A DISCRIMINATION OF SOFTWARE IMPLEMENTATION SUCCESS CRITERIA

Alan N. Pryor, B.S., M.B.A.

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSPHY

UNIVERSITY OF NORTH TEXAS

August 1999

APPROVED:

Carl Steven Guynes, Major Professor
Perry McNeill, Minor Professor
John C. Windsor, Committee Member
Robert J. Pavur, Committee Member
Victor R. Prybutok, Coordinator of the Doctoral Program
John C. Windsor, Chair of the Department of Business Computer Information Systems
Richard E. White, Associate Dean and Doctoral Director of the College of Business
C. Neal Tate, Dean of the Robert B. Toulouse School of Graduate Studies

Pryor, Alan N., A Discrimination of Software Implementation Success Criteria. Doctor of Philosophy (Business Computer Information Systems), August 1999, 155 pp., 11 tables, 3 illustrations, 67 references.

Software implementation projects struggle with the delicate balance of low cost, on-time delivery and quality. The methodologies and processes used to create and maintain a quality software system are expensive to deploy and result in long development cycle-time. However, without their deployment into the software implementation life-cycle, a software system will be undependable, unsuccessful.

The purpose of this research is to identify a succinct set of software implementation success criteria and assess the key independent constructs, activities, carried out to ensure a successful implementation project. The research will assess the success of a software implementation project as the dependent construct of interest and use the software process model (methodology) as the independent construct.

This field research involved three phases: (1) criteria development, (2) data collection, and (3) testing of hypotheses and discriminant analysis. The first phase resulted in the development of the measurement instruments for the independent and dependent constructs. The measurement instrument for the independent construct was representative of the criteria from highly regarded software implementation process models and methodologies, e.g., ISO9000, Software Engineering Institute's Capability Maturity Model (SEI CMM). The dependent construct was developed from the categories and criteria from the Delone and McLean (1992) MIS List of Success Measures. The data collection and assessment phase employed a field survey research

strategy to 80 companies involved in internal software implementation. Both successful and unsuccessful software implementation projects (identified by the Delone/McLean model) participated. Results from 165 projects were collected, 28 unsuccessful and 137 successful. The third phase used ANOVA to test the first 11 hypotheses and employed discriminant analysis for the 12th hypothesis to identify the "best set" of variables, criteria, that discriminate between successful and unsuccessful software implementation projects.

Twelve discriminating variables out of 67 were identified and supported as significant discriminators between successful and unsuccessful projects. Three of the 11 constructs were found not to be significant investments for the successful projects.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

Adam Smith (1937) theorized that low cost was the key to economic wealth. Michael Porter (1990) pledged that obtaining a competitive advantage was the primary component of achieving a successful business. Over two decades ago, many deemed quality management (continuous improvement) as the key to the success of a business (Crosby 1979). Quality management, which later evolved into Total Quality Management (TQM), has continued to be an area of attention for business success since its inception. The most recent quality improvement technique to gain popularity is Business Process Reenginnering (BPR). BPR is a quality initiative, which essentially involves starting from scratch and rebuilding the business around its "true" processes. In theory, BPR results in significant change versus continuous incremental change obtained from TQM (Hammer and Champy 1993). These examples represent only a small portion of the literature providing suggestions to business success. Many of the more recent publications, however, agree that the investment in quality, if not the most important initiative, is at a minimum imperative to business success.

Corporations confront the tradeoff between the maximum investment in quality and the minimum cost investment to obtain optimal profit. The Malcolm Baldrige award is the most prestigious quality award given to an American company each year. Several

companies that have won the Malcolm Baldrige award have gone out of business due to financial problems. Post interviews with their CEOs indicate that their emphasis on quality programs was too costly. While these initiatives improved their internal processes, products and services, the cost to implement them resulted in major profit losses, which in-turn resulted in a failed business (Shetty 1993 and Laser 1994).

The dilemma that businesses face is identifying the necessary investments of quality initiatives and activities that impact the success of a business. For example, firms are inundated with financial ratios upon which they based their investment decisions. They continually question which financial ratios are the best indicators of success. A discriminant analysis was conducted on the financial ratios used by companies for a set of homogeneous successful and unsuccessful firms. The discrimination results reduced this massive list of financial ratios to a select few. This study provided evidence that the new select set of ratios were the "true" indicators of the success or failure of a firm (Sharma 1996). The intent of this research is to apply the technique of discrimination of the ratios that indicated the success of financial firms to that of software implementation criteria in determining the success of a software project. Software implementation projects currently have a massive list of tasks they are required to employ to ensure their projects meet quality standards and are successful. The primary research question to address this dilemma is as follows. What tasks contribute to the success or failure of a software implementation project? The primary software implementation project success constructs will be identified and tested.

