/content page. In this case, the Lower Control Limit (LCL) was 0.0. That was not always the case, but for a C Chart, the LCL can never go less than 0.0 since there can never be negative defects.

www.manaraa.com

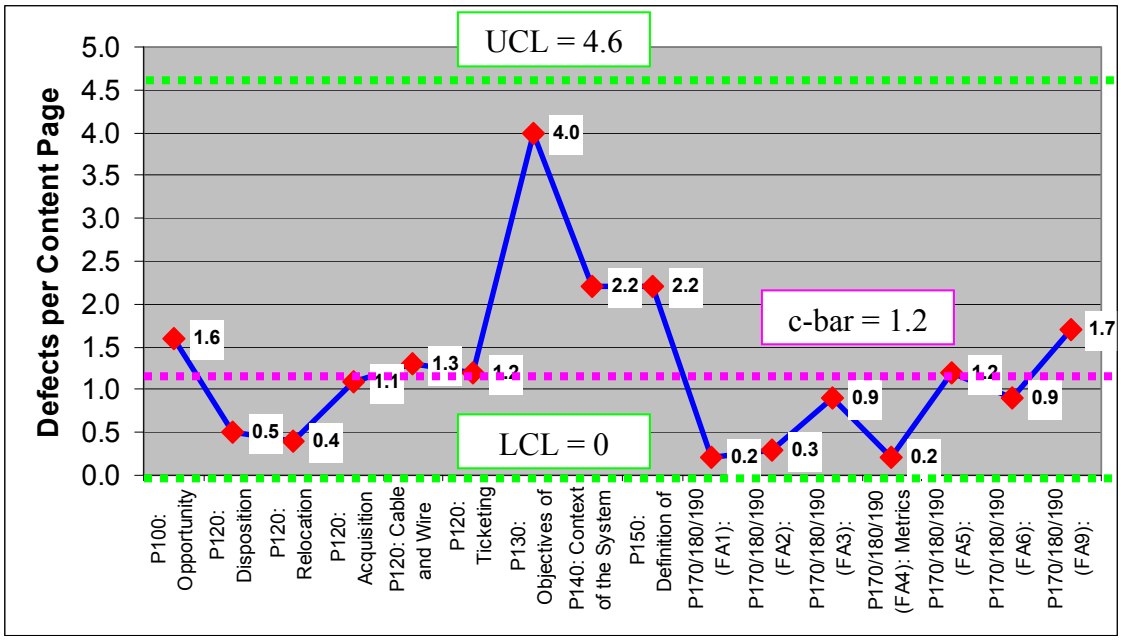

Figure 7. Control chart - agile inspections defects per content page (DD)

The next observation was that the c-bar for all the requirements documents was 1.2 defects per content page. Defects per content page was also described elsewhere in the software development community as Defect Density (DD). DD could be applied to documents or code (Defects/Thousand Source Lines of Code). As part of this study that goes beyond traditional use of DD, the project manager and his supervisor were asked the following questions:

1. In your estimating of project schedules, what is the DD that you expect in your documentation?
2. What do you think is your actual DD for this project?
3. What goal DD would you like to project to achieve at some point?

www.manaraa.com

4. Assuming that we are inspecting deliverables that have already been approved to better understand project impact and not to fix the deliverables, what would be the DD threshold that if crossed by a document would be sufficient quality grounds for disapproving the document and not showing it as complete on your schedule?

The project's response to these questions was shown in Table 10. There was already a preliminary indication of tension in the project in that the difference between the Expected DD and the Anticipated DD was 0.5 major defects per page. That implied that the project manager and supervisor already knew that the quality of their development deliverables was low than they expected.

Table 10.

Project Defect Density Expectations

Project role	Expected DD	Anticipated Actual DD	Goal DD	Reject DD
Project Manager	0.2	1.0	0.1	0.5
Supervisor	0.3	0.5	0.1	1.0
Study Average DD	0.25	0.75	0.1	0.75

This table itself led to some interesting further observations. There was a gap between what the project DD expects to be on a nominal project and what the project leaders thought the actual DD was on this project. The ratio was 3 to 1 worse for this project. The explanation in this case was that the Expected DD was what the project would like to see but the Actual DD was what the project thought was the real case. Of importance to the study in the use of the methodology was that the Expected DD was

www.manaraa.com

what was used by the project to estimate the amount of rework needed to be inserted into the project schedule.

Another observation concerns the reject DD threshold. There were 16 documents being reviewed and it appeared that most of them would have been rejected had this process been in place. The last column in Table A-7 in Appendix A, indicated whether the deliverable would have been allowed to proceed (YES) or by what margin of DD if failed. There were five deliverables that would have been allowed to proceed on in the software development process and 11 deliverables that would have been required to be redone. This was a red flag that there were major quality problems in the requirements. These were deliverables that had successfully been through the process of a Capability Maturity Model Integration Level 3 project which included peer reviews prior to approval.

The c-bar of the set of deliverables was 1.2 DD. If all the defects were compared to all the content pages as if they were one deliverable, then the DD was 330 defects divided by 384 content pages resulting in a DD of 0.86. On both measures, c-bar and cumulative DD, the project requirements deliverables DD as a whole was greater than the DD reject threshold of 0.75 DD.

The next data point to collect from the Agile Inspection data was the amount of time that it took to gather the information. The total amount of minutes used by all reviewers to inspect the deliverables and record the results was 540 minutes or 9.0 hours. This was one of the significant appeals of this method, especially for a resource limited project. For an investment of only 9.0 hours, which was insignificant in almost any

www.manaraa.com

project, but especially for this project which ultimately used 39,383 hours, the project manager and executive management would have information about the anticipated impact on the project that they didn't have previously. This information was available early in the project life cycle just after the requirements were completed. If the requirements gathering process was having problems, we could anticipate that there would be further challenges farther down the waterfall methodology. It should be noted that there was nothing in this methodology that required the reviewers to wait for all requirements documents to be completed. The inspections could have been completed as the requirements documents were completed and the affect on the project, either positive or negative, could be cumulatively assessed.

The Parametric Equation for Projected Project Impact

The structure of the simulation of the QPI method was parametrically defined for two major reasons. It allowed the step by step methodical construction of the methodology impact on the project to be generated, and there were necessary assumptions made in some of the independent variables in the construction of the methodology. These assumptions could be challenged by other project or management personnel. By allowing parametric construction, these assumptions could be changed to fit the culture of the project being examined without doing damage to the simulation and at the same time satisfying those participating in the use of the QPI method in their organization. Appendix C contained a list of all the variables alphabetically. Figure B-5 and Figure B-6 in Appendix B showed the inputs and outputs of the QPI method using a spreadsheet to capture the parametric equations within the QPI method. The IVs were in

www.manaraa.com

blue text, the DVs were in black text; the red text was the final process output for the method. The research process flow is shown conceptually in Figure 8 and in detail in FigureB-4 in Appendix B. The requirements documents in the project were all inspected for defects. The totals of the inspections, the pages inspected and the project parametric values were input into the QPI simulation which produced the following projected project impacts:

1. QPI = 0.29
2. Schedule Impact = 3.6 months
3. Resource Impact = 14,764 hours

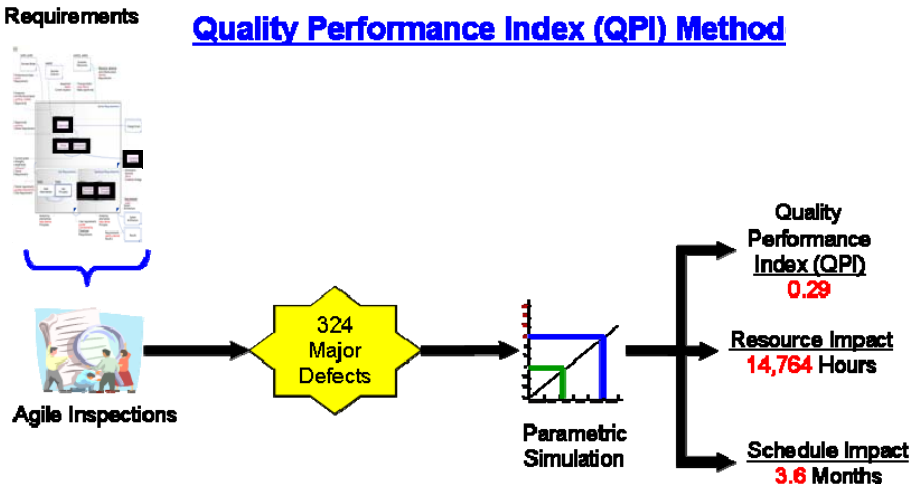

Figure 8. Conceptual research process flow.

Defects Variables - Quality

In the QPI method simulation the total defects (TRD) for the project requirements documents inspected were 330 defects. The total of the content pages (TCP) for the

www.manaraa.com

project requirements documents inspected was 384 content pages. The defect density for this project was 0.86 defects per content page. The expected defects (ED) were 0.25 defects per content page from Table 10. The unanticipated major defects detected per content page were 0.61 defects per content page.

For all the documents in this project the total number of defects was 990 major defects (TDD). The total unanticipated major defects, variable UD, were 702 major defects. In this project, some of the defects that had been found in the inspections were probably also found and fixed by the project prior to going into the test phase. The consensus of the project was that the percent of defects found prior to test was about 50%. For the purpose of this study the number of requirements defects found and fixed prior to the software getting to test was 351 defects. Using the criteria of estimated divided by actual a quality performance index (QPI) was created. In this project the QPI at the end of the requirements phase was ED (0.25) divided by DD (0.86) which resulted in a QPI of 0.29.

Effort Variables - Schedule

Design/code phase effort. The effort to find and fix requirements defects in the design/code phase was the dependant variable DE which was 3,510 hours. This translated into the dependent variable of additional months of schedule impact AME which was 0.9 months of project schedule impact due to unanticipated defects found during the design/code phase of the project.

Test phase effort. The independent variable of percent of defects found in test PDFIT was typically 33%. The dependant variable of the effort needed to find and fix all the

www.manaraa.com

requirements defects found in the test phase TE was 11,583 hours which converts to the dependant variable AMT which was 2.8 months of project schedule impact due to unanticipated defects found during the design/code phase of the project. The total projected schedule impact on the project based on the unanticipated requirements defects found in both the design/code and test phases was cumulative. The dependant variable of total schedule impact TAM was 3.7 months.

Effort Variables - Cost

The effort in hours could also be converted into cost. The company standard personnel rate was the independent variable RAP which was $110 per hour. The projected cost impact to the project was a dependant variable TC which was $1,660,230.00. If no additional budget was added and the project was re-baselined, the CSI would be less than 1.0 at the completion of the project.

Validation of the Agile Inspection Methodology

National Institute of Standards and Technology (NIST) Defects Model

The National Institute of Standards and Technology had produced a defects model for software development based on accumulated data from many software projects (NIST, 2002) as cited by Borland (Borland, 2006). The percentages assigned to each phase by the model were captured in the second column of Table 11.

Project Defects Data

The next step was to compare this model to the available project data to infer a DD independent of the Agile Inspection data. The project under study did capture both pre-release defects and post-release defects. For this project, the number of pre-release

www.manaraa.com

defects that were logged was 1,447. The number of post-release defects that were logged was 151. The relationship of the project to the NIST model is shown in the 3rd column of Table 11.

NIST Defects Model Predictions of Requirements Defects

There were two data points for calculating the number of Requirements/Design defects using the NIST model: prerelease defects and post-release defects. Due to the way that the defects measures were logged into the measurement and metrics tool in this Capability Maturity Model Integration Level 3 group, the number of post-release defects did not represent the total number of post release defects associated with the particular release under study. The pre-release defects for this release were, however, all associated with the release. Therefore the number of pre-release defects more accurately reflected the true state of the software development effort for the release when using the NIST defects model. The pre-release defect count was used, resulting in the total number of defects for the project of 2,631 (1447/0.55). The number of Requirements defects would then be 395 (2631*0.15). This calculation was shown in the 4th column in Table 11.

NIST Defects Model Compared to the Agile Inspection Methodology

The number of defects found in the project requirements documents using the Agile inspection methodology was 324. This number was less than the number that the NIST model produced but the scope of the inspections was against only requirements documents. The application of the NIST model to the project and the comparison of the Agile Inspection results were captured in the 5th column of Table 11.

www.manaraa.com

The comparisons of the two sources of data were mutually reinforcing. By having actual defect data from the software and comparing it with the inspection defect data obtained, we were able to conclude that each set of data was within the parameters of the NIST model. The NIST model was suggestible that the Agile Inspection methodology for determining number of major defects was correct and a useful methodology.

Table 11.

NIST Defect Model and Agile Inspections Methodology

NIST Model	% Defects per Phase	Defects Logged	NIST Defects Predicted	Inspection Defects
Requirements/Design	15%		395	324
Code/Unit Test	20%			
Test Total (incl Beta)	55%	1447		
Post-Release	10%	151		
Model Total			2631	

QPI method Predictions verses Actual Project Schedule Performance

Actual Project Schedule

The project under study was started on October 1, 2004. At the time it was started, the scheduled completion date was the end of May 2005. The project developed its requirements documents through the 4th quarter of 2004 and early into 2005. According to the project manager, the requirements were complete on the 22nd of February. If the management were using the QPI method, they would have known that

www.manaraa.com

the QPI method was predicting a 3.6 month slide in the schedule. The initial schedule with a QPI method analysis superimposed at the correct date was shown in Figure 9.

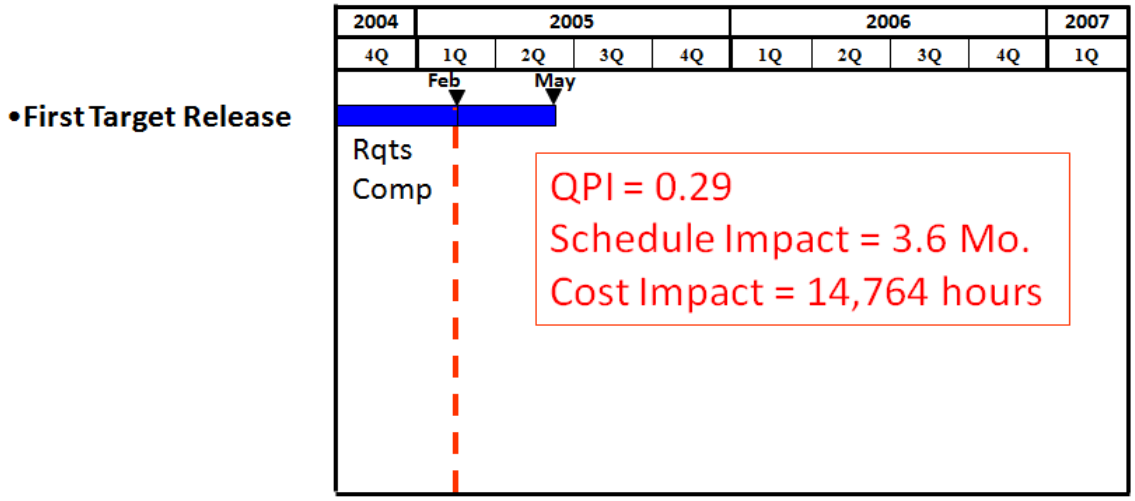

Figure 9. Project initial schedule and QPI predictions

Project Actual Schedule Performance

The project did go through some major schedule perturbations. One month before the project was originally scheduled to be completed, a 4-month slide was added to the delivery data. During that four month extension, the project went through a complete re-baseline that now had it completing in June 2006. Having finally gone through a thorough analysis of what it would take to deliver to the customer what they wanted, the customer declared that June 2006 was way too late. They needed it by February 2006. With the customer's acknowledgement that moving the delivery date up by that much would produce a product that had not be thoroughly checked out, the project agreed to the new proposed date.

As the project was getting ready to deliver the software, the customer discovered that they had left of a critical requirement that made the software unusable as coded. The

www.manaraa.com

incorporation of this additional requirement and the subsequent retesting resulted in the software being delivered at the end of July 2006. Then the project spent the next five months fixing major defects that were still in the delivered software. This project history was captured in Figure 10.