Purpose of the Research

The measure of quality software has been emphasized by organizations since the early years of information systems implementation (Stevens, Myers, and Constantine 1974). A visit to an organization's software shop will result in displays of bulletin boards posted full of colorful, up-to-date software metrics. The majority of software process improvement metrics are reactive, after the fact results, (e.g., defects per lines of code, cycle time to develop and install a system, system up-time/availability, etc.). Companies need predictive metrics if they expect to make significant improvements (Ross 1990).

Companies, especially large established corporations, have a defined and documented process describing the steps to produce Information Systems (IS) solutions. Even smaller businesses have some form of documented plan to pursue an IS solution. The widespread deployment of Management Information Systems (MIS) in the 1980's led to the realization that high IS support costs and defective IS products were the result of not following a defined process (Boehm and Papaccio 1988). The problem is identifying the correct methodology for each business's situation and optimally defining and deploying the process within each software implementation activity. Many of the steps are ambiguous or deemed unnecessary by the project team. The steps deemed necessary can often be difficult to deploy due to the dynamic business environment and resistance by the diverse team member personalities. Resistance from the project team members and the amount of time required to perform each activity properly, often results in non-synergistic project management goals. This resistance leads to projects that exceed the forecasted budgets and miss scheduled release dates.

Within the past decade Information Systems (IS) shops have begun to emphasize the overall measure of their software process activities. These attempts resulted in documenting methodologies to guide and direct the software processes and embed quality in the software work products. Even with this rigid process guidance, IS shops still suffer from lack of sound proactive measures of the abilities to conceptualize, create, and support software solutions for business needs (Humphrey 1996). What is the most appropriate measure of software quality? What components (set of criteria) measure the "true" success of a software implementation project? The conflicting objective that projects face is that of payback on their investment of software quality. What minimum investments are necessary to achieve maximum quality? To define and deploy a set of proactive measures of software quality requires significant investment. Project managers will agree that this investment at times may exceed the IS solution's financial payback (Jones 1996).

Many of the organizations that prioritize investment in software quality may use an external set of criteria to guide and measure their progress. Some of the highly regarded software improvement methodologies (models and/or measures) presently in use are:

- ISO 9000 (International Organization for Standardization)
- Malcolm Baldrige Criteria, Category 2
- Nolan, Norton & Company
- Real Decisions (Gartner Group)
- R.S. Pressman and Associates Inc.

- Software Productivity Research (Capers Jones)
- Software Engineering Institute's (SEI) Capability Maturity Model (CMM)
- Software Engineering Laboratory (SEL)

To highlight a few of these, ISO 9000 is the most widespread worldwide set of quality standards to date. ISO 9000 contains many classifications, each with its own emphasis. For example: ISO 9000-1 is a set of guidelines for selection and use of the ISO standards; ISO 9001, 2, 3 are standards used for certification; ISO 9004, 9004-2, 9004-3 are guidelines. The SEI CMM gained its momentum by use in the United States government contracted operations. It is the premiere model today, but it is difficult to tailor and very costly to deploy. The Malcolm Baldrige Category 2 criteria emphasizes the identification of decision making data and analysis of information and are often used as a reasonable measure of the overall quality of an organization's software process. The other models are narrower in scope and, in some cases, more manageable. However, for the most part they only scratch the surface relative to the comprehensiveness of ISO 9000 and the SEI CMM (Saiedian and McClanahan 1996).

The highly regarded models are not mutually exclusive and in most cases contain subsets of similar guidelines for software process improvement. Choosing the correct process model depends on many factors such as organization size, available funding, scope of project, type of customers, etc. There is no perfect model for a given software implementation project or business. There are many common activities among these models. These similarities serve as the basis for analysis within this study. The ISO 9000 and SEI CMM share many common concerns regarding software quality and process

management. They are also the most comprehensive and contain almost every aspect of the other models. Thus, the independent variable set is primarily from these two models.

The Problem Motivating This Study

Each of the mentioned software models has its own set of strengths and weaknesses. Each software solution varies in scope and investment, and thus, so should the methodology and process to guide it. Organizations face several issues in the areas of software work product measurements, methodologies, and processes. Some of these concerns are below.

- What is the best measure of software for each specific situation?
- What criteria contribute to the success of a project?
- How can one avoid unnecessary, but management enforced criteria and measures?
- How can the process be tailored to be as dynamic in “ease-of-use” as the software solution’s scope and investment?
- What process model will result in optimal payback, given a solution’s investment and life expectancy?

The problem motivating this study is not a result of just one of the specific concerns mentioned, but the aggregate impact of all problems resulting from the subjectivity in determining how to deploy successfully software quality initiatives and tasks. The intent of this research is to identify the software implementation criteria that determine the success or failure of a software implementation project.

Significance of Problem

The dilemma faced by software implementation projects is identifying the necessary investments in quality initiatives and activities that impact success. The results of this research will provide evidence as to what are the "true" success indicators for investment in a software implementation project.

The significance of these findings will have a propitious impact on the software implementation community, in that the creation and deployment of project methodologies are costly and contain steps that do not impact the success or failure of the project. The purpose of this research is to identify a concise, select set of "true" success measures for a software implementation project that may easily be tailored to fit the project scope, duration, and life-expectancy.

General Limitations

Limitations are an inherent part of any research. Generalizing the results depends upon the willing participation of organizations and personnel in a variety of businesses and with varying size and scope of operations. A random sample of subjects from the universe of companies performing software implementations is not feasible.

As with any field research, only partial control is possible and there is limited ability to accommodate extraneous variables (Buckley, Buckley, and Chiang 1976). Also, participants have different levels of expertise and familiarity with the research topic.

Key Terms and Definitions

Information System

Information System (IS) refers to a physical process that supports an organizational system by providing information to achieve organizational goals (Turban, McLean, and Wetherbe 1996).

Information Technology

Information Technology (IT) refers to the technological side of an information system, including hardware, databases, software networks, and other devices and can be viewed as a subsystem of an information system (Turban, McLean, and Wetherbe 1996).

Software Implementation

Software Implementation refers to the development of instructional coding that manipulates the hardware in a computer, the deployment of third party instructional coding that manipulates the hardware in a computer system, or a combination of the two.

Software Process or Methodology

Software process or methodology refers to the documented process of activities and proactive measures of the abilities to conceptualize, create, deploy, and support software solutions for business needs (Humphrey 1996), or the activities and associated information that are required to develop a software system (Sommerville 1996).

Software Work Product

Software work product defines any component used in the software implementation life cycle (e.g., requirement specification, design specification, signoff lists, source code).

Organization of the Paper

This paper is organized into six chapters. Chapter 1 presents the purpose, problem, significance, general limitations, and key terms pertaining to this research. Chapter 2 includes a summary of the literature and prior research. Chapter 3 presents the theoretical framework and model on which the research is based and describes the research methodology. Chapter 4 presents the results of the data analysis and Chapter 5 discusses these results. Chapter 6 summarizes this research effort and offers suggestions for further research.

CHAPTER 2

PRIOR RESEARCH

Introduction

A search of previous literature on this topic may be categorized into two areas: first, the process used to implement Information Technology (IT) solutions, and second, the measure of the success of an IT solution once installed. The process is often referred to as systems analysis and design or the lifecycle of an Information System (IS). Its lifecycle spans from identification of the problem (conception) through system installation and support of that system once fully deployed. The documented process to implement IT solutions is referred to as the methodology or model to be followed. The second area, the measure of IS success, has received much analysis and has been categorized into the areas of system quality, information quality, use, user satisfaction, individual impact, and organizational impact (Delone and McLean 1992). These categories are not mutually exclusive, but provide an organized approach to understanding the success of a system. The scope of this research will use the term and category of system quality to represent the success of an IS.

Systems Analysis and Design Evolution

When computer systems evolved little emphasis or thought was given to structure or a process model to provide guidance on the appropriate steps and activities at a given

phase from concept to delivery (Stevens, Myers and Constantine 1974). It did not take long for system support staffs to verify that high support costs were a result of ad hoc requirements gathering and unstructured, undocumented code. Publications to control cost all pointed to the need for a defined, repeatable process (Boehm and Papaccio 1988). Watts Humphrey stated in 1995, "software is now a critical element in many businesses, but all too often the work is late, over budget, or of poor quality" (Myers 1995). He, among others, preach that the imperative to a successful software system is a well designed, deployed software process. The term "software process" defined by Sommerville (1996, p. 269) represents "the activities and associated information that are required to develop a software system." Humphrey (1996, p. 77) defines software process as "a sequence of steps required to develop or maintain software."

As early as 1969 the realization of the need for structure in coding began to take shape. Bachman (1969) introduced data structure diagrams and emphasized the need for a standard data management language. The first published software process life-cycle model was by Royce (1970). To this date, many organizations' models are based on this process life-cycle model. Four years after the Royce model was introduced, 1974, Stevens, Myers and Constantine proposed that structured design is a set of proposed general program design considerations and techniques to make coding, debugging, and modification easier, faster, and less expensive by reducing complexity. There have been several evolutions of the Royce model. The most popular is the waterfall model (which many give Royce credit for developing, even though he did not use that term), which includes for any given project the following phases each done in completion before

proceeding to the next. The first is documenting the requirements specification, second involves the system design and implementation, the third step consists of system integration and testing, and the fourth is system operation, support, and maintenance. The only real problem with the waterfall model is the lack of feedback from one stage to another (Sommerville 1996). In an attempt to couple evolutionary development and management processes more closely Barry Boehm (1988) proposed a spiral model. This methodology categorizes the various steps in the software life-cycle and spirals outward from the requirement specification to other process steps. This spiral technique allows a risk assessment and requirements verification to be performed at each phase of progression. It also accommodates the development of different parts of a delivered system using various approaches. The most appropriate process model, for a given project, depends on many factors, including organizational environment, scope and duration of the project, customer expectations, and capabilities of the personnel and resources. In an attempt to better address dynamic requirements, Davis, Bershoff, and Comer (1988) described a paradigm to compare and contrast each of the alternative life cycle models with the more conventional waterfall model in the face of constantly evolving user needs. Their work was the first to address the constantly changing requirements that inherently exist in the business world.