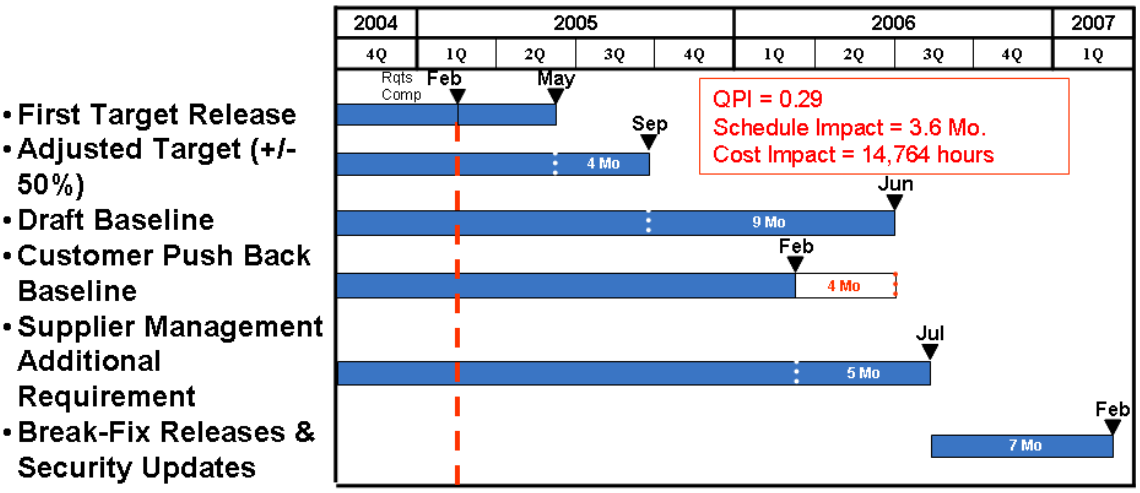

Figure 10. Actual schedule history for the project under study

Figure 10 shows the original schedule with QPI predictions as well as the additional history of the project that included direct customer intervention and a major requirements change at the end of the project.

Research Questions Findings

Research Question 1

The first research question was, What are the requirements for software quality during project execution, prior to the test phase that will produce reliable predictions of future impacts on the cost and schedule commitments of the project? The findings of this study indicated that there were software quality attributes that can be used to produce reliable predictions of potential future project impact to both schedule and resources. The

www.manaraa.com

emphasis on this study was to determine if there was a quality aspect to the status of the project as opposed to the status of the product that the project is producing. The requirements for a prediction include expected values being compared to actual values. The requirements for software quality were defects with both expected and actual values. The expected value of quality was necessary so that status could be assessed against it. This status of the product in the project process contains information that was unavailable by examining cost or schedule information.

Research question 1 findings.

1. The QPI method was unique in definition and implementation.
2. The QPI method produced reliable predictions of future impacts of project cost and schedule.
3. The Expected Defect Density of software development products were a requirement for the QPI method to succeed.
4. The Actual Defect Density of software development products were a requirement for the QPI method to succeed.
5. The prediction capability of the QPI method derived from the difference between the expected defect density and the actual defect density.
6. The use of a single inspector produced industry usable and academically suggestible results.
7. Management commitment was needed to ensure availability of inspection resources.

www.manaraa.com

Research Question 2

The second research question was what is a software quality measurement technique currently available that can be adapted as an in-process (before testing) project quality measure? The findings of this study indicated that inspections of software development products (in this study requirements documents were analyzed) was a process that could be adapted from using the inspections information to adjust the quality of the product to using the inspections information to adjust the quality of the project status. This study showed that the agile inspections technique was applicable to providing valid inspections information. The powerful aspect of agile inspections was that their resource requirements are minimal compared to the cost of formal (Fagan or PEP) inspections. In this study, for an investment of 540 minutes (9.0 hours out of 39,383 hours in the project), a predicted 3.6-month schedule impact was predicted when the project status was being shown as on schedule.

Research question 2 findings.

1. The comparison of the NIST model with the agile inspection results produced a suggestible correlation with respect to the integrity of the agile inspection process.
2. Inspections and Agile Inspections in particular could be adapted to be used as part of a project status QPI method.
3. The impact of agile inspections on the project resources was minimal with no impact on the project schedule.

www.manaraa.com

Research Question 3

The third research question was what is the in-process (before testing) project impact based on product quality that requires a communication mechanism or method not currently part of the structure of established project management reporting practices? This study used the currently existing executive project status communication tool of EVM and modified it to add a project quality component in addition to the cost and schedule component. A QPI method was developed that attributed to quality status the same characteristics as schedule and cost status (expected status verses actual status). The QPI produced an in-process status prior to the test phase of the project.

QPI is not currently part of the established project management structure of EVM, so in that sense it clearly satisfied the research question. Further, because it had the same construct as the existing SPI and CPI in Earned Value Management, it was capable of adding information to executives without the executives having to learn a new terminology relating to software development, quality, or project management.

Research Question 3 Findings.

1. QPI was a new quality communication mechanism for project status.
2. QPI had the same characteristics as the existing project status method of EVM and can be considered the third performance indicator of EVM (the first two being CPI and SPI).
3. Presenting project quality in terms of QPI required no additional training of executive management to correctly interpret the QPI status information.

www.manaraa.com

Conclusions

As stated at the beginning of the chapter, when the project researched in this study completed releasing their requirements documents, the project status was on schedule and within budget according to the parameters set by the organization. Using the Quality Performance Index (QPI) method, this study determined that the following information applied to the project:

1. The QPI for the project was 0.29. In the Earned value Management methodology, no project can recover from a performance index that low. To continue, it must be re-baselined.
2. A schedule slide of 3.6 months was projected.
3. An additional resource impact of 14,764 hours was projected.

Had the QPI method been available at that time, the executive team reviewing the project status would have know that the project was in major trouble despite the green status being reported by the project manager.

Summary

The results from the use of the QPI Method on the selected project were quite dramatic. The use of the QPI method cost the project an additional 540 minutes (9.0 hours) of effort. The dividends from that effort were way out of proportion on the positive to the effort involved. This was in spite of the fact that the inspection conditions were less than ideal. Three reviewers were requested to participate in the inspections; only two did and only one inspected all the documents. This is what happened in the real world of resource conflicts within organizations. It was actually a very good test as to the

www.manaraa.com

effectiveness of the QPI method in groups that are struggling, which is where these project status problems usually occur. From a research point-of-view, more data going into the methodology would have definitely been preferred; see suggestions for further research in the next chapter.

If the QPI method had been incorporated into the project methodology, then it would have told the executives to expect approximately a 4-month slide in the schedule when they still had three months to go on their current schedule. The QPI of 0.29 was at such a low value that projects with similar SPI and CPI values do not recover without re-baselining. This would lead management to be very skeptical about any attempt at a recovery plan within existing cost and schedule constraints. The agile inspection method was vindicated by validating the defect counts that came out of the methodology against the NIST defects model. The next chapter will include a summary of the study, suggestions for further research and potential influences for social change.

www.manaraa.com

Chapter 5: Summary, Conclusions and Suggestions for Further Research

This chapter represents the conclusions and implications derived from the results of testing the three research questions, It covers (a) the general research conclusions, (b) the application development implications of the research findings, (c) the management implications of the research findings , (d) recommendations for further research, and (e) the implications for social change.

General Research Conclusions

The literature review revealed a plethora of research and practical implementation activity in the problem area over the last 40 years. Five of its major areas were aimed at the three research questions. The QPI method was developed as a result of attempting to gain a better understanding of all the research in disciplines not previously integrated and the implementations, and thus bring about a comprehensive viewpoint.

Using deliverable inspection data as a measure of project status—as opposed to product status—caused the project and management teams to change their viewpoints on quality. They could now ask new questions about the defect-generating capability of the project or organization, as well as about the concept of a certain known defect density that the project or organization could live with given the resource requirements necessary to further reduce the defect density.

With the introduction of the QPI method, a more integrated (holistic) approach could be applied to project management. QPI gave interested personnel a noninvasive method of determining project status, either in real-time for project and executive information, or as part of a process improvement discipline for the organization.

www.manaraa.com

Application Development Significance and Implications

The application development implication is that a project does have a measurable defect-generation capability. This capability has been factored into project estimates, whether consciously or unconsciously. This study heightens awareness of this fact and provides an incentive for projects to embrace this QPI method as a best practice—if nothing else, for self-protection against project surprises.

Agile Inspections

Agile inspections took fewer resources than formal inspections (Fagan or PEP). This affordability factor for information made the knowledge gained from the Agile Inspections very attractive to the educated project manager. The finding—that the Agile Inspection methodology is suggestive with just one reviewer—should boost the project manger's interest in implementing this technique in the project work breakdown structure (WBS). If implemented, it would also allow management more flexibility in allocating resources for agile inspections, depending on management's intention to use the data for project status. The more resources allocated, the more accurately the inspection data would reflect the quality of the deliverables inspected.

As further research progresses in the use of agile inspections as part of a cost effective component of the QPI method the following needs to be noted:

1. While the use of less than three inspectors and sometimes just one produced results that a software development organization could use, this study did not in itself validate the correctness of the data produced from such a process from a professional and academic viewpoint.

www.manaraa.com

2. When further research is conducted as suggested in the next section, additional care needs to be taken to advance the study of the validity of inspectors in a resource challenged environment.
3. The ultimate academic goal would be to produce a QPI construct that delivers rigid results of the same magnitude as CPI and SPI.

QPI Method

The QPI Method if implemented by the project manager provides another dimension of project status. What it provides for the project manager is an independent assessment of the status of the deliverables. Currently, when a deliverable is completed, the project status took full credit for the completion without any negative impacts. Using the QPI method provides a viewpoint that asserts there may be negative impacts on the project later on that would temper the project's taking full credit for the deliverable.

Management Significance and Implications

QPI Method

The origins of this study stemmed from a concern about the management implications of the perception of software quality as received when reviewing the status of software projects. The management implications of the QPI method addressed these concerns by executives on the validity of the software project status being presented to them. With the implementation of the QPI method, management has an independent assessment of an aspect of the project status that was previously not available.

www.manaraa.com

Executive Communications

The QPI method fits into the already existing program management structure of EVM. As a result, most of the managers who will be using the QPI method would not have to learn new concepts or technical terms. The implications of a high or low QPI were the same as a high or low SPI or CPI. The knowledge that a performance index in EVM required a planned value and an actual value would reinforce in management's project world view that there was a realistic expected defect density for a project with the implied recognition that a project could continue successfully even if there are known defects in the requirements.

Potential for Further Research

Research Question 1

Research Question 1 asks, "What are the requirements for software quality during project execution, prior to the test phase that will produce reliable predictions of future impacts on the cost and schedule commitments of the project?" Future studies by other researchers are suggested in the following areas:

1. Study additional IT projects.
2. Gather more inspection data.
3. Study Agile inspections results with more consistent support for inspector resources (between three and five inspectors).
4. Study projects that use different development methodologies.
5. Study projects that are on different Capability Maturity Model Integration maturity Levels.

www.manaraa.com

6. Study embedded software development projects.

Research Question 2

Research Question 2 asks, "What is a software quality measurement technique currently available that can be adapted as an in-process (before testing) project quality measure?" Future studies by other researchers are suggested in the following areas:

1. Other defect detection techniques that can increase the precision of the number of defects in a software development product.
2. Other quality assurance (QA) processes within the software development process that can benefit from the agile inspection data given its current limitations.
3. New QA processes to be created as a result of the additional information provided by the agile inspections.

Research Question 3

Research Question 3 asks, "What is the In-Process (before testing) project impact based on product quality that requires a communication mechanism or method not currently part of the structure of established project management reporting practices?" Future studies by other researchers are suggested in the following areas:

1. Develop a project quality indicator for non EVM project management methodologies.
2. Determine the affect of management's embracing the QPI method, on the actions management takes compared to non QPI management actions.

www.manaraa.com

The Impact of the QPI Method and Its Influences on Social Change

The ability of the QPI method to provide early and accurate information on the status of an IT software development project would have an enormous impact on the software development, software project management and software quality professions and the organizations utilizing these professions to produce working software. With the early guidance that the QPI method provides, decisions could be made about re-scoping or cancelling projects long before the disastrous results of building and testing the products shows the dire straits of the project.

An immediate impact would be millions if not billions of dollars in software development cost saved every year in IT software development and embedded software development budgets. With an increased bottom line, the software development organizations would be able to make investments in improvements of their operations, yielding even more savings and improvements.

The enhanced performance of the organizations would allow them to respond more quickly to external threats. Government organizations could spread the taxpayers' dollars over more projects that would enhance our security and liberty. Long-range business plans and commitments could respond in a more agile manner to changes in the market and/or regulatory environments. The money saved at institutions of higher learning could be refocused into more contemporary class content and pay professors and staff a more equitable wage when compared to industry.

www.manaraa.com

Summary

This study developed a new tool in the battle to control software development projects: the QPI method. Much as a velocity is composed of two components, speed and direction, so the application of the QPI method has two components, direction of the project and the magnitude of that direction. The study showed that a project direction can be obtained using the QPI method but in this study, the magnitude of that direction was only suggestible; more research is necessary to get statistically relevant magnitude information. The combination of the modification of EVM along with the minimal resource requirements needed to implement the QPI method has very positive social implications since large projects use EVM and have the potential for the most cost savings and efficient use of resources, both public and private.

www.manaraa.com

References

Abba, W. F. (1997). Earned Value Management-Reconciling Government and Commercial Practices. *Program Manager*. 01997114. Jan/Feb97. Vol. 26 Issue 1.

Alstad, J. (2004). Bottle and Sell WGS SCP Detailed Design Process. SEPG-261, Process Improvement Proposal. *Boeing Co*., Chicago, IL. March 26, 2004.

Anbari, F. T. (2003). Earned Value Project Management Method and Extensions. *Project Management Journal*. December 2003. p. 12-23.

Anthes, G. H. (2004). Quality Model Mania. *Computerworld*. March 08, 2004.

Auruml, A., Petersson, H. and Wohlin, C. (2002). State-of-the-art: software inspections after 25 years. *Software Testing, Verification and Reliability*. 12:133–154 (DOI: 10.1002/stvr.243). 2003.

B-SEPG (2004). Boeing Software Engineering Process Group – Wichita. *Boeing Intranet.* 2004.

Baldrige (2004). Baldrige National Quality Program. *National Institute of Standards and Technologies (NIST)*. Gaithersburg, MD 20899-1070.

Basili, V. R., McGarry, F.E., Pajerski, R., & Zelkowitz, M. V. (2002**).** Lessons learned from 25 years of process improvement: the rise and fall of the NASA software engineering laboratory. *Proceedings of the 24th International Conference on Software Engineering*. International Conference on Software Engineering. Association for Computing Machinery. New York, NY. P79-89.

Bauer, F. L. (1969). Software Engineering. *Report on a conference sponsored by the NATO Science Committee,* Garmish, Germany, October 7-11, 1968.

www.manaraa.com

BDS (2001). Boeing Data Services: Appendix A — Glossary and Acronym Definitions. *Boeing*. Retrieved from Boeing Intranet. 2001.

Billings, C. and Clifton, J. (1994). Journey to a Mature Software Process. *IBM Systems Journal*. Vol. 33 Issue 1. p. 46-61.

Blaine, J. D. (2004, July 1,). *Three Easy Metrics to Help Ensure Project Success*. Gilb London 2004 Evo Symposium. The New Connaught Rooms, Covent Garden, London,

Blanco, Lt. Cmdr. V. D. (2003, March/April). Earned Value Management: A Predictive Analysis Tool. *The Navy Supply Corps Newsletter*. Retrieved from http://findarticles.com/p/articles/mi_m0NQS/is_2_66/

Boehm, B. (1991). Software Risk Management: Principles and Practices. *Institute of Electrical & Electronics Engineering Software*, vol. 8, no. 1, pp. 32-41, Jan./Feb. 1991.