The search continues for an easy to use, trustworthy model. According to Strehlo (June 24, 1996), Raytheon, a large defense contractor, has dedicated personnel who hunt for and shake down new, easier to use software process improvement tools. Competition continues to put pressure on the software industry and encourages organizations to

examine the effectiveness of their software development and evolution processes. Many are now attempting to establish performance baselines, setting improvement goals, explicitly defining the normally implicit processes and measuring progress toward their improvement goals (Krasner et al. 1992). Organizations that invest in the Software Process Improvement (SPI) journey are "looking for a magic route: a single one-size-fits-all strategy that guarantees higher process-maturity levels. In reality, dozens of possible strategies will get the organization to its goal" (Pressman 1996). Pressman's first point of the hunt for the magic solution is substantial. In fact, that is the quest of this research. However, his second point - that dozens of strategies will get them there - demands further elaboration. Though dozens of strategies are available, the difficulty lies in finding which one is right for which situation and selecting the components from it that are easy to deploy and will result in positive results at a minimal investment. The SEI is working on this and has recently introduced its new Personal Software Process (PSP), which is a framework of techniques to help engineers improve their performance, through a step-by-step, disciplined approach to measuring and analyzing their work (Humphrey 1996). Though this new approach is an excellent training technique to embed better structure and repeatability into designs, it does not address the need for a tailorable, concise, dependable software process model. Just as the waterfall model approach, discussed earlier, was becoming fully elaborated, people were finding that its milestones did not fit an increasing number of project situations. In order to address this issue, Boehm (1996) was convinced he discovered the most consistent correlates of success versus failure. The success or failure of a project is dependent on the degree to which the

project employed its life-cycle objectives, life-cycle architecture, and initial operational capability. He admits that the weakness to the 1988 proposed model is the difficulty in applying the model due to lack of explicit process guidance in determining objectives, constraints, and alternatives. He proposes a win-win spiral model by process model extensions in the areas of determining objectives, determining constraints, identifying and evaluating alternatives, recording commitments, and cycling through the spiral.

Systems Analysis and Design Theory

In the early era of programming computers the need to represent the flow of activities and variables was necessitated in order to design structured (properly arranged) code. Two initial methods for assisting with this were flow charts and data structure diagrams. These simple methods have now evolved into complex data modeling techniques, which will be discussed later in more detail. Charles Bachman (1969) defined the two basic elements in data structure diagrams. He proposed two elements for data structure diagrams, blocks and arrows, and their use. A block should be used to represent entity classes (e.g., employee, department), and arrows are to be used to represent set classes and to designate the roles of the owner and/or member established by that set class. With this scheme structures can be modeled either as a hierarchy, a network, or a tree. This early piece laid the foundation for flowcharting and today's data modeling techniques of pictorially representing business activities.

As the mainframe computer continued to grow in use and size, more sophistication was needed to ensure the deployment of dependable systems. Stevens,

Myers, and Constantine (1974) emphasized this by proposing a structured design to the systems development process. They defined structured design as "a set of proposed general program design considerations and techniques for making coding, debugging, and modification easier, faster, and less expensive by reducing complexity" (Stevens et al. 1974, p. 115). They found that simplicity is the primary measurement recommended for evaluating alternative designs relative to reduced debugging and modification time. They also provided some definitions of key IS terms, which are still in use today. For example, a module is a set of one or more contiguous program statements having a name by which other parts of the system can invoke it and preferably having its own distinct set of variables.

Within their research on structured design, Stevens, Myers, and Constantine (1974) explored the topic of system complexity and discussed in-depth the role of coupling and cohesion. Understanding coupling and cohesion is an important part of designing a sound software process. These terms will be briefly explained. Stevens et al. found that minimizing connections between modules also minimizes the paths along which changes and errors can propagate into other parts of the system, thus resulting in disastrous "ripple" effects. These "ripple" effects also occur where changes in one part cause errors in another, necessitating additional changes elsewhere, giving rise to new errors, etc. When two or more modules interface with the same area of storage, data region, or device, they share a common environment and become more cohesive. Coupling is a measure of the strength of association established by a connection from one module to another. A key component making up the overall complexity of a system is

the amount of coupling between modules. Coupling also increases with the obscurity of the interface. Coupling is reduced when the relationships among elements not in the same module are minimized. In regard to coupling and cohesive systems, the objective is to reduce coupling by striving for high binding. Cohesiveness is a scale from low to high (it is not linear). The scale of cohesion from lowest to highest is coincidental, logical, temporal, communicational, sequential, and functional. Coincidental cohesiveness might result from splitting an existing program into parts, or duplicate coding. Logical binding implies some logical relationship between elements of a module. Temporal binding is the same as logical, except elements are also related in time. Communicational cohesiveness has elements related by a reference to the same set of input and/or output data. Sequential cohesiveness results from flowcharting the problem to be solved, and then defining modules to represent one or more blocks in the flow chart. And in the strongest type, functional cohesiveness, all of the elements are related to the performance of a single function. Predictable or well–behaved modules, when given the identical inputs, operate identically each time they are called.

Stevens, Myers, and Constantine (1974) also built on IBM's HIPO (Hierarchy-Input-Process-Output) chart, which shows all functions in separate blocks, and on other structured programming coding techniques. They evaluated alternatives for portions of the system that will be programmed on a computer and found that structured design reduces the effort needed to fix and modify programs. The tradeoffs to structured design involve the overhead in execution time and memory space used by a particular language to affect the call. Structured design techniques divide the design process into general

program design (what functions are needed) and detailed design (how to implement the functions). They provided a preliminary set of design guidelines to meet these goals. These guidelines require matching the program to the problem, documenting the scope of effect and control, identifying the module size, and preparing for errors, reaching the end of the file, program initialization, selecting modules, isolating specifications, and reducing parameters. Though Stevens, Myers, and Constantine proposed these ideas in 1974, they still serve as the foundation for many systems development processes in use today. Structured design is becoming increasingly important to the data-processing industry to be able to produce more programming systems with fewer errors, at a faster rate, and in a way that additional modifications can be accomplished easily and quickly.

There have been several proposals over the past 20 years to apply outside (non-traditional IS) techniques to the software process. Hebert Simon (1986) proposed that Artificial Intelligence (AI) might augment the development of software engineering tools, resulting in faster developed and lower cost software than when created by humans. Software engineering is a labor-intensive process. Development of applications software tends to be a lengthy and costly undertaking. Simon conceives of artificial intelligence as "something that one might try to evoke in any situation where a task is to be done that requires some kind of mind, or intelligence" (Simon 1986, p. 727). When strong methods or algorithms are not sufficient to solve problems in a given domain, researchers use weak methods, when heuristics should be applied. Heuristics are rules of thumb that are generally obtained by observing humans solving problems. Simon states his view of how progress should be made toward automating the software engineering process. Progress

should be made toward developing a formal specification language for the software engineering process. Simon believes that natural language interfaces should be another area of research. Databases of data representations as well as databases of software development strategies are needed to automate this type of design task. Artificial intelligence takes an unstructured problem and turns it into a formalized situation to which algorithms can apply. In these unstructured situations, humans use weak methods or fallback procedures. Most of the methods involve a heuristic search, search with rules of thumb used, in a problem space; set of possibilities, situations, or partial problems, which might be generated in the course of a heuristic search. The primary concern is whether artificial intelligence is powerful enough for software engineering. Software development must be structured from the top down. Organization and order must be in the human part of the task. And an expert system must be designed in interactive mode for progressive modification.

Several other techniques directed toward providing structure to the software process have been adopted from other disciplines. One example of this is prototyping. Prototyping has been primarily an engineering preliminary modeling technique that has resulted in more easily manufactured designs. Janson and Smith (1985) assessed the use of prototyping in systems development. They found that different types of prototypes are used for a variety of purposes and integrated into the various stages of the systems development life cycle. They found many advantages to incorporating prototypes into the software process, such as uncovering bugs earlier, and they also provided guidance on how to avoid misapplications of prototyping in IS development. Prototyping was best

applied as a method for systems design where users have difficulty in specifying, or are unable to specify their information needs. By seeing a model the users were able to express their requirements more accurately. Janson and Smith (1985) categorized prototypes into three groups. Real-life prototypes are a full scale representation of the basic design idea employing material intended to be used for the final design. These prototypes can often be used in the final product. Simulated prototypes are similar, but use a different medium for construction from the final intended design. The third type is a combination of the other two, real-life/simulated prototype, where some portion of the prototype may be used in the final product. They presented a view of prototyping based on the analysis of engineering design processes similar to those used in the development of IS.

Additional research in the area of applying prototyping to the IS process was done by Alavi and Wetherbe (1991). They set out to determine if combining two techniques available to systems designers would help in the development of the right system. They looked at adding data modeling as a preliminary step to prototyping to assess if this gave prototyping more structure and made the process more efficient. They addressed the dependent variables of task satisfaction, process satisfaction, perceived self-determination, stress and task complexity. They found that information systems development has two simple objectives. First, make sure the system is right for its purpose. And second, make sure it works correctly. Their results demonstrated that data modeling is a useful preliminary step to prototyping. Even though the subjects were not happy with the development process and found it to be more complex than their previous

techniques, they developed a system that more accurately met the specifications in less time. Prototyping facilitates creating the requirements specification while data modeling enhances data structuring and efficiency. By combining the two techniques a better system will result.