Boehm, B., & Turner, R. (2003). *Balancing agility and discipline: A guide for the perplexed*. Boston, MA: Addison-Wesley Professional. Pearson Education, Inc.

Boehm, B., Horowitz, E., Madachy, D., Reifer, D., C., Bradford K., Steece, B., Brown, A. Winsor, C. S., & Abts, C. (2000). *Software cost estimation with cocomo II*. Upper Saddle River, NJ. Prentice Hall PTR.

Boehm, B. (1981). *Software engineering economics*. Upper Saddle River, NJ: Prentice Hall PTR.

Borland (2006). Software Quality Management. *Borland Software Corporation*. April 2006. Retrieved from http://www.borland.com/

www.manaraa.com

Brandon Jr., D. M. (1998). Implementing Earned Value Easily and Effectively. *Project Management Journal*. June 1998. Vol. 29. Issue 2. p11-18

Bush, M. (1990*). Improving Software Quality: The Use of Formal Inspections at the Jet Propulsion Laboratory*. *Experience Report*. IEEE 1990. p.196-199.

Cass, D. J. (2000). Earned Value Programs for US Department of Energy Projects. *Cost Engineering*. Vol. 42. No. 2. February 2000. p.24-43.

Chang, A. S.-T. (2001). Defining Cost/Schedule Performance Indices and Their Ranges for Design Projects. *Journal of Management in Engineering*. April 2001. p. 122-130.

Charette, R. N. (2005). Why Software Fails: We waste billions of dollars each year on entirely preventable mistakes. *IEEE Spectrum*. September 2005.

CMD (2004). CMD Symphony. *CMD Corporation*. Retrieved from http://www.cmdcorp.com/index.html.

CMM (2007). Capability Maturity Model®. *Software Engineering Institute*. Carnegie Mellon University. Pittsburgh, PA.

CMMI (2007). Capability Maturity Model® Integration. *Software Engineering Institute*. Carnegie Mellon University. Pittsburgh, PA.

CMMI (2006). CMMI for Development, Version 1.2: Improving Processes for Better Products. *CMU/SEI-2006-TR-008*. Software Engineering Institute. Carnegie Mellon University. Pittsburgh, PA. August 2006.

Crockett, R. O. and McGregor, J. (2006). Six Sigma Still Pays Off At Motorola. *The Corporation. Business Week.* December 4, 2006.

www.manaraa.com

Conway, W. E. (2007). Chairman & CEO, Conway Management Co. Retrieved from http://www.conwaymgmt.com/.

Cooper, D. R., & Schindler, P. S. (2003). *Business research method* (8th ed.). New York, NY: McGraw-Hill.

Crymble, S. (2001). Quality Systems Overview: ITIL/ ISO/ CMM & Malcolm Baldrige. *Toronto Spin*. 2851 John St., PO Box 42073, Markham, ON L3R 5R7. Sept 20, 2001.

D&C (2006). Macroscope: Design and Construction. *Macroscope Productivity Centre*. Retrieved from http://macroscope45.web.boeing.com/DMRPEn/html/En_P_Phase04.html

DeMarco, T. (1997). *The deadline: A novel about project management*. New York, NY: Dorset House Publishing Company.

DeMarco, T and Lister, T. (1987). *Peopleware: Productive projects and teams*. New York, NY: Dorset House Publishing Company.

DeMarco, T. and Lister, T. (2003). *Waltzing with bears: Managing risk on software projects*. New York, NY: Dorset House Publishing Company.

DeMarco, T. (2001). *Slack: Getting past burnout, Busywork, and the myth of total efficiency.* New York, NY: Dorset House Publishing Company.

DeMarco, T. (1982). *Controlling software projects: Management, measurement and estimation*. Englewood Cliffs, NJ: Prentice-Hall.

DeMarco, T., & Plauger, P. J. (1979). *Structured analysis and system specification*. Upper Saddle River, NJ: Prentice Hall PTR.

DMR (2004). *DMR Consulting*. Fujitsu, Inc. 2004.

www.manaraa.com

Dromey, R. G. (1998). SOFTWARE PRODUCT QUALITY: Theory, Model, and Practice. *Software Quality Institute.* Griffith University, Nathan, Brisbane, QLD 4111 AUSTRALIA. Volume 15. Issue 02. 09/03/98

DSDM (2004). *Dynamic System Development Method*. DSDM Consortium, Ltd. 2004.

Evensmo, J. and Karlsen, Dr. J. T. (2004). Reviewing the Assumptions Behind Performance Indexes. *AACE International Transactions*. CSC.14. 2004. p. 14.1-14.7.

Finn, J. (2001). *Implementation of the Agile Review/Extreme Inspection Method*. Retrieved from e-mail to Tom Gilb from Jeff Finn, Microsoft. May 22, 2001.

Fujitsu (2002). *Macroscope.* Fujitsu consulting (Canada), Inc. Version 4.0. April 2002.

GAO (2008). Defense Acquisitions – Progress Made in Fielding Missile Defense, but Program Is Short of Meeting Goals. *GAO-08-448*, March 2008.

Gilb, T. (2009). Agile Specification Quality Control: Shifting emphasis from cleanup to sampling defects. *Testing Experience: The Magazine for Professional Testers*. March 2009. p. 87-93.

Gilb, T. (2005). *COMPETITIVE ENGINEERING: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*. *Elsevier Butterworth-Heinemann*. Linacre House, Jordan Hill, Oxford OX2 8DP, UK. 2005.

Gilb, T., & Gilb, K. (2004). Agile Reviews: the use of sampling for 'measurement' reviews (for requirements, design, test plans, test cases, code). *Specification Quality Control.* 2004.

Gilb, T. (1988). *Principles of software engineering management*. London, UK: Addison-Wesley: Pearson Education Limited.

www.manaraa.com

Gilb, T., & Graham, D. (1993). *Software inspections.* London, UK: Pearson Education Limited.

Gilray, J. J. (1996). *Applying the code inspection process to hardware descriptions. Hewlett-Packard Journal*, *47* (1), 68-72.

Glass, R. L., & DeMarco, T. (2006). *Software creativity 2.0.* Atlanta, GA. developer.* Books.

Harrington, H. J., & Harrington, J. S. (1995). *Total improvement management: The next generation in performance improvement*. New York, NY: McGraw-Hill.

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

Hedger, R. J. (2003). High Quality Low Cost Software Inspections. . *IBM Systems Journal,* Vol. 42, No. 2. 2003. p.397-399.

Hille, H. (1997). Fundamentals of a Theory of Measurement. Lecture delivered on the Spring Meeting of the German Physical Society. *Ludwig-Maximilians University*. Munich, Germany. March 1997.

Hopkins, W. G. (2000). Quantitative Research Design. *Dept of Physiology and School of Physical Education. University of Otago, Dunedin, New Zealand*. *Sports Science Perspectives: Research Resources*. 2000.

Houdek, F., Schwinn, T. and Ernst, D. (2002). Defect Detection for Executable Specifications – An Experiment. *International Journal of Software Engineering and Knowledge Engineering.* Vol. 12, No. 6. 2002. p. 637-655.

www.manaraa.com

IBM (2003). Using the RUP for small projects: Expanding upon Extreme Programming. *Rational, IBM*. 2003.

ISACA (2009). Control Objectives for Information and related Technology (COBIT). *ISACA*. 3701 Algonquin Rd., Suite 1010, Rolling Meadows, IL. 60008. 2009.

I (2006). *Macroscope: Implementation*. Macroscope Productivity Centre. Retrieved from http://macroscope45.web.boeing.com/DMRPEn/html/En_P_Phase05.html

ITIL (2006). Information Technology Infrastructure Library. *Office of Government Commerce. Norwich, Norfolk NR7 0HS UK.*

Jones, C. (2004). Software Project Management Practices: Failure versus Success. *CrossTalk: The Journal of Defense Software Engineering*. October 2004. P: 5-9.

Kan, S. H. (2002). Software Quality Metrics Overview. *Addison-Wesley Professional*. Pearson Education, Boston, MA. Dec. 20, 2002.

Knutson, C. and Carmichael, S. (2001). Verification and Validation. *Embedded Systems Programming.* June 2001. p. 24-36.

KovacsBurns, K. (2005). Non-experimental Research Designs. *Nursing 503*, University of Alberta, Alberta, Canada. Oct. 5, 2005.

Lee, M-C. and Chang, T. (2006). Applying TQM, CMM and ISO 9001 in Knowledge Management for software development process improvement. *International Journal of Services and Standards* 2006 - Vol. 2, No.1 pp. 101 - 115

Lewis, J P. (1995). The Project Manager's Desk Reference. *Chicago*. *IRWIN Professional Publishing.* 1995.

www.manaraa.com

Lindeman, J., Romero, B. and Tavel, H. (1999). Putting Executive Scorecards on the Web With SAS® Software. *Proceeding of the 24th Annual SAS® Users Group International Conference.* April 11-14, 1999. Miami Beach, FL. P. 244-248. 1999.

Lutenist, T. (2006).Capability Maturity Model. *Everything2* Feb. 22, 2006. Retrieved from http://everything2.com/title/Capability%2520Maturity%2520Model

Macroscope (2002). 'Methodware' Magic Quadrant for Software Processes. *Boeing Co., Chicago, IL*. 2002.

Macroscope (2004). Macroscope Version 4.5. *Boeing Co., Chicago, IL.*. Retrieved from Boeing Co. Intranet.

Major, J., Pellegrin, J. F., Pittler, A. W. (1998). Meeting the Software Challenge: Strategy for Competitive Success. *Research Technology Management*. Jan/Feb98, Vol. 41 Issue 1. p. 48-56.

ME1 (2005). ME1-Measurement-Program-Overview. *Boeing Co., Chicago, IL* Retrieved from Boeing Co. intranet.

Manganelli, R. L. & Klein, M. M. (1994). *The Reengineering Handbook: A Step-By-Step Guide to Business Transformation.* AMACOM, American Management Association, New York, NY. 1994.

McConnell, S. (2002). Cost of Software Quality: Delivering Software Project Success. *Construx Software Builders*, Inc. 2002.

Mendel, T., Garbani, J. P., Ostergaard, B. & van Veen, N. (2004). Implementing ITIL: How To Get Started. *Best Practices*. Forrester. Sept. 21, 2004.

www.manaraa.com

NASA (2006). Goal-Question-Metric (GQM). *Experience Factory*. *Software Engineering Laboratory.* NASA. 2006.

NASA (2007). What Is Earned Value Management? *NASA*. 2007.

NAS (1998). Total Ownership Cost: Implementation Guidebook Version 1.0 (Revised). *Naval Air Systems. US Government*. 1998.

Naur, P. and Randell, B. (1969). Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, *Scientific Affairs Division, NATO*. 1969. 231pp.

NIST (2002). The Economic Impact of Inadequate Infrastructure for Software Testing. *United States Department of Commerce, National Institute of Standards and Technology (NIST).* May 2002.

OE (2006). Macroscope: Opportunity Evaluation. *Macroscope Productivity Centre*. Retrieved from http://macroscope45.web.boeing.com/DMRPEn/html/En_P_Home.html

Olson, T. (2008). Private discussions between Tim Olson and the author concerning proprietary methods. *Museum of Flight, Seattle, WA*. March 18, 2008.

Orci, T. (1999). Software Metrics for Process Improvement Experiments. *European Systems & Software Process Improvement and Innovation (EURO SPI^2)* - EuroSPI99. 1999.

PA (2005). Macroscope: Preliminary Analysis. *Macroscope Productivity Centre*. Retrieved from http://macroscope45.web.boeing.com/DMRPEn/html/En_P_Phase02.html

www.manaraa.com

Park, R, E., Goethert, W. B. & Florac, W. A. (1996). *Goal-Driven Software Measurement — A Guidebook. HANDBOOK.* CMU/SEI-96-HB-002 Software Engineering Institute Carnegie Mellon University. Pittsburgh, PA. August 1996.

PCMM (2008). People Capability Maturity Model: Version 2.0. Frequently Asked Questions. *Software Engineering Institute. Carnegie Mellon University.* Pittsburgh, PA. 2008.

Petrasch, R. (1999). The Definition of 'Software Quality': A Practical Approach. *FastAbstract ISSRE.* 1999.

PMBOK (2004). *A Guide to the Project Management Body of Knowledge, 3rd Edition*. Project Management Institute. Newtown Square, PA. November 2004.

Praxiom (2007). ISO 9001 2000**.** *Praxiom Research Group Limited*, Edmonton, Alberta. 2007.

Primavera (2005). Earned Value Graphs reporting from Primavera Project Manager for Enterprise. *Primavera*. Dec. 12, 2005.

QuantumPM (2004). *QuantumPM, LLC*. CMD Corporation. 2004.

Ragavan, C. & Hook, C. S. (2005). Fixing the FBI. *Us News and World Report*. March 28, 2005.

Rational (2001). Rational Unified Process: Process Made Practical. *Rational Software Corporation.* G-060E; Rev. 9/01: Part No. 800-024698-000. 2001.

Roberts, D. (2010). Google China Hacking Brouhaha Raises Business Concerns. *Business Week*. January 15, 2010. Retrieved from

www.manaraa.com

http://www.businessweek.com/globalbiz/blog/eyeonasia/archives/2010/01/google_brouhaha.html

Ross, J. (1999). Ways of Approaching Research: Quantitative Designs. *R&RA Session 2.* April 10, 1999. Retrieved from http://www.fortunecity.com/greenfield/grizzly/432/rra2.htm.

Rothman, J. (2000). What Does It Cost You To Fix A Defect? And Why Should You Care? *Rothman Consulting Group.* October, 2000. Retrieved from http://www.jrothman.com/Papers/Costtofixdefect.html

Rus, I., Halling, M. and Biffl, S. (2003). Supporting Decision-Making in Software Engineering with Process Simulation and Empirical Studies. *International Journal of Software Engineering and Knowledge Engineering.* Vol. 13, No. 5. p. 531-545. 2003.

SBA (2009). Section H – Special Contract Requirements: SECURITY & CMM CONTRACT REQUIREMENTS. *US Small Business Administration*. Washington, DC. Retrieved from http://www.sba.gov/idc/groups/public/documents/sba_program_office/sba_ocio_security_cmm_req.pdf

SCEA (2003). Earned Value Management Systems (EVMS): Tracking cost and schedule performance on projects. Developed by TASC. *The Society of Cost Estimating and Analysis*. Unit 5. Module 15. Sept 24, 2003.

Schmitt R. B., (2005). New FBI Software May Be Unusable: A Central Feature of the Agency's $581-Million Computer Overhaul Aimed at Coordinating Anti-Terrorism Efforts is Reportedly Inadequate. *LA Times*. Jan. 13, 2005.

www.manaraa.com

SEI (1994). The Capability Maturity Model: Guidelines for Improving the Software Process. *Software Engineering Institute. Carnegie Mellon University*. Addison-Wesley. Reading, MA. 1994.

SEI (2009). *Software Engineering Institute*. Pittsburgh, PA Retrieved from http://www.sei.cmu.edu/

Sethi, V., & King, W. R. (1998). *Organizational transformation through business process reengineering: Applying the lessons learned*. Upper Saddle River, NJ: Prentice-Hall Inc.

Shore, B. (2008). Systematic Biases and Culture in Project Failures. *Project Management Journal*. Dec. 2008. p. 5-16.

Simmons, E. (2002). Implementation of the Agile Review/Extreme Inspection Method. Retrieved from e-mail to Tom Gilb from Erik Simmons. *Intel*. January 9, 2002.

Sinks, S. (2007). *Quantitative Research Design*. *University of Minnesota Duluth*. Duluth, MI. April 10, 2007

SML (2004). GQM Method *Application. Software Measurement Laboratory* SML@b, FIN, IVS, AG Software Engineering, Otto-von-Guericke-University of Magdeburg, Magdeburg, Germany. 2004.

Snow, A. P. and Keil, M. (2001). The Challenge of Accurate Software Project Status Reporting: A Two Stage Model Incorporating Status Errors and Reporting Bias. *IEEE.* September 2001. 0-7695-0981-9/01. 2001.