The classic waterfall model, defined by Royce (1970) and later refined by Boehm (1976) has been the primary basis for software development since its introduction. Davis, Bersoff, and Comer (1988) compared and contrasted the conventional waterfall model with four alternative life cycle models (i.e., rapid throwaway prototyping approach, incremental development approach, evolutionary prototyping approach, and the automated software synthesis approach). This comparison was performed relative to evolving user needs (i.e., dynamic requirements result in aiming at a moving target). The steps in the classical waterfall model include system requirements, software requirements, preliminary design, detailed design, code and debug, test and pre-operations, and operations and maintenance.

A summary of the four alternative methods is provided. The rapid throwaway prototyping approach addresses the issue of ensuring that the software product being proposed really meets the users' needs. To do this, a partial implementation of the system is constructed prior to the requirements stage. Incremental development is the process of constructing a partial implementation of a total system and slowly adding increased functionality or performance. Evolutionary prototyping is where the developers construct a partial implementation of the system, which meets known requirements. The prototype is employed by its intended users so that the full requirements may be better understood.

Automated software synthesis is a term used to describe the transformation of requirements or high-level design specifications into operational code.

For each of these methods categories were assigned to assess various aspects of the models. The categories are shortfall, lateness, adaptability, longevity, and inappropriateness. Shortfall defines the difference between the actual requirements and the system, and measures how far the system is from meeting the actual requirements at any given time. Lateness represents the time delay associated with achievement of a level of functionality and measures the rate at which the software solution can adapt to new requirements. Adaptability is the rate at which the software solution can adapt to new requirements and measures the time that elapses between the statement of a new requirement and its satisfaction. Longevity is the time a system solution is adaptable to change and remains viable and measures the time a solution is adaptable to change and remains viable. And finally, inappropriateness represents the behavior of shortfall over time and measures the difference between user's needs and what the solution provided. The standard model for software development is the waterfall method (which is also known under various other names).

As a result of the assessment by category, all five of the approaches reduce the time between the user's needs and the implementation of a functional system when compared to the traditional method. Therefore, alternate life cycle models improve product development. They also can be used to improve the traditional model by improvements to each step, such as consideration of requirements changes at various steps in development. Costs can also be compared and measured between methods.

Productivity can be measured as functionality provided by hour of labor. The ultimate goal is to reduce the backlog of unanswered requests for projects. These alternatives may provide helpful measures for increasing the development output and improving the process (Davis, Bershoff and Comer 1988).

One of the most recognized pieces in the area of the software lifecycle is by Barry Boehm (1988). This publication describes a model for software development that is based on a spiral. A software process model is presented with the intent of improving the software process model situation. He proposes that the primary functions of the software process should be to determine the order of the stages involved in software development, their evolution, and to establish the transition criteria for progressing from one stage to the next. Software process models provide guidance on the order, phases, increments, prototypes, validation tasks, etc., in which a project should carry out its major tasks. A software methodology's primary focus is on how to navigate through each phase, determining data, control, or "uses" hierarchies; partitioning functions; allocating requirements, and how to represent phase products, structure charts; stimulus-response threads; state transition diagrams. The primary advantage of the spiral model is that its range of options accommodates the good features of software process models currently in use, while its risk-driven approach avoids many of their difficulties. The three problems with using the spiral model involve matching it to contract software, relying on risk-assessment expertise, and the need for further elaboration of spiral model steps. Boehm also provided an assessment of other process models. Table 1 summarizes his findings.

Table 1. Barry Boehm's 1988 Assessment of Software Process Models

Model	Technique	Problems
Code-and-fix Model	Write some code fix the problems in the code	poorly structured code poor match to users' needs expensive to fix with each iteration
Stagewise Model	Software is developed in successive stages: Operational Plan, Operational specifications, coding specifications, coding, parameter testing, assembly testing, shakedown, system evolution.	
Waterfall Model	Staged development like stagewise model feedback loops to successive stages initially incorporates prototyping in the software life cycle	Emphasis on fully elaborated documents as completion criteria for early requirements and design phases. some stages are pursued in the wrong order
Evolutionary Development Model	Stages consist of expanding increments of an operational software product matched to 4GL	Is difficult to distinguish it from the old code-and-fix model, with spaghetti code and lack of planning based on the assumption that the user's operational system will be flexible enough to accommodate unplanned evolution paths.
Transform Model	A formal specification automatic transformation of the specification into code an iterative loop exercise of the resulting product outer iterative loop to adjust the specifications	automatic transformation capabilities are only available for small products in a few limited areas the assumption that user's operational systems will be flexible knowledge-base-maintenance problem in dealing with reusable software components
The Spiral Model	Determine objectives for performance, functionality, ability to accommodate change, etc. Evaluate alternative means of implementing (prototyping, simulation, benchmarking, reference checking, administering user questionnaires, analytic modeling, etc.) Determine the constraints (cost, schedule, interface, etc.) Determine risks Risk resolution Risk resolution results Each cycle is completed by a review involving the primary people. Review products and develop future plans.	it is, as yet, difficult to accommodate contract software it relies on risk-assessment expertise (only as good as people who use it). it is still an immature system and needs elaboration of the spiral model steps