Software Testing/Quality Conference (1995). Software Testing/Quality. *Computer Conference Analysis Newsletter.* June 6, 1995. Issue 365. p. 1-10.

www.manaraa.com

Solomon, P. and Young, R. (2006). Performance-Based Earned Value. *Wiley-IEEE Computer Society Press*. December 2006.

Solomon, P. (2006). Practical Performance-Based Earned Value. *Systems and Software Technology Conference,* Track 5: 2:25 – 3:10 p.m. Room 251 D-F. 2 May 2006.

SPC (2004). Software Productivity Center. *Software Productivity Center Inc*. 2004.

Swartz, J. (2007). Chinese hackers seek U.S. access. *USA Today – Technology*. March 11, 2007.

Tingey, M. (1996). Comparing ISO 9000, Malcolm Baldrige, and the SEI CMM for Software: A Reference and Selection Guide. *Prentice Hall PTR* (October 3, 1996).

TQE (2009). Malcolm Baldrige National Quality Award. *Total Quality Engineering Inc*.15997 Grey Stone Rd, Poway, CA 92064. Retrieved from http://www.tqe.com/baldrige.html.

TQM (2007). Total Quality Management. *Wikipedia: The Free Encyclopedia*. 2007.

Tripathy, P. (2007). Proposals: Effective Executive Summary. *A Technical Communication Community*. June 15, 2007.

Tully, C. (2002). Halve Software Development Waste. *Workshop on Grand Challenges for Computing Research,* UK Computing Research Committee. October 2002.

Twaites, G. and Sibilla, M. L. (2002). Software Engineering in an SEI Level-5 Organization. *International Journal of Reliability, Quality and Safety Engineering*. Vol. 9, No. 4. 2002. p. 347-365. 2002.

Van Genuchten, M., Cornelissen, W., & Van Dijk, C. (1998). Supporting Inspection with an Electronic Meeting System. *Journal of Management Information Systems.* Winter 97/98, Vol. 14 Issue 3. p. 165-178.

www.manaraa.com

Appendix A Tables

Table A1.

Quantification of Quality Models Properties

Quality Model	IT Relevance 1-Holistic 3-General 5-Specific	Level of Abstraction 1-Low 3-Moderate 5-High	Quality Usability Index (QUI)
TCO	5.0	1.0	5.0
ITIL	4.2	1.7	7.1
CMM/CMMI	4.0	2.3	9.2
CobiT	3.5	2.7	9.5
Six Sigma	2.7	3.5	9.5
ISO 9000	2.2	3.9	8.6
Malcolm Baldrige	1.7	4.3	7.3
Scorecards	1.0	5.0	5.0

www.manaraa.com

Table A-2.

Quantification of Methodology Models Properties

Methodology Model	Ability to Execute 1-Low 3-Medium 5-High	Completeness of Vision 1-Niche Player 3-Average 5-Visionaries	Methodology Usability Index (MUI)
SPC	1.7	1.7	2.9
DSDM	2.3	1.7	3.9
CMD	2.3	3.5	8.1
Rational RUP	4.5	2.8	12.6
Macroscope	4.2	4.5	18.9

www.manaraa.com

Table A-3.

Project Management Capability based on Model Analysis

Methodology Model	Software Productivity Centre - SPC	DSDM	CMD	Rational RUP	Macro-scope
MUI	2.9	3.9	8.1	12.6	18.9

Quality Model	QUI	Product Capability Index (PCI)				
TCO	5.0	14	20	40	63	95
ITIL	7.1	21	28	57	90	135
CMM/ CMMI	9.2	27	36	74	116	174
CobiT	9.5	27	37	76	119	179
Six Sigma	9.5	27	37	76	119	179
ISO 9000	8.6	25	34	69	108	162
Malcolm Baldrige	7.3	21	29	59	92	138
Score cards	5.0	14	20	40	63	95

www.manaraa.com

Table A-4.

Quality, Project Management and Communication Evaluation

0. Emphasis area not part of approach
1. Minimal mention of emphasis area
2. Emphasis Area included
3. Detailed applications of emphasis area
4. Major player in emphasis area
5. Industry Leader in emphasis area

Approach	Quality	Project Management	Executive Communication	Effectiveness Penetration Index (EPI)
Total Quality Management (TQM)	5	3	1	3.0
Total cost of Ownership (TCO)	5	2	2	3.0
Capability Maturity Model (CMM)	3	3	1	2.3
Six Sigma	5	3	1	3.0
ISO 9001	4	2	1	2.3
Malcolm Baldrige National Quality Program	4	2	4	3.3
Fujitsu Macroscope	1	5	0	2.0
CMD Symphony	2	3	1	2.0
Dynamic System Development Method (DSDM)	2	3	1	2.0
Software Productivity Center (SPC)	3	3	1	2.3

www.manaraa.com

Table A-4. (con't)

Quality, Project Management and Communication Evaluation

Approach	Quality	Project Management	Executive Communication	Effectiveness Penetration Index (EPI)
Rational RUP	2	3	1	2.0
Progress Metric	2	4	2	2.7
Effort Metric	2	4	1	2.3
Cost Metric	1	4	3	2.7
Results Metric	2	4	2	2.7
Trouble Reports	3	3	1	2.3
Requirements Stability	5	4	1	3.3
Size Stability	2	2	1	1.7
Computer Resource Utilization (CRU)	1	2	1	1.3
Training	1	3	1	1.7
Software Inspections	5	5	2	4.0
Earned Value Management (EVM)	3	5	5	4.3
Defect Performance Index (DPI)	5	5	3	4.3
Performance based-Earned Value Management	5	5	3	4.3

www.manaraa.com

Table A-5.

Earned Value Management Attributes

EVM Attribute	Acronym	Equation	Description
Actual Cost or Actual Cost of Work Performed	AC or ACWP		The cumulative actual cost spent to a given point in time to accomplish an activity, work package or project and to earn the related value.
Budget at Completion	BAC		The total budget baseline for the activity work package.
Earned Value or Budgeted Cost of Work Performed	EV or BCWP		The cumulative earned value spent to a given point in time. It is the amount budgeted for performing the work that was accomplished.
Planned Value or Budgeted Cost of Work Scheduled	PV or BCWS		The time phased budget baseline.
Cost Performance Measurement			Compares EV to AC
Schedule Performance Measurement			Compares EV to PV

www.manaraa.com

Table A-5. (con't)

Earned Value Management Attributes

EVM Attribute	Acronym	Equation	Description
Variances			Generally based on cumulative data and also called inception-to-date data and project to date data
Cost Variance	CV	CV = EV – AC CV = BCWP – ACWP (Evensmo, 2004)	Budgetary conformance of actual cost of work performed
Schedule Variance	SV	SV = EV – PV SV = BCWP – BCWS (Evensmo, 2004)	This measures the conformance of actual progress to the schedule. Brandon Jr. labels this Schedule Variance (monetary units)
Budget Variance	BV	BV = BCWS – ACWP (Evensmo, 2004)	
Spend rate/burn rate			Average AC per time period.
Baseline Schedule at Completion	SAC		
Planned Accomplish-ment rate	PV Rate	PV Rate = BAC/SAC	Average PV per time period.
Time Variance	TV	TV = SV/(PV Rate)	Brandon Jr. labels this Schedule Variance (time units)

www.manaraa.com

Table A-5. (con't)

Earned Value Management Attributes

EVM Attribute	Acronym	Equation	Description
Cost Variance Percent	CVP	CVP = CV/EV	This is the measure of the budgetary conformance of actual cost of work performed. Brandon Jr. has the equation more technically correct in that CV has to be multiplied by 100 to get a number in percent.
Schedule Variance Percent	SVP	SVP = SV/PV	This is the measure of the conformance of actual progress to the schedule.
Schedule Variance Percent based on Earned Value	SVPev	SVPev = SV/EV	
Cost Performance Index	CPI	CPI = EV/AC	This is the measure of the budgetary conformance of actual cost of work performed.
Schedule Performance Index	SPI	SPI = EV/PV	This is the measure of the conformance of actual progress to the schedule.
Critical Ratio	CR	CR = CPI x SPI	This is also called the Cost-Schedule Index: CSI

www.manaraa.com

Table A-5. (con't)

Earned Value Management Attributes

EVM Attribute	Acronym	Equation	Description
Estimate to Complete	ETC	ETC = (BAC – EV)/CPI	This is the estimated cost to complete the remainder of the project.
Variance at Completion	VAC	VAC = BAC – EAC	Estimated cost overrun or underrun at the completion of the project.
Time Estimate to Complete	TETC		
Cost Performance Index (alt)	CPI	CPI = % Complete/% Spent	
Estimate at Completion (alt)	EAC	EAC_3 = AC/ (%Complete)	

www.manaraa.com

Table A-5. (con't)

Earned Value Management Attributes

EVM Attribute	Acronym	Equation	Description
Estimate at Completion Or Cost Estimate at Completion	EAC CEAC	$EAC_1 = AC + ETC$ $EAC_2 = AC + BAC - EV = BAC - CV$ $EAC_3 = AC + (BAC - EV)/CPI = BAC/CPI$ $EAC_{3v} = AC + (BAC - EV)/CR$ (Evensmo, 2004) $EAC_4 = BAC$ $EAC_5 = EAC_s = BAC/CR$	EAC_2 used when past performance is not deemed a good indicator of future performance. EAC_3 used when past performance is deemed a good indicator of future performance. A variation on EAC_3 is to use CR instead of CPI. EAC_4 is rarely achieved if CPI is poor. EAC_5 used for an EAC adjusted for schedule performance.
Variance at Completion	VAC	$VAC = BAC - EAC$	Estimated cost overrun or underrun at the completion of the project.
Time Estimate to Complete	TETC		

www.manaraa.com

Table A-5. (con't)

Earned Value Management Attributes

EVM Attribute	Acronym	Equation	Description
Time Estimate at Complete	TEAC	$TEAC_1 = AT + TETC$ $TEAC_2 = SAC - TV$ $TEAC_3 = SAC/SPI$ $TEAC_4 = SAC$ $TEAC_5 = TEAC_s = SAC/CR$	$TEAC_2$ Used when past performance is not deemed a good indicator of future performance. $TEAC_3$ is used when past performance is deemed a good indicator of future performance. $TEAC_4$ is rarely achieved if CPI is poor. $TEAC_5$ is TEAC adjusted for cost performance.
% Complete		% Complete = EV/BAC	
% Spent		% Spent = AC/BAC	
Cost Performance Index (alt)	CPI	CPI = % Complete/% Spent	
Estimate at Completion (alt)	EAC	$EAC_3 = AC/(\%$ Complete)	
Time Estimate at Complete (alt)	TEAC	$TEAC_3 = AT/\%$ Complete	
Cost Performance Index (alt)	CPI	CPI = Planned Unit Cost /Actual Unit Cost	

www.manaraa.com

Table A-6.

Characteristics of C/SPIs (Chang, 2001)

	S1	S2	C1	C2	S3	S4	C3	C4
Index	SPI(p)	SPI (pf)	CPI(p)	CPI (pf)	SPI(m)	SPI(mf)	CPI(m)	CPI (mf)
Level	Project	Project	Project	Project	Mile-stone	Mile-stone	Mile-stone	Mile-Stone
Cost/ Schedule	Schedule	Schedule	Cost	Cost	Schedule	Schedule	Cost	Cost
Period	Month	To Completion	Month	To Completion	To-date	To Completion	To-date	To Completion
Current/ Forecast	Current	Forecast	Current	Forecast	Current	Forecast	Current	Forecast
Judgment	N/A	N/A	N/A	N/A	N/A	Allowed	N/A	Allowed

Table A-7.

Reject DD for Project Inspected Deliverables

Deliverable	Inspected DD	Reject DD Threshold	Within DD Range or (Outside DD Range)
1. P100: Opportunity Evaluation	1.6	0.75	(0.8)
2. P120: Disposition	0.5	0.75	YES
3. P120: Relocation	0.4	0.75	YES
4. P120: Acquisition	1.1	0.75	(0.4)
5. P120: Cable and Wire Network and Design	1.3	0.75	(0.5)
6. P120: Ticketing Infrastructure	1.2	0.75	(0.4)
7. P130: Objectives of the System	4.0	0.75	(3.3)

www.manaraa.com

Table A-7. (con't)

Reject DD for Project Inspected Deliverables

Deliverable	Inspected DD	Reject DD Threshold	Within DD Range or (Outside DD Range)
8. P140: Context of the System	2.2	0.75	(1.4)
9. P150: Definition of the Subject	2.2	0.75	(1.5)
10. P170/180/190 (FA1): Service Request	0.2	0.75	YES
11. P170/180/190 (FA2): Approval	0.3	0.75	YES
12. P170/180/190 (FA3): System Manager	0.9	0.75	(0.1)
13. P170/180/190 (FA4): Metrics and History	0.2	0.75	YES
14. P170/180/190 (FA5): Catalog and Reference Data	1.2	0.75	(0.4)
15. P170/180/190 (FA6): Product Requisition	0.9	0.75	(0.2)
16. P170/180/190 (FA9): Dispatch Routing	1.7	0.75	(1.0)

www.manaraa.com

Table A-8.

Project Requirements Deliverables Defect Review Results

P+ Deliverable	Content Pages	Reviewer A		Reviewer B		Reviewer C		Total Defects	Defects/ Page
		Defects	Minutes	Defects	Minutes	Defects	Minutes		
1. P100:	32	25	30	0	45	DNR	DNR	50	1.6
2. P120: Disp	39	10	30	DNR	DNR	DNR	DNR	20	0.5
3. P120: Relo	53	10	30	DNR	DNR	DNR	DNR	20	0.4
4. P120: Acq	18	10	30	0	30	DNR	DNR	20	1.1
5. P120: Cable	16	10	30	DNR	DNR	DNR	DNR	20	1.3
6. P120: Ticket	17	10	30	0	25	DNR	DNR	20	1.2
7. P130:	4	8	15	DNR	DNR	DNR	DNR	20	4.0
8. P140:	11	12	20	DNR	DNR	DNR	DNR	24	2.2
9. P150:	9	10	40	DNR	DNR	DNR	DNR	20	2.2
10. P170...(FA1)	38	3	60	DNR	DNR	DNR	DNR	6	0.2
11. P170…(FA2)	6	1	15	DNR	DNR	DNR	DNR	30	0.3
12. P170…(FA3)	14	6	15	DNR	DNR	DNR	DNR	12	0.9
13. P170…(FA4)	47	4	30	DNR	DNR	DNR	DNR	8	0.2
14. P170…(FA5)	47	28	30	DNR	DNR	DNR	DNR	56	1.2
15. P170…(FA6)	26	12	20	DNR	DNR	DNR	DNR	24	1.7
16. P170…(FA9)	7	6	15	DNR	DNR	DNR	DNR	12	1.7
Total	384	165	440	0	100	0	0	330	0.9

Notes: DNR = Did not review; Total Defects = Two times the max Defects detected by a single reviewer

www.manaraa.com

Table A-9.

Key Words by Literature Type

Key Words	Journal	Professional Publication	Professional Technical Book	Gov't Technical Publication	Article	Technical Conference	Industry Publication	Totals
Metrics		1		2	1	1	5	10
Earned Value	3	3	1	2		1	1	11
Software Quality	2	2		1	1	1	3	10
Software Metrics	2	3	3			2	1	11
Software Project Failure	2	1	2	2			2	9
Software Project Measurement	2		1	1	1	1	3	9
Software process quality	4	1	3	4	1	2	4	19
Software Project Success	3	1	6	3		1	3	17
Quality	1	2	1	3			2	9
Software Inspections	4	1	1		1		2	9
Other	1	3	4	1			0	9
Totals	24	18	22	19	5	9	26	123

www.manaraa.com

Appendix B Figures

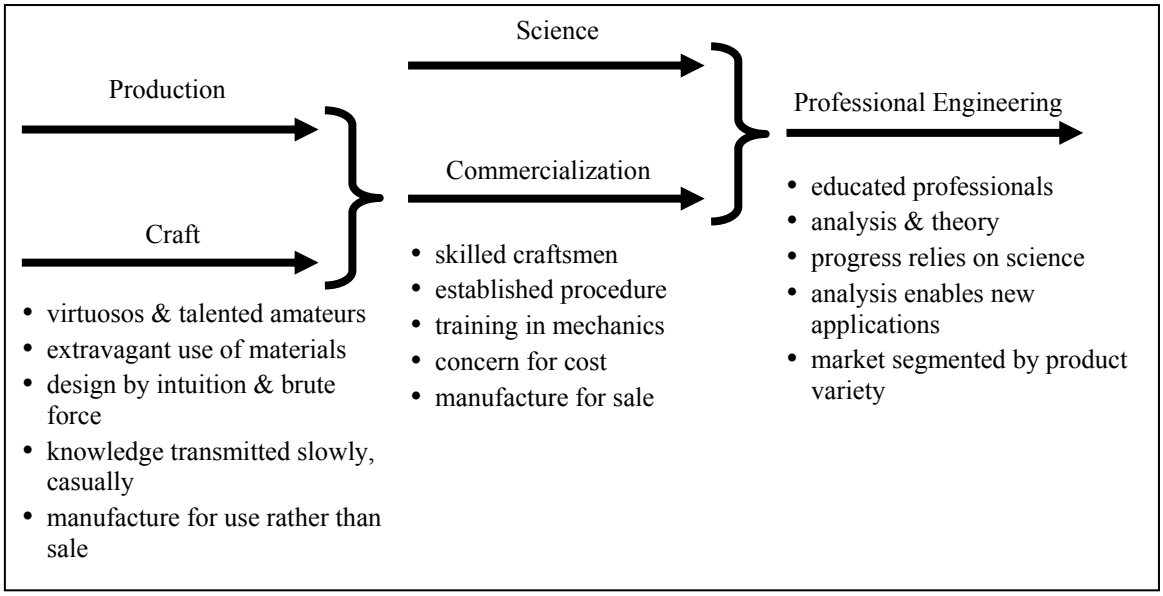

Figure B-1. Engineering evolution paradigm.

www.manaraa.com

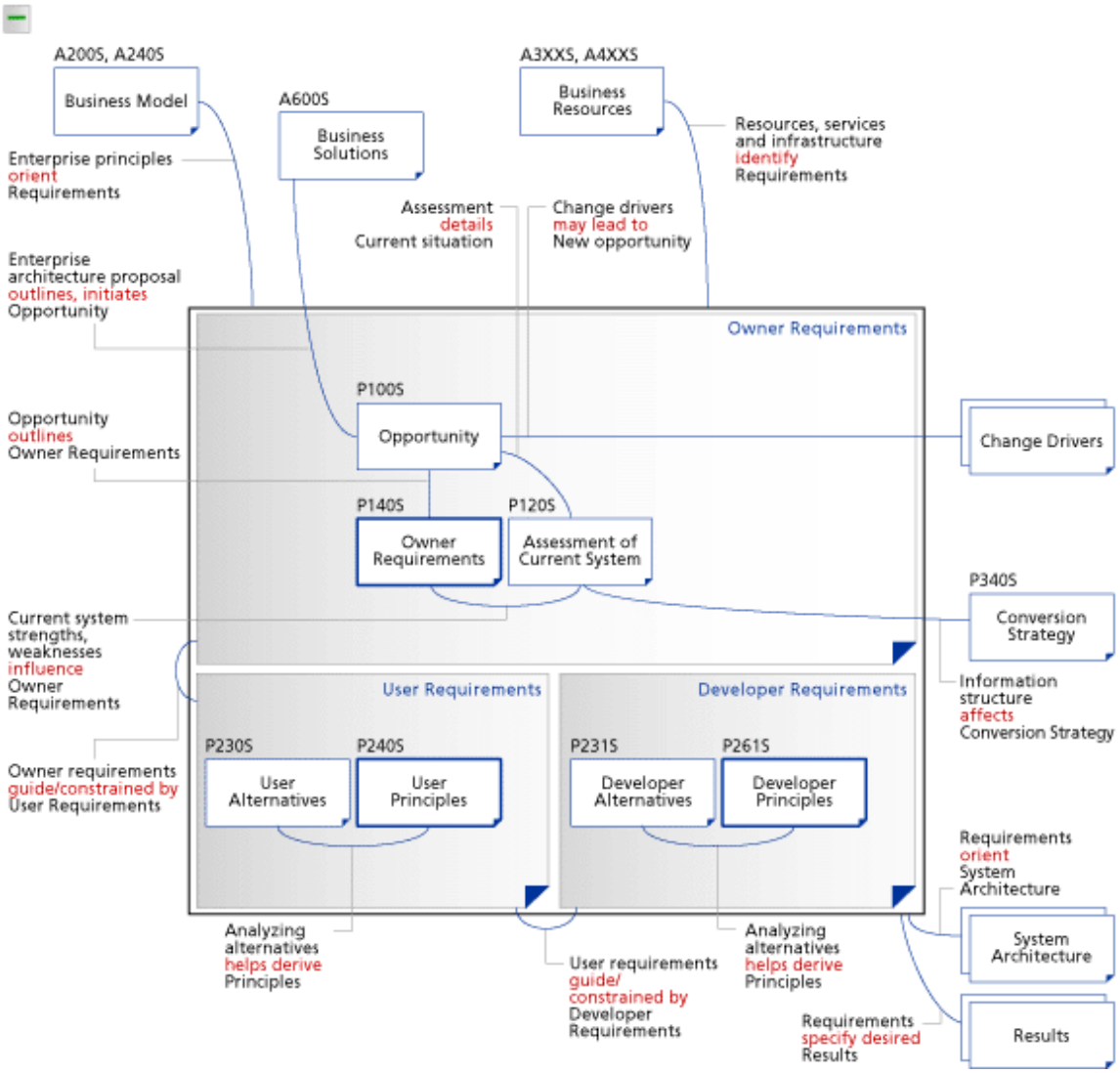

Figure B-2. Macroscope 4.5 requirements deliverables

www.manaraa.com

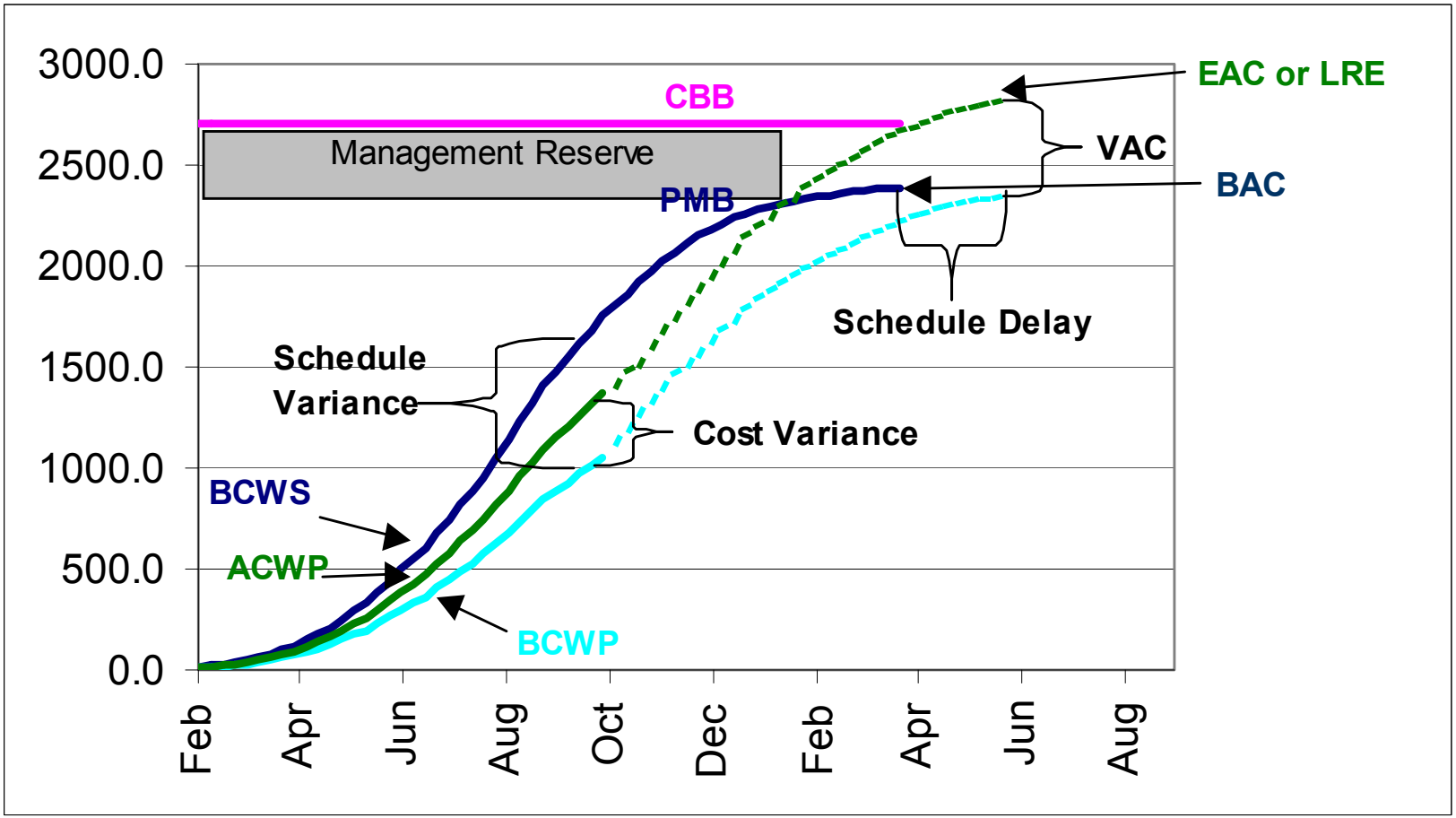

Figure B-3. EVM graphical analysis

www.manaraa.com

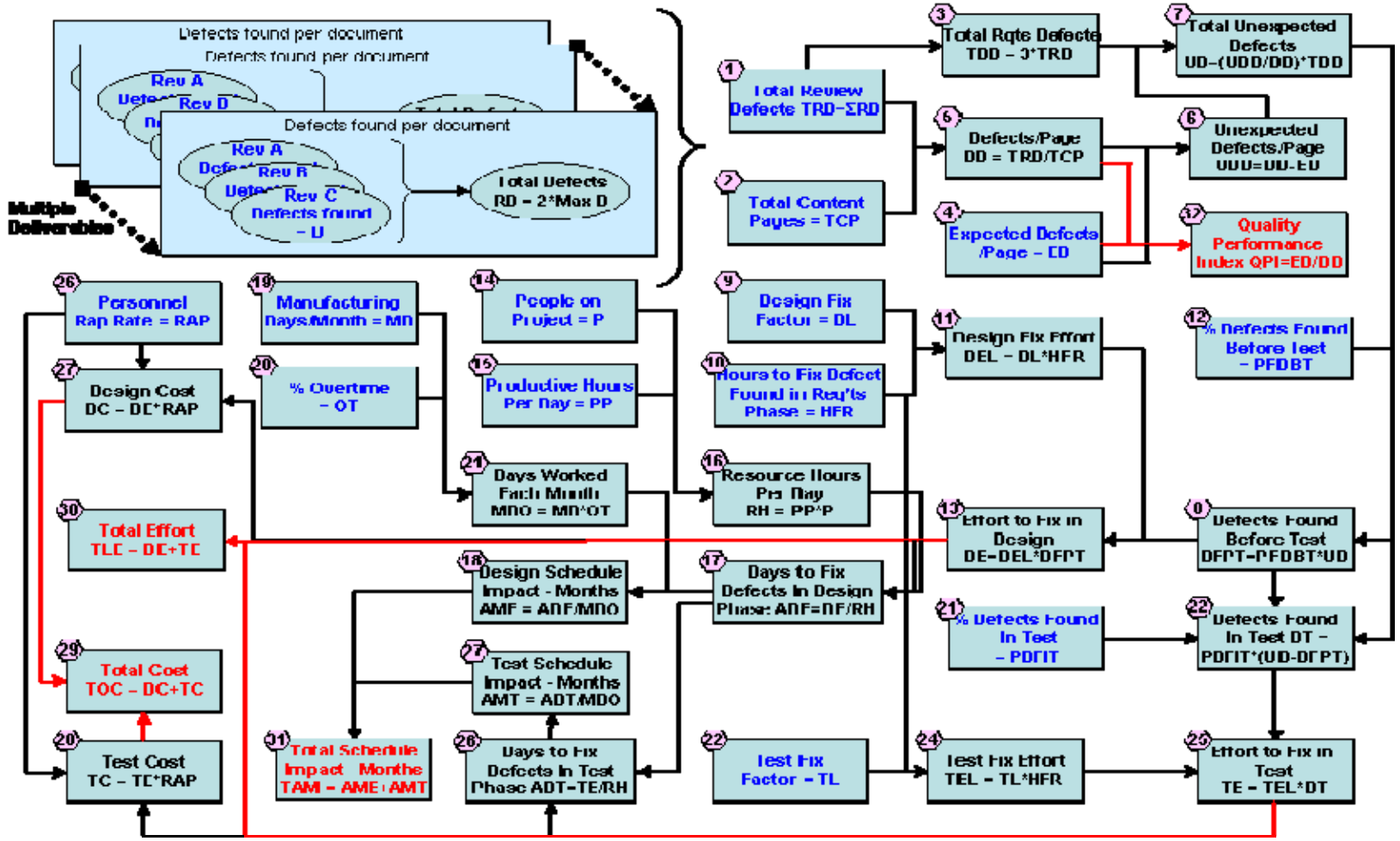

Figure B-4. Process flow from agile inspections for defects to determining project impact

www.manaraa.com

1-Total Reviewer Defects Detected (TRD)	330	2-Total Content Pages (TCP)	384
3-Total Requirements Defects (TDD) = 3*TRD	990	4-Expected Defects/Content Page (ED)	0.25
5-Defects Detected/Content Page (DD) = TRD/TCP	0.86	11-Effort Leverage for Defect Correct in Design (DEL) = DL*HFR	10
6-Unanticipated Major Defects Detected/Content Page (UDD) = DD – ED	0.61	24-Effort Leverage for Defect Correct in Test (TEL) = TL*HFR	100
7-Unanticipated Major Defects (UD) = (UDD/DD)*TDD	702	15-Personnel Productivity (PP) in (hours/day)	6.5
8-Defects found prior to test (DFPT) = PFDBT*UD	351	14-Number of people on project (P)	29
13-Effort to fix in Design (DE) = DEL*DFPT	3510	19-Manufacturing Days per month (MD)	21
17-Additional Days (ADE) to fix defects found prior to test = DE/RH	18.6	12-% Defects found and fixed before test (PFDBT)	50%

Figure B-5. IV and DV values for project under study

www.manaraa.com

18-Additional Months (AME) Schedule Impact prior to test = ADE/MDO	0.9	16-Resource Hours used per day RH = PP*P	188.5	32-QPI = ED/DD	0.29
22-Defects found in test (DT) = PDFIT*(UD-DFPT)	116	21-% of Defects found in Test (PDFIT)	33%		
25-Effort to fix in test (TE) = TEL*DT	11583.0	20-% Overtime (OT)	4%	9-Leverage of Requirements Defect found in Design (DL)	10
26-Additional Days (ADT) to fix defects found in Test=TE/RH	61.4	21-Days worked each month (MDO) = MD*OT	21.84	23-Leverage of Requirements Defect found in Test (TL)	100
27-Additional Months (AMT) Schedule Impact = ADT/MDO	2.8	10-Hours to Fix a Defect in the Requirements Phase (HFR)	1	26-Rap Rate (RAP)	$110
31-Total Months Impact (TAM) = AME + AMT	3.7	30-Total Effort (TLE) = DE + TE	15093	29-Total Cost (TOC) = DC +TC	$1,660,230
27-Design Cost (DC) = DE*RAP	$386,100	28-Test Cost (TC) = TE*RAP	$1,274,130		

Figure B-6 IV and DV values for project under study (con't)

www.manaraa.com

Appendix C – Additional Variables and Equations

RD = Total Defects found by the reviewers. The rule of thumb for estimation is that the total unique defects found is twice the number that is that maximum of the largest number of defects found by each reviewer. Equation C-1 is RD = (Max D)*2.