The spiral model was developed under the direction of Barry Boehm, by TRW Defense Systems Group, in order to initiate and complete a task to improve their software productivity. The spiral system addresses many weaknesses of previous models. Boehm (1988) discusses eight advantages of the spiral system. First, the range of options accommodates the good features of existing software process models while its risk-driven approach avoids many of their difficulties. Next, it focuses early attention on options involving the reuse of existing software. Third, it accommodates preparation for life cycle evolution, growth, and changes of the software product. Fourth, it provides a mechanism for incorporating software quality objectives into software product development. Fifth, it focuses on eliminating errors and eliminating unattractive alternatives early in the process. Sixth, for each of the sources of project activity and resource expenditure, it answers the key question, "How much is enough?" Seventh, it does not involve separate approaches for software development and software enhancement or maintenance. And finally, it provides a viable framework for integrated hardware-software system development.

Boehm and Papaccio (1988) addressed the worldwide growth of software costs reported at $140 billion in 1985. They stated that because software costs are growing and many candidate software projects were not developed due to lack of funds, "understanding and controlling software costs can get us better software, not just more software" (Boehm and Papaccio 1988, p. 1462). They state that even though frameworks for controlling software budgets, schedules, and work completed have been published, they are not enough for controlling software costs. They propose two methods to analyze

software costs, “black box” or influence approach and the “glass box” or cost distribution approach. The "black box" or influence-function approach compares the results of many software projects to determine the effects of certain characteristics like methodology, personnel experience, etc. The other method, the "glass box" or cost-distribution approach, analyzes one or more software projects to characterize internal distribution of costs among such sources as labor versus capital costs, code versus documentation costs, development versus maintenance costs, or their distributions of cost by phase or activity.

Their proposal is summarized in a "Software Productivity Improvement Opportunity Tree," which is composed of six branches as follows, make people more effective, make steps more efficient, eliminate unnecessary steps, eliminate rework, build simpler products, and reuse components. They propose two means of implementing software construction projects: management by objectives, which compares the actual performance to the planned performance, and a risk driven approach of optimizing software performance around software predictability and control.

Kemerer and Porter (1992) applied the measures and reporting of effectiveness and efficiency of internal operations to the software process. They applied the degree of reliability of function points (FPs) as a software metric. Software development and maintenance encompasses two major functions, planning and control. FPs size a system in terms of its delivered system components, measured as a weighted sum of the number of inputs, outputs, inquiries, and files. FPs are reliable and may indicate a wider acceptance as software metrics. The results of this analysis provide guidance to FPs as standard setting bodies in their deliberations upon rule clarification. Also, guidance is

given to practitioners as to where the difficulties lie in the current implementation of FPs. They proposed that the result of this effort should continue the process of improving the quality and reliability of the measure of software size, productivity, and quality.

Baxter (1992) laid a foundation for design maintenance systems (DMS). A DMS requires a representation for programs (that the software system be formally specified), a transformation engine, an agenda-oriented meta-programming language, a representation for the justification (desired maintenance deltas may be formally specified), justification revision mechanisms (delta integration procedures). Software construction models address areas where conventional software construction generally loses two critical classes of design knowledge, i.e., the problem specification and the design justification. The use of a formal transformation system as a construction methodology uses the formal specifications and applies transformations to the specifications to construct the final program. A transformation system requires a specification of what program is desired. A transform is defined to be any function which maps programs into programs. A transform is often applied to a particular place of a larger program; this is called a locator. Multiple transforms may apply to any program. In the case of a sort, it may be implemented by refining it into a bubblesort or a mergesort. Design capture (i.e., history) requires knowing what was desired or specified, how it was achieved or implemented and why the implementation works or is justified. Design maintenance focuses on how to maintain the design and derive the program from the design, rather than focus the maintenance process. Maintenance deltas are a classification of the desired change. Deltas are of two fundamental varieties: specification deltas (those that affect the problem

definition) and support deltas (those that affect how the solution is implemented). Types of maintenance deltas are performance deltas, functional deltas, technology deltas, method deltas, performance predicate library deltas, and other deltas relating to performance measurement. A DMS is the construction of an incremental maintenance system. System analysts compare needs against existing system specifications, and produce maintenance deltas. The deltas are integrated into the existing design history for the existing software artifact, producing a revised history and a revised artifact. Baxter makes the design more complicated in his presentation than is necessary. It is good to include the design in the maintenance phase, but it will not replace the standard life cycle.