TCP = Total number of content pages reviewed. This is a cumulative number of pages reviewed by the reviewers as more documents are reviewed. Equation C-2: TCP $= \sum_{j=1}^{j=n} CP_j$

TD = Total Defects in the document. The rule of thumb for estimation is that the total number of defects in the document is three times the number detected. Equation C-3 is TD = 3*RD.

TDD = Summation of Total Defects found in each document. This is a cumulative number of defects detected by the reviewers as more documents are reviewed. Equation C-4: TDD = where n = the number of documents reviewed.

TRD = Summation of Total Defects found in each document. This is a cumulative number of defects detected by the reviewers as more documents are reviewed. Equation C-5 is TRD = $\sum_{j=1}^{j=n} RD_j$ where n = the number of documents reviewed.

DD = Detected Defects per Content Page. This is the total number of defects detected divided by the total number of content pages. Equation C-6 is DD = TDD/TCP.

UDD = Unanticipated Defects per Content Page. The inspected defects per content page minus the expected defects per content page. Equation C-7 is UDD = DD-ED.

www.manaraa.com

QPI = Quality Performance Index. QPI is the ratio of the expected quality to the detected quality. Equation C-8 is QPI = ED/DD.

UDD = Unanticipated Defects per Content Page. This is the number of defects per content page that were not expected in the planning of the project. Equation C-9 is UDD = DD – ED.

www.manaraa.com

Appendix D – Quality Review Checklist – Requirements (Example)

Reviewer's Name: A
Document Name: P150
Content Pages: 9
Document Type: Requirements
Project Phase: PA

Major Defects Requirements Checklist - there is a major defect if the requirement:

a. Is Ambiguous to the intended Readership
b. Is Not Clear enough to test.
c. Has Design specs (= 'how to- be good') mixed in
    a. Mixed up in the Requirements
    b. (Req'ts. = 'how good - to be') not how to be good!
    c. MARK Design as "D": Except if it is a conscious Design Constraint - which is a requirement type
4. Does not conform to the organization's requirement's template (if any)

Major Defect Definition:

A defect severity where there is potential for:

a. High loss (>10 lost engineering hours)
b. Later downstream impact (test, field).

Results:

Time Spent in Review: 0:40 min Pages Reviewed: 9
Total Defects: 10 Major Defects: 9 Design Defects: 1

www.manaraa.com

Appendix E Major Quality Models

Total Quality Management (TQM). In the mid-80s, the US Department of Defense initiated the extensive use of the term Total Quality Management (TQM). TQM has never been clearly defined in spite of the many books and articles written about each. Each one seems to be slightly different. However, the following are common elements associated with a basic TQM process (Harrington & Harrington, 1995): Top management involvement; flow down to all levels of management; gain understanding of the customer's requirements; prevent errors from occurring; implement statistical disciplines for root cause analysis and process management; teach employees to team; train employees in problem solving; focus on improving the process for better quality, not the people as the cause of poor quality; develop fewer but better suppliers; establish quality and customer-related metrics; focus on all stakeholders, internal and external, and solve problems and make decisions, at all levels of the enterprise, with teams.

A survey by MAPI (Manufacturers' Alliance for Productivity and Innovation) reported the following results from those organizations surveyed who had implemented TQM, to one degree or another (Harrington and Harrington, 1995): 40% of the companies surveyed reported a significant improvement; 45% of the companies surveyed reported some improvement; 15% of the companies surveyed reported marginal improvement, and 0% of the companies surveyed reported no improvement.

A survey by the American Society for Quality Control reported that 31% of those organizations surveyed had made some mistakes. The following areas frequently were cited (Harrington and Harrington, 1995): not beginning sooner; failing to make it a

www.manaraa.com

priority; making it a project rather than a continuing process; expecting immediate financial payback; not involving everyone, and not emphasizing metrics enough.

Total Cost of Ownership (TCO). TCO has been used in many different businesses and organizations. The US government, in particular, has been stressing the use of the concept and could definitely be considered a sponsor of TCO. It is also known as Total Ownership Cost and consists of the total cost to operate an enterprise or deliver a product. This includes hardware, software, personnel and all other resources necessary to accomplish the organization's mission (NAS, 1998). TCO looks at all aspects of any product or service. Unlike other Quality approaches that may view Quality from a single aspect. TCO tries to integrate all the data from all the aspects of delivering a quality product or service.

Capability Maturity Model (CMM). Capability Maturity Model was created by the Software Engineering Institute which is part of Carnegie Mellon University, Pittsburgh, PA. SEI- Capability Maturity Model is a collection of best practices for software development and maintenance. The theoretical foundation for Capability Maturity Model and all Capability Maturity Model derivatives is the concept of establishing professional engineering practices in software development in the same way that they exist in other engineering professions. The chronological/functional flow of a technical profession from craftsman to engineering (Lutenist, 2006) is shown in Figure B-1 (see Appendix B).

Capability Maturity Model allows companies to assess their practices and compare them to those of other companies. SEI- Capability Maturity Model's focus is on process maturity. There are five levels of maturity that have auditable criteria for

www.manaraa.com

completion: Initial, Repeatable, Defined, Managed, and Optimizing (SEI, 1994). Capability Maturity Model is a very detailed model. It was developed specifically to address the challenges of software development organizations. As an organization moves up the maturity scale the focus shifts to continuous improvement. The model itself can be used as a template for performing an internal organizational self-assessment.

Capability Maturity Model Integration (CMMI). Capability Maturity Model Integration is an update of SEI-CMM by the Software Engineering Institute. It combines the Software Capability Maturity Model (SW-CMM), the Systems Engineering Capability Maturity Model (EIA-731) and the Integrated Product Development Capability Maturity Model into an integrated CMM renamed Capability Maturity Model Integration. People Capability Maturity Model (PCMM) has not been incorporated into Capability Maturity Model Integration although People Capability Maturity Model Version 2.0 takes into account the more advanced aspects of the higher levels of Capability Maturity Model Integration maturity (PCMM, 2008). Capability Maturity Model Integration is a very detailed and is designed specifically for large software development organizations. It places even more emphasis on continuous improvement, not on just being certified to a certain level. Capability Maturity Model Integration doesn't address the basic IT operations issues, such as security, change management, configuration management, capacity planning, troubleshooting, and help desk functions.

Capability Maturity Model Integration sets goals but does not specify how to accomplish these goals. A glaring example is in the area of requirements. Capability Maturity Model Integration identifies the need to perform the function of requirements analysis. Capability Maturity Model Integration does not specify or give guidance in

www.manaraa.com

how to successfully accomplish that task. Such an approach allows the principles to remain common while the implementation details may vary. Such flexibility is desirable recognizing that while processes may be common, implementation of process using specific procedures can vary based on methodology and/or technology.

Control Objectives for Information and Related Technology (CobiT). CobiT is the product of the Information Systems Audit and Control Association and the IT Governance Institute. CobiT is a set of guidelines for IT processes, practices and controls that is mainly intended to be used for purposes of audit. CobiT stresses integrity, reliability and security with the goal reducing risk (CobiT, 2009). The following four major domains are addressed: Planning and Organization; Acquisition and Implementation; Delivery and support, and Monitoring. Like CMM, CobiT has maturity levels, but it has six to CMM's five. CobiT functions well as a checklist for IT. CobiT's approach allows IT organizations to deal with risks not explicitly addressed by other quality models. CobiT also assists IT organizations in passing audits. CobiT has a capability to interact well with other quality models, and is specifically structured to interact with the ITIL model. CobiT has IT General Controls and Application [software] Controls in it processes that run in parallel to each other. As the maturity level increases, measurement plays a larger part in the IT process management decisions. Score cards first appear at level 3. Efficiency and effectiveness are used by a level 4 organization (ISACA, 2009).

Information Technology Infrastructure Library (ITIL). After the Falkland Islands war with Argentina, the United Kingdom Office of Government Commerce commissioned a study on how to develop a common approach to IT. It had become

www.manaraa.com

apparent that during the war organizations were using the same terms and meaning entirely different things. Out of this effort was born ITIL. Now, Pink Elephant and other suppliers are conducting classes and holding certification training in ITIL in the USA. ITIL has developed a set of IT processes and best practices for IT service management and operations (such as service-desk, incident, change, capacity, service-level and security management). ITIL is now a well established, mature, and detailed quality model. It focuses on IT production and operational quality issues.

ITIL has a perspective of service delivered and IT software applications are part of a service delivery. As part of any service delivery an Evaluation Plan is generated. The Evaluation Plan contains provisions for consistently measuring performance of a service change and providing metrics associated with that change.

Six Sigma. Six Sigma was originally developed by Motorola Inc. and has lately been successfully implemented on a corporate scale by General Electric. Six Sigma identifies statistical methods for quality and process improvement. It can focus on quality from a customer's point of view or a user's point of view. Motorola chose to focus on the manufacturing aspects of quality and attempted to drive product quality to a six sigma defect error rate for manufacturing operations. GE chose to focus on six sigma defect error rate from the customer's point of view. Initially, Motorola had some difficulty integrating the discipline of six sigma with the need to respond quickly to innovation. They have since over come that and have successfully integrated the two business approaches (Crockett and McGregor, 2006). Among other things, Six Sigma can be used to define service levels and measures variances from those levels. Projects typically go through five phases: define, measure, analyze, improve, and control. This is

www.manaraa.com

almost identical to Deming's ubiquitous Plan, Do, Check, Act (PDCA) cycle. The six sigma approach focuses on how to develop and apply methods and principles for the creation of defect-free products or services, rather than trying to improve existing ones.

Six Sigma's diagnostic phases are designed to identify root causes of problems and undesirable situations. Once a root cause of a process problem is identified, then a plan can be developed to update the process to eliminate the problem, or at least mitigate the effects of the problem. With the case of software development projects, understanding the root cause of process failures in the development lifecycle is a key to successful project management of the ongoing projects.

ISO 9001. ISO is an acronym for International Standards Organization, which is European based. Compliance to the ISO set of standards has become an economic requirement for trade with the European Union. It is a well established and mature quality model. ISO is a set of standards for quality management systems. ISO 9001 is the software standard. They are customer-oriented and an organization can be audited against them for compliance. These standards tend to focus on control, repeatability and good documentation of processes as opposed to products. Traditional software development process emphasizes testing techniques, but show weakness in planning. However, it does not satisfy the requirement of the user to cause failure cost heavily. ISO 9001 Quality Management System, Capability Maturity Model (CMM) and Total Quality Management (TQM) are all quality management technologies. These technologies can be applied to the software quality industry (Lee and Chang, 2006).

Malcolm Baldrige National Quality Program. The Malcolm Baldrige award is a U.S. government program and is sponsored by the National Institute of Standards and

www.manaraa.com

Technology, U.S. Department of Commerce. Enhancing the competitiveness, quality, and productivity of US organizations for the benefit of all residents is the goal of the award (Baldrige, 2004). The Malcolm Baldrige award has a quality measurement in seven areas: Leadership, Strategic Planning, Customer Focus, Measurement Analysis and Knowledge Management, Workforce Focus, Process Management and results (TQE, 2009). Each area is rated on approach, execution and results. There is a minimum score of zero and a maximum of 100 in each area. The award addresses all aspects of a business or organization. The approach is general enough that it can be applied at a top company or organization level as well as be used at a supervisor's level. Its criteria are not specifically addressing software or IT, but the categories can be used by organizations in those disciplines.

The Malcolm Baldrige Award process methodology is often compared against the ISO 9001 and Capability Maturity Model Integration/Capability Maturity Model Integration methodologies for achieving high levels of software quality (Tingey, 1996). The Malcolm Baldrige process has a Measurement Analysis area of quality but does not get as specific as some other methodologies. It has generally been acknowledged as a more holistic methodology than others that are more specific (Crymble, 2001).

www.manaraa.com

Appendix F – Software Development Process Models/Methodologies

Fujitsu Macroscope. Macroscope was developed by DMR, a Canadian consulting firm. The first methodology product was an application development methodology that had a waterfall Software Development Life Cycle (SDLC); it was called P^{+}. The product was expanded to include other areas of software development such as Architecture (A^{+}). Macroscope is a development methodology that bases its quality and schedule on deliverables at each phase of development (Fujitsu, 2002). Macroscope is extremely detailed and requires that all activities be defined by deliverables as evidence of successful completion. It has expanded from an initial concentration on software development (ProductivityCentre) to also include Architecture (ArchitectureLab), Project Management (ManagementSuite), Strategic Business (StrategyForum) and Value Management (ResultStation).

CMD Symphony. CMD Symphony, not surprisingly, is a product of the CMD Corporation. It is a complete set of development methodology paths, and each path is also a full life cycle methodology. CMD provides a complete project environment including clearly defined roles and responsibilities, deliverable descriptions, templates and examples, concise task descriptions, and Meta model definitions and effects. Each methodology describes the systems development life cycle as a series of phases such as Strategic Visioning, Analysis and Architecture, Design, Construction, Transition and Production. CMD's terminology includes a Phase which involves a formal review and confirmation and is composed of a series of sub-phases which involve one or more tasks. The tasks are where responsibilities are assigned; specific deliverables are identified, and

www.manaraa.com

detailed schedules are maintained. Each task is described in terms of its objectives, input, deliverables, dependencies, roles and responsibilities, Meta model effects, and other references. Tasks can have Subtasks which are specific procedures that describe the necessary detailed activities required to meet the objectives of the task. There is additional documentation such as Management Guidelines, Client/Server Concepts and Principles, Synergy/Framework Techniques, Product Reference and Vendor Identification, Evaluation and Selection, as well as a Sample Deliverables Framework.

Dynamic System Development Method (DSDM). The DSDM Consortium has developed its own methodology. DSDM uses an iterative process based on prototyping and involves the users throughout the project life cycle (DSDM, 2004). The most recent Version is 4.2. There are some purported benefits in the DSDM approach and they include the following (DSDM, 2004): the users are more likely to claim ownership of the solution; the risk of building the wrong system is greatly reduced; the final system is more likely to meet the users' real business requirements; the users will be better trained as their representatives will define and coordinate the training required, and the implementation is more likely to go smoothly because of the cooperation of all parties concerned throughout the development.

Software Productivity Center (SPC). As with the previous methodology, here too the product name is the company name. Software Productivity Center, Inc. (SPC) is the creator of Software Productivity Center**.** SPC consists of the major elements of Process Improvement, Software Requirements, Project Management and Planning, Configuration Management, Quality Assurance, Testing and Managed Outsourcing.

www.manaraa.com

These elements are incorporated into the tools produced by Segue and Merant as well as the Estimate Professional tool.

Rational RUP. Rational RUP (Rational Unified Process) is a product of Rational, which is now owned by IBM. According to IBM, Rational RUP was never intended as a one-size-fits-all process (IBM, 2003). RUP consists of the major capabilities of business modeling, requirements, analysis and design, implementation, test, configuration and change management, deployment, project management and environment (Rational, 2001). It identifies phases and attempts to integrate shills and disciplines over the life cycle of the project. The various capabilities are linked together functionally so that they resemble the Deming Plan/Do/Check/Act (PDCA) cycle.

www.manaraa.com

Appendix G Metrics

Macroscope Metrics. Macroscope version 4.5 defines metrics in terms of the success of achieving the application quality requirements. In Macroscope, Quality Criteria consists of a set of defined and documented rules and conditions which are used to decide whether the total quality of a specific product is acceptable or not (Macroscope, 2004). Macroscope focuses on Information Technology applications which have characteristics of the quality requirements below. Quality requirements are defined in terms of certain quality characteristics. The characteristics that have been adapted from ISO9126 are efficiency, functionality, maintainability, portability, reliability, security and usability (Macroscope, 2004). Quality metrics are the quantitative measurement of the characteristics of each quality requirement. Included in these metrics are rating levels that define how acceptable the measured values are.

Information Technology Infrastructure Library (ITIL) Metrics. ITIL is focused on Information Technology processes to develop and maintain the IT infrastructures of businesses and governmental organizations. ITIL focuses on service and responsiveness of the IT organization to their customers and business partners. ITIL therefore recommends that service quality metrics should be introduced from the very beginning (Mendel et al, 2004). This should be coupled with a strong focus on automating recurring tasks.

Progress Metrics. Progress indicators provide information on how well the project is performing with respect to planned task completions and keeping schedule commitments (USC, 2001, p31). Tasks are scheduled and then progress is tracked to the

www.manaraa.com

schedules. Metrics are collected for the activities and milestones identified in the project schedules. Metrics on actual completions are compared to those of planned completions to determine whether there are deviations to the plan. The difference between the actual and planned completions indicates the deviations from the plan. Each project identifies tasks for which progress metrics will be collected. The completion criteria for each task must be well defined and measurable. The project should establish range limits (thresholds) on the planned task progress for the project. The thresholds are used for management of software development risk. The metric is depicted by the cumulative number of planned and actual completions (or milestones) over time. Each project is expected to produce multiple progress charts for different types of tasks, different teams, etc.

Effort Metrics. Effort indicators allow the software manager to track personnel resources (USC, 2001, p32). They provide visibility into the contribution of staffing to project costs, schedule adherence, product quality and the amount of effort required for each activity. Effort indicators include trends in actual staffing levels, staffing profile by activity or labor category, or a profile of unplanned staff loses. Effort indicators may be used by all levels of project software management to measure the actual profile against the plan. Each level of management forms a profile for its area of control and monitors the actual profile against the plan.

Determining the number of staff needed at any one time is an important function performed by software management. By summing the number of staff during each reporting period, the composite staffing profile for the project can be determined. These indicators are applied during all life-cycles phases, from project inception to project end.

www.manaraa.com

Effort metrics are to be collected and reported at least on a monthly basis. The effort and cost metrics are related. By convention, effort metrics are non-cumulative expenditures of human resources, and cost metrics are cumulative levels of effort as tracked by earned value. Thus, cost metrics are a cumulative depiction of effort. This metric is depicted by a plot of monthly actual versus planned effort.

Cost Metrics. Cost management is an important activity for the success of a project, and labor is the primary component of software development cost (USC, 2001, p. 33). Managers must define the work in their area, determine the skill level required to perform the work, and use productivity estimates and schedule constraints to determine budgeted costs over time. Use staff-hours to measure cost, rather than dollars. The dollars per staff-hour varies over time and by labor category, and the conversion is made only by Finance. Cost is related to the effort indicator, with cost defined as an accumulation of effort expenditures. (The total project cost also includes non-labor costs, but they are not tracked here.) Only those projects using earned value can report the earned value quantities.

A Work Breakdown Structure (WBS) is established to define the structures that will be used to collect the costs. The WBS identifies separate elements for requirements, design, documentation, code and unit test, integration, verification, and system testing. Costs can also be segregated by component, function, or configuration item. Work packages are derived from the WBS. Costs are allocated to work packages using an earned value method. This system allows managers to track the actual costs and measure them against the budget for their respective areas of responsibility. This metric is

www.manaraa.com

depicted with actual and budgeted quantities are derived from an earned value system, and are shown in terms of staff-hours.

Results Metrics. Review Results indicators provide insight into the status of action items from life-cycle reviews (USC, 2001, p34). The term *Action Item* (AI) refers to inspection defects and customer comments. Reviews include the following: formal inspections of software documents or code; formal customer milestones, e.g., SSR, PDR, CDR, or TRR; informal peer evaluations of products, e.g., walkthroughs, technical reviews, or internal PDRs; management reviews, and process reviews, e.g., SQA audits, SEI CMM assessments, or the causal analysis from formal inspections.

There are standards for some reviews, as well as procedures for conducting them. For example, formal inspections result in assertion logs that document the minor and major defects uncovered by the inspection process. Therefore, standard review result indicators for formal inspections are: counts of minor/major defects; rates of defect detection (e.g., assertions per inspection meeting minute, defects per inspected document page, or defects per KSLOC of code inspected), and defect status (e.g., age of open defects, number of open/closed defects, and breakdown by defect categories). A customer-conducted review such as a Preliminary Design Review (PDR) generates AIs that must be closed before approval of the Software Design Document. Therefore, standard review result indicators for a PDR are the number of comments written and their status (open, closed, and age). Review metrics record the AIs identified in the review findings and track them until they are resolved. These metrics provide status on both products and processes. Review results are not to be used to evaluate the performance of individuals. Review Results are collected and reported at least monthly at every stage of

www.manaraa.com

the software life cycle, but preferably weekly for key AIs. This metric is depicted by the cumulative counts of AIs written and closed by reporting period.

Trouble Reports (TR) Metrics. TR indicators provide managers with insight into the quality of the product, software reliability, and the effectiveness of testing (USC, 2001, p35). They also provide information on the software development process. The terms defect and problem are used interchangeably herein. Monthly tracking of TR indicators shows the project's trends in the following areas: the rate at which TRs are being written and resolved; the type and severity of the TRs; relationship between the number of TRs and the number of test cases passed or the number of test steps passed; the TR density (the number of TRs per unit size); the number of defects in each software application/unit.

TR indicators are applicable only in the following life cycle stages (and each release of the software within these stages, and during the informal and formal test segments of these stages) (1) application test and integration, (2) system test, (3) acceptance test. Thus the TR indicators are applicable only to defects during the operation or execution of a computer program. Due to the shortness of testing periods, and the dynamics involved between the test team and the implementation team that analyzes the TRs and fixes the defects, the TR indicators are generally evaluated on a weekly basis. The terms open and closed are defined as follows: Open - the problem has been reported, and Closed - The investigation is complete and the action required to resolve the problem has been proposed, implemented, and verified to the satisfaction of all concerned. In some cases, a TR will be found to be invalid as part of the investigative

www.manaraa.com

process and closed immediately. This metric is depicted by a cumulative count of total, open, and closed TRs over time (weekly periods).

Requirements Stability Metrics. Requirements Stability provides an indication of the completeness, stability, and understanding of the requirements (USC, 2001, p36). It indicates the number of changes to the requirements and the amount of information needed to complete the requirements definition. A lack of requirements stability can lead to poor product quality, increased cost, and schedule slippage. Requirements stability indicators are in the form of trend charts that show the total number of requirements, cumulative changes to the requirements, and the number of TBDs over time. A TBD refers to an undefined requirement. Based on requirements stability trends, corrective action may be necessary.

Requirements stability is applicable during all life-cycles phases, from project inception to the end. The requirements stability indicators are most important during requirements and design phases. Requirements stability metrics are collected and reported on a monthly basis. This metric is depicted by the total number of requirements, the cumulative number of requirements changes, and the number of remaining TBDs over time. It may be desirable to also show the number of added, modified and deleted requirements over time.

Size Stability Metric. Software size is a critical input to project planning. The size estimate and other factors are used to derive effort and schedule before and during a project (USC, 2001, p. 37). The software manager tracks the actual versus planned software product size. Various indicators show trends in the estimated code size, trends by code type, the variation of actual software size from estimates, or the size variation by

www.manaraa.com

release. Size stability is derived from changes in the size estimate as time goes on. It provides an indication of the completeness and stability of the requirements, the understanding of the requirements, design thoroughness and stability, and the capability of the software development staff to meet the current budget and schedule. Size instability may indicate the need for corrective action. Size metrics are applicable during all life-cycle phases. Size metrics are collected and reported on a monthly basis, or more often as necessary. This metric is depicted by plotting planned and currently estimated software size per release over time. Besides showing re-allocation of software content between releases, this also shows the growth in the total estimated size.

Computer Resource Utilization. Computer Resource Utilization indicators show whether the software is using the planned amount of system resources (USC, 2001, p. 38). The computer resources are normally CPU time, I/O, and memory. For some software, the constraints of computer resources significantly affect the design, implementation, and testing of the product. They can also be used to replan, re-estimate, and guide resource acquisition. Computer resource utilization is planned during the requirements activity and reviewed during the design activity. Resources are monitored from the start of implementation activity to the end of the life cycle.

For memory utilization, the unit of data is the byte, word, or half-word. For CPU time, the unit of data is either MIPS (millions of instructions per second), or the percentage of CPU time used during a peak period. For I/O time, the unit of data is the percentage of I/O time used during a peak period. Resource Utilization data is collected and reported at least monthly, with the period between collection and reporting becoming shorter as the software system nears completion and a better picture of software

www.manaraa.com

performance can be seen. Note that the resource utilization is normally an estimate until integration occurs, at which time the actual data is available. This metric depicts the CPU and memory use as a percent of available and the maximum allowed.

Training Metrics. Training indicators provide managers with information on the training program and whether the staff has necessary skills (USC, 2001, p. 39). A trained staff is a commitment. The manager must ensure that the staff has the skills needed to perform their assigned tasks. The objective of the training indicator is to provide visibility into the training process, to ensure effective utilization of training, and to provide project software managers with an indication of their staff's skill mixture. The manager should investigate the deviations in the number of classes taught from the number of classes planned, and the deviation of the number of staff taught to the planned number. The quality of the training program should also be determined from completed course evaluation sheets. The number of waivers requested and approved for training should also be tracked. This metric depicts a graph of the total monthly attendance of personnel attending training classes. It represents the sum of the number of personnel attending all classes.

www.manaraa.com

Appendix H – Notification of Approval to Conduct Research – Lawrence Day

Subject : Notification of Approval to Conduct Research-Lawrence Day

Date : Mon, Apr 12, 2010 10:11 AM CDT
From : IRB@waldenu.edu
To : ldayx001@waldenu.edu

CC : research@waldenu.edu, Raghu.Korrapati@waldenu.edu

Dear Mr. Day,

This email is to serve as your notification that Walden University has approved BOTH your dissertation proposal and your application to the Institutional Review Board. As such, you are approved by Walden University to conduct research.

Please contact the Office of Student Research Support at research@waldenu.edu if you have any questions.

Congratulations!

Jenny Sherer
Operations Manager, Office of Research Integrity and Compliance

Leilani Endicott
IRB Chair, Walden University

www.manaraa.com

Appendix I – Major Defects Requirements Checklist

Major Defects Requirements Checklist - there is a major defect if the requirement:

d. Is Ambiguous to the intended Readership

e. Is Not Clear enough to test.

f. Has Design specs (= 'how to- be good') mixed in

d. Mixed up in the Requirements

e. (Req'ts. = 'how good - to be') not how to be good!

f. MARK Design as "D": Except if it is a conscious Design Constraint - which is a requirement type

5. Does not conform to the organization's requirement's template (if any)

Major Defect Definition: A defect severity where there is potential for:

c. High loss (>10 lost engineering hours)

d. Later downstream impact (test, field).

www.manaraa.com

Appendix J – Boeing Co. Data Use Agreement

DATA USE AGREEMENT

This Data Use Agreement ("Agreement"), effective as of*4127109,* is entered into by and between Lawrence Day ("Data Recipient") and The Boeing Company ("Data Provider"). The purpose of this Agreement is to provide Data Recipient with access to a Limited Data Set ("LDS") for use in connection with the study entitled "A Systems Approach to the Integration of Software Quality into Software Project Management" ("Study").

1. Responsibilities of Data Recipient. Data Recipient agrees to:

a. Use or disclose the LDS only as permitted by this Agreement or as required by law;

b. Use appropriate safeguards to prevent use or disclosure of the LDS other than as permitted by this Agreement or required by law;

c. Report to Data Provider any use or disclosure of the LOS of which it becomes aware that is not permitted by this Agreement or required by law;

d. Upon prior written approval of Data Provider, require any of its subcontractors or agents that receive or have access to the LOS to agree to the same restrictions and conditions on the use and/or disclosure of the LOS that apply to Data Recipient under this Agreement;

e. Not use the information in the LOS to identify or contact any individuals who are data subjects; and

f. The Data Recipient shall comply with all Data Provider policies including, without limitation, all procedures that address the external release of proprietary information such as PRO-3439.

2. Permitted Uses and Disclosures of the LOS. Data Recipient may use and/or disclose the LOS for its research activities only.

3. Term and Termination.

a. Term. The term of this Agreement shall commence as of the Effective Date and shall continue for so long as Data Recipient retains the LOS, unless sooner terminated as set forth in this Agreement. Data Provider hereby reserves the right to withdraw from the Study at any time in its sale discretion.

b. Termination by Data Recipient. Data Recipient may terminate this agreement at any time by notifying the Data Provider and returning or destroying the LOS.

c. Termination by Data Provider. Data Provider may terminate this agreement at any time by notifying Data Recipient.

d. For Breach. Data Provider shall provide written notice to Data Recipient within ten (10) days of any determination that Data Recipient has breached a material term of this Agreement. Data Provider shall afford Data Recipient an opportunity to cure said alleged material breach upon mutually agreeable terms. Failure to agree on mutually agreeable terms for cure within thirty (30) days shall be grounds for the immediate

www.manaraa.com

termination of this Agreement by Data Provider.
e. Effect of Termination. Sections 1,4,5, 6(e) and 7 of this Agreement shall survive any termination of this Agreement under subsections c or d.
4. Miscellaneous.
a. Change in Law. The parties agree to negotiate in good faith to amend this Agreement to comport with changes in federal law that materially alter either or both parties' obligations under this Agreement. Provided however, that if the parties are unable to agree to mutually acceptable amendment(s) by the compliance date of the change in applicable law or regulations, either Party may terminate this Agreement as provided in section 6.
b. No Third Party Beneficiaries. Nothing in this Agreement shall confer upon any person other than the parties and their respective successors or assigns, any rights, remedies, obligations, or liabilities whatsoever.
c. Counterparts. This Agreement may be executed in one or more counterparts, each of which shall be deemed an original, but all of which together shall constitute one and the same instrument.
d. Headings. The headings and other captions in this Agreement are for convenience and reference only and shall not be used in interpreting, construing or enforcing any of the provisions of this Agreement.
IN WITNESS WHEREOF, each of the undersigned has caused this Agreement to be duly executed in its name and on its behalf. July 23, 2009.

DATA PROVIDER	DATA RECEPIANT
Signed: Susan Gellatly	Signed: Lawrence E Day
Print Name: Susan Gellatly	Print Name: Lawrence E. Day
Print Title: Director, Boeing IT	Print Title: Software Analyst

www.manaraa.com

Curriculum Vitae

LAWRENCE E. DAY, PMP, CSQA

44624 SE 159th Street
North Bend, WA 98045
(425) 865-1022 (Office) (425) 445-8938 (Cell)
Email: lawrence.e.day@boeing.com

Career Objective

Obtain a position that is both technically and administratively challenging and provides the opportunity to contribute to the economic and technical success of the enterprise.

Fourteen years of embedded software engineering development (all life cycle phases from algorithm simulation through development, test, competition fly-off, production and customer support). Sixteen years of exceptionally broad based software, systems engineering, and program management experience in a variety of strategic corporate disciplines. Ten years in IT computing and S/W development project management. Strong engineering/technical background with extensive program and project management experience. Demonstrated ability to manage large projects, budgets and schedules in cross-functional technical environment. Outstanding leadership, problem-solving, communication and organizational skills with a record of successfully motivating diverse teams to high efficiency levels.

Technical specialty areas include the following:

- Lean+ (VSM Coach, 6-Sigma Green Belt)
- S/W Engineering
- SEI CMM/CMMI
- Macroscope S/W Development Methodology
- Requirements Management
- Structured Analysis/Structured Design
- Engineering/Computing Project Mgmt
- Process Management, Design and Implementation
- Software Engineering
- S/W Inspections (Fagan, Gilb)
- Systems Engineering
- Metrics: Goal/Question/Metrics (GQM)
- Government S/W Development Methodologies (Military & FAA)
- Technical Sub-Contract Mgmt
- Object-Oriented Analysis & Design
- S/W Quality Assurance
- Radar Engineering
- Software and Documentation Configuration Management

www.manaraa.com

Education/Certification:

- ω Certified Project Management Professional (PMP 18442)
- ω Certified S/W Quality Analyst (CSQA 2050)
- ω ITIL Certified
- ω BS Electrical Engineer - 1969
- ω MBA (Technology/Engineering Management) - 1988
- ω Th.D – 2001
- ω Ph.D. in Information Systems Management - 2011

CAREER HISTORY & HIGHLIGHTS

THE BOEING COMPANY – PROGRAM AND SYSTEMS ENGINEERING MANAGEMENT

As INTERNAL SERVICES SYSTEMS (ISS) QUALITY AND CONTROL QUALITY AND METRICS ANALYST (2005 - Present) Boeing IT

Requested to lead the Boeing IT Internal Services Systems (ISS) Metrics Team to develop common metrics to track organizational improvement. Developed metrics analysis approach that was adopted by Boeing IT Quality and Effort & Schedule Variance Sub-Teams. The ISS sister organization of Finance Systems and the Vice-President level organization of Business Systems, which includes Finance and ISS, have implemented metrics analysis programs based on Lawrence's model. The Boeing SSG IT Partners monthly status meetings are also implementing a metrics dashboard based on Lawrence's analysis. As the Subject Matter Expert (SME) in Quality, Process Improvement, and Measurement/Metrics in the Engineering Operations and Technology Information Systems (EO&T IS) on Lawrence led them through the process of achieving a CMMI Lvl III appraisal. Lawrence is a certified Independent QA Audit auditor and has worked with the EPES tool. Lawrence was requested to be the Process Improvement Focal for the EO&T IS organization. He chaired weekly SEPG meetings that focused AD&S and metrics changes into the EO&T IS organization and acted as a communications vehicle to common process implementations.

Lawrence is the ISS rep to the Q&C Measurement Team and is the chairman of the Quality Sub-Team and the Agile Sub-Team as well as a member of the Effort and Schedule Variance Sub-Team. Lawrence is a Boeing certified Value Stream Mapping coach who led or coached thirteen VSMs and is a Six-Sigma Green Belt. A Paper on S/W Metrics was published in the QAI Journal for July 09 (Vol. 23, No. 3, p12-19).

As CUSTOMER ACCESS PROGRAM (CAP) SYSTEMS ENGINEER (2003-2005) Boeing IT

Requested to join the Customer Access Program (CAP) as the Request Broker manager. Completed a 6 month technical feasibility study that identified the Request Broker requirements for interfacing with the Boeing portal. Developed Technical Performance Measurements (TPMs) and Affordability criteria. Responsible for the

www.manaraa.com

development and implementation of the Requirements Management Plan and CAP Product Systems Engineering deliverables.

As IT e-COMMERCE PROJECT MANAGER (2000-2003) Shared Services Group (SSG)

Requested to join the Boeing Desktop Product Management group to develop and manage integrated workflow applications (web/database). This workflow application approach includes technical process development and implementation, s/w development, and database design. These projects affected all of Boeing's major business Groups and the workflow processes implemented included Technology Management Labs, Year 2000 User Commercial Off The Shelf (UCOTS), Windows 2000 Desktop Requirements Database, Windows 2000 Application Migration Status, Logistics & Support PC Process Repository, Computing Inventory Information Management, Leased Equipment Management System, Web Change Request Management, Windows 2000 OPS Server Deployment Status, Windows 2000 Enterprise Program, Enhanced POD (ePOD), and S/W License Renewal.

Demonstrated the leadership required to set up project teams, perform successful cross-functional coordination, manage requirements, implement S/W development methodologies and execute all project management disciplines to bring these projects to a successful conclusion, including sub-contract management.

As YEAR 2000 IT MANAGER (1998-2000) Shared Services Group

Requested to join the Boeing Year 2000 (Y2K) project and manage the User Commercial Off The Shelf (UCOTS) infrastructure effort for Y2k for all of Boeing. Designed and implemented the UCOTS process. Designed and developed an integrated workflow application (web/database) that implemented the agreed to process across the entire company. Set up project teams, staffed and executed various efforts to bring the Y2k effort to a successful conclusion. Effort spanned Boeing worldwide and included all areas of IS delivery systems from mainframes to desktops and everything in between. All the business Groups within the company used the UCOTS application, for Y2k activity, as well as to display Y2k product readiness status. Y2k was a top priority project in Boeing. Caught up from a seven month backlog and successfully completed the project in a year and a half by developing and implementing a web based UCOTS automatic process for collection and dispositioning of Y2k UCOTS product status requests. Task was completed using a group of three people, instead of nine, thus saving Boeing nine man-years of effort. Designated as the Boeing representative at the Y2k Platinum Consortium meetings.

As IT MANAGER (1995-1998) Information and Support Services

Selected to perform computing and S/W management of the Technology Services Distributed Integration Test Facility (DITF). Developed technical processes for managing the $22.7 million computing and S/W capital/expense plan for DITF that encompassed 20 different organizations. Developed the server layered software blockpoint methodology and implemented the process for Client/Server Blockpoints. Managed DITF lab,

www.manaraa.com

Client/Server operating system and layered software blockpoint, UNIX and X-Windows consultants, and electronic software distribution.

- Developed a Working Together Agreement with Compaq that saved, and continues to save, Boeing $1M a year.
- Provided infrastructure support for enterprise Tier I (desktop) and II (server) computing.
- Led development and implementation of processes to increase Client/Server blockpoint quality and dependability.
- Implemented computing process for equipment purchases, saving an average of 2 weeks flow time and 10 man-hours per approved Request for Equipment (RFE).
- Developed an integrated workflow application (web/database) for DITF Lab management and resource allocations.

Chairman of the Boeing Software Quality Assurance Council.

As S/W DEVELOPMENT MGR (1992-1995) Computer Services/Commercial Airplane

Selected to manage software development, cross-functional management and systems engineering for the 777 Automatic Test Equipment (ATE) software. Responsible for Object Oriented Analysis and Design (HP/UX platform) using C++; program and resource planning; product and task scheduling; software methodology development; and project tracking.

- Successfully built and delivered on schedule the manufacturing ATE carts for the first 777 aircraft.
- Managed the development and implementation of an OOA/D methodology.
- Implemented a progress and status management communication methodology that was developed by the rest of the program.
- Developed and managed the successful implementation of S/W Inspections in the S/W development process (see conference presentations above).
- Chaired the 777 ATE Continuous Quality Improvement (CQI) Steering Committee.
- Grew technical team from 5 to 24 in 3 months while developing successful project strategy and schedules.

As ENGINEERING MANAGER (1990-1992) Computer Services

Selected to manage approximately 40 software engineers and analysts as well as serving as Software Quality Manager. Responsible for allocating Software Engineering resources to various projects. Provided software expertise in real-time control systems and other manufacturing operations. Served as charter member of the Boeing Embedded Software Task Team (ESTT) and division spokesman for that team.

As SYSTEMS ENGINEERING MGR (1988-1990) Aerospace Company

Selected to manage 3 different Systems Engineering functions including Command/Telemetry, Power and Databases. Tracked and oversaw projects; provided both internal and external technical guidance, resource planning and cross-functional coordination.

www.manaraa.com

- Developed database management methodology and brought systems engineering database development under control.
- Developed coordinated database and delivery processes that crossed five organizational boundaries.
- Generated phased approach to product implementation of system delivery interrelationships that was adopted by the program.
- Performed an independent software audit of the Peace Shield program.

As PROGRAM MANAGER (1987-1988) Aerospace Company

Selected as a Program Manager to military and space contractor division with responsibility for all aspects of an Artificial Intelligence development program concerned with satellite autonomy in the "Star Wars" initiative. Managed staff of systems and S/W engineering groups with cross-functional management over Material, Finance, Human Resources and Contracts. Served as Software Development Manager for Phase I of a multi-phase government procurement.

- Developed satellites that could operate with few ground controller direct commands using elements of Artificial Intelligence.
- Used Structured Analysis/Design Methodology to manage System Requirements.

As TECH. SUBCONTRACT MGR (1985-1987) Aerospace Company

Selected to provide subcontract technical management of Software Engineering, Systems Engineering and Test groups totaling 140 personnel.

- Engineering and contract Manager for the $200M Westinghouse Electric Corporation (WEC) systems/hardware/software/documentation deliverables for Peace Shield Communication, Command and Control System.
- Developed standard communications processes and conducted successful Critical Design Reviews on hardware/software deliverables.
- Chosen as part of the Boeing/WEC team that did Manufacturing Availability Review (MAR) on a Westinghouse plant in Puerto Rico.
- As Software Engineering Manager, developed, and delivered to the Air Force, a 1500 page software requirements document.

As S/W MANAGER (1982-1985) Military Aircraft Company

Promoted into S/W Engineering Management. Responsible for the approval, and delivery to the customer, of the Westinghouse B-1B Radar Software (code and documentation). Responsible for computing engineering management of the B-1B S/W development labs and IS infrastructure.

- Brought Westinghouse software under control and got the software documentation off the Secretary of Defense's list of top three B-1B problems.
- Managed the B1-B S/W Development environment.

As RADAR & S/W ENGINEER (1969-1982) Boeing & Westinghouse

Served as radar engineer and radar software engineer, prior to promotion into Boeing management.

- Developed RADAR signal processing algorithmic simulations.

www.manaraa.com

- Developed RADAR signal processing and embedded operating system S/W.
- Supported deployment of Advanced Warning And Control System (AWACS) in demonstrations and to the customer.
- Generated and maintained AWACS radar manufacturing testing S/W that saved Boeing $11M.
- Generated a Value Engineering Change Proposal for AWACS radar S/W loads that saved the Air Force $1.15M.

www.manaraa.com

CONFERENCE PRESENTATIONS AND PAPERS

Orlando	Quality Assurance Institute Journal: "Software Metrics". (Vol. 23, No. 3, p12-19)	July 2009
Seattle	Boeing – Luxoft Technical Summit (BoLTS) Presentation: "A Design Manufacturing Approach to Software Development"	October 2008
Chicago	International IT Quality Assurance Conference One Day Tutorial: "S/W Verification Testing Metrics"	April 2008
Bilboa, SP	QA Test 2007 Tutorial: "S/W Verification Testing Metrics" (Doron Cherkovsky – Co-Author)	October 2007
London	Quick (Lean) Decision Making Presentation: "3D Earned Value: Lean Application of S/W Inspections for Quality Project Decisions"	July 2007
Orlando	QAI Testing Conference Tutorial: "S/W Verification Testing Metrics"	November 2006
Bilboa, SP	QA Test 2006 Tutorial: "Agile S/W Inspections"	October 2006
London	Value Driven Planning Conference "S/W Quality Value Prediction"	June 2006
Orlando	International IT Quality Assurance Conference One Day Tutorial: "Agile S/W Inspections"	April 2006
Tel Aviv	SELA University Multiple Courses: "Agile S/W Inspections", "Principles of Successful Software Projects", "S/W Metrics" SIGiST Keynote Address: "Using Verification Methods to Improve Product Development"	April 2006
Düsseldorf	6th ICSTEST International Conference on Software Testing One Day Tutorial: "Principles of Successful Software Projects"	April 2005
Spokane, WA	Spokane Rotary Keynote Speaker: "World Class Quality and the Culture of Blame"	February 2005
Seattle	Seattle Area S/W Quality Assurance Group Presentation: "Use of Inspections As A Risk Management Tool"	January 2005
Bilboa, SP	ICSTEST-E Keynote Speaker: "Successful Embedded Software Verification and Testing: A Case Study"	November 2004
Orlando	International S/W Testing Conference Keynote Speaker: "Using Verification Methods to Improve Product Development" One Day Tutorial: "Principles of Successful Software Projects"	October 2004
Madrid	8th International Software Re-Use Conference Presentation: "Requirements Re-Use in the IT Business Process WEB Implementation Product Family"	July 2004
London	Evolutionary Deployment (EVO) Engineering Conference	June 2004

www.manaraa.com

Presentation: "EVO e-Implementation of Organizational Business Processes"

Orlando International Conference on Effective Methods for IT Quality May 2004
One Day Tutorial: "Principles of Successful Software Projects"

Krakow, PL East European Conference on Systems and Software April 2004
Keynote Speaker: "World Class Quality and the Culture of Blame"

Düsseldorf 5th ICSTEST International Conference on Software Testing April 2004
Keynote Speaker: "The Boeing-Approach to (Independent) Software Verification and Validation"

Amsterdam EuroSTAR 2003 December 2003
Presentation: "Use of Inspections As A Risk Management Tool"

Tuloca, México El Instituto Tecnológico y de Estudios Superiores de Monterrey October 2003
Presentation: "World Class Quality and the Culture of Blame"

Minneapolis PSQT/PSTT North September 2003
Presentation: "Use of Inspections for Product and Process Improvement"

London Systems Architecture Engineering Conference June 2003
Presentation: "Data-Driven Workflow Applications Architecture"

London Co-Teach Testing Master Class June 2003

Orlando International Conference on Effective Methods for IT Quality May 2003
Double Session Presentation: "Process Quality Assurance Using the AUTOmatic PROcess DrivenTask Execution And Management (AUTOPROD TEAM) Approach"

Cologne 4th ICSTEST International Conference on Software Testing April 2003
Presentation: "Use of Inspections As A Risk Management Tool"

London Co-Teach Testing Master Class March 2003

London Systems/Requirements Engineering Conference June 2002
Presentation: "S/W and Systems Requirements Management Using Inspections"

Orlando International Conference on Effective Methods for IT Quality April 2002
One Day Tutorial: "Process Quality Assurance Using the AUTOmatic PROcess Driven Task Execution And Management (AUTOPROD TEAM) Approach"

Düsseldorf 3rd ICSTEST International Conference on Software Testing April 2002
Presentation: "Use of Inspections As A Risk Management Tool"

Seattle Seattle Area S/W Quality Assurance Group Sept. 2001
Presentation: "AUTOmatic PROcess Driven Task Execution And Management (AUTOPROD TEAM)"

London Competitive Systems and S/W Engineering Conference June 2001
Presentation: "Process Implementation Using A Competitive and Quality S/W Engineering Approach"

Seattle IT E-Commerce Applications Conference June 2001
Presentation: "Process Implementation Via the Web Ensures Quality and Data Integrity"

Orlando International Information Technology Quality Conference April 2001

www.manaraa.com

	Presentation: "Process Embedding in Web Applications – Ensuring Quality and Data Integrity"	
Kansas City	E-Commerce Applications Conference	June 2000
	Presentation: "Successful Intranet Usage for Technical Transactions"	
Seattle	Quarterly Y2k Platinum Meeting	July 1998
	Presentation: "Y2k Supplier Product Readiness Management – Year 2000 Ready"	

www.manaraa.com

ANCILLARY BUSINESS EXPERIENCE

As Aircraft Leasing Business Owner and Flight Instructor (1980 to Present)

Past owner of an aircraft leasing business. Give flight instruction. Past President and Treasurer of the Boeing Employees Flying Association (BEFA) with 400 members, approx. 18 aircraft, $1M budget and $750K balance sheet. As a Board member of a highly government (federal, state and local) regulated non-profit organization with $1m+ budget and a 400+ membership I developed strategic business plans, implemented yearly budgets, interfaced with government regulatory personnel, committed to business and financial contracts, managed staff and provided organizational leadership.

Current FAA ratings include:

Pilot Certificate:

- Commercial/Instrument
- Single engine Land and Sea
- Multi-engine Land

Flight Instructor Certificate:

- Single Engine
- Multi-Engine
- Instrument

Ground Instructor Certificate:

- Advanced
- Instrument

As Board Member of the Boeing Employees Flying Association – BEFA

ω President 1987-1988

ω Treasurer 1999-2005

www.manaraa.com