To implement software systems effectively, quality and software improvement must be combined with the process. Litton Industries, a systems and software integrator, base their software process improvement (SPI) on an integration of its corporate quality improvement (QI) program and the model-based initiatives of the Software Engineering Institute (SEI). Quality in their daily work identifies, controls, and improves key work processes. Priority management focuses on achieving breakthrough improvements in the highest priority areas (Hollenbach et al. 1997). Bennets et al. (1998) introduced a soft systems methodology designed to assist the resolution of ill-structured problems. This work shows that information systems development is not well structured. It emphasizes the importance of the political and human factors involved in the process. Collecting and analyzing metrics is a critical component in identifying how and where to make improvements to the software process. Ebert (1999) proposed a method of technically controlling software projects. This technique identifies, measures, accumulates, analyzes

and interprets project information for more accurate planning and tracking, decision-making, and cost accounting.

Hoffnagle and Beregi (1985) define an architecture of a software engineering support facility to support long term process experimentation, evolution, and automation. They presented three software development challenges. The first challenge is that of the customers' demand for software that existing resources cannot satisfy. Second, as technology to develop larger and more complex software systems is realized, the use of these systems and the demand for increasingly complex systems expose problems that exceed the capabilities of the technology, and finally, the need for the achievement of uncompromising quality and increased productivity. They provide some key definitions of terms. A definition of quality is the absence of any form of defect. A definition of productivity is the ability to achieve the goal of quality with the minimum expenditure of resources. The goals of a software engineering support facility are to provide an integrated environment for the support of software development tools and software development process automation. The facility must be flexible so as to accommodate local process and tool variations. It must run in many operating environments, each using different processes, life-cycle methodologies, and tools. The architecture must specify a facility framework, functions, data, interfaces, and event recording. Process and tool independence is required to support flexible process and tool evolution. System and data service isolation is needed to support tool portability. A common data model and a consistent user interface will support tool and user integration. And a process mechanism is needed to support formal process definition and automated process control.

There are several problems that accompany this approach. First, tools are not portable enough to be moved from one environment to another and therefore must be rewritten; this increases the maintenance costs. Tools and methodologies are essentially independent and must be united by a common interface, still not sharing data. Knowledge of the tools and processes may be limited to a few organizations and/or individuals. Tools and/or methodologies are often designed for a specific operating system. And finally, a variety of data organizations and data base management systems for storing tools cannot share results. Traditional approaches centered on methodologies that define boundaries for the developer. When tools were introduced, both manual and automated, they reinforced the concept of methodologies. Emphasis is shifting from using tools and methodologies on low level design to areas of high level design, requirements, and maintenance. Concepts of quality are changing from defect detection to defect prevention. Maintenance remains a major cost. Tools and methodologies improve software quality and productivity, but problems still exist (Hoffnagle and Beregi 1985).

Wojtkowski and Wojtkowski (1990) present a comprehensive overview of Computer Aided Systems Engineering (CASE). They present two views of CASE. First, standalone CASE, is where standalone tools automate a specific task in a single project. Standalone tools place the integration burden on the user of the tool and offer flexibility that allows for different development methodologies. The second view of CASE is integrated CASE, which describes fully integrated environments that support almost all

of the life cycle on more than one project. They encourage the user to accept a particular vendor's lifecycle definitions, techniques, and development methodologies.

Texas Instruments' Composer provides the most comprehensive and sophisticated CASE tool available. (Note: Composer was originally IEF [Information Engineering Facility] and is now Cool:Gen which is owned by Sterling Software.) With Cool:Gen users can capture information needs conceptually and transform them into executing application systems. Key elements of Cool:Gen include the support for the entire scope of the lifecycle, fully integrated diagrams, automatic transformation of results from one stage of the process to another, artificial-intelligence-based inference rules, built-in rules for consistency and completeness, automatic consistency checking, automatic confirmation of a task at each stage of the process, automatic high-level language and database code generation, and project coordination. To some, CASE describes a productivity tool; to others it means a fully integrated environment that supports most of the system development life cycle (such as Cool:Gen). Standalone tools put the emphasis on the user where the integrated environments constrict the user to the vendor's concepts. Determining which is better depends on the user's requirements. European vendors use the term Integrated Software Production Environments (ISPE) to distinguish between standalone systems and integrated environments. Manley's (1984) definition of CASE is adopted here as a system of automated software life cycle support aids that permits the generally accepted principles of software engineering to be used effectively in a practical and coordinated manner.

It is not yet possible to produce complex application programs with some procedural code. Research on CASE tools is in the area of three categories: development, implementation, and managerial. CASE can be considered as two generations. First, it is the collection of tools that automate manual processes, and second it incorporates a database tool-set and support for geographically-distributed development teams. IBM's product, AD/Cycle, which appeared in the 1989 set of standards of development for the U.S. industry, set the US standards in the area of CASE. European standards are established for integrating the European community. In Japan, research is government-sponsored and is aimed at using Artificial Intelligence (AI) to get ahead of U.S. technology. A serious problem with CASE tools is the inability to capture requirements. Organizations often do not do this well. Size of software systems continues to rise, predicted to increase by a factor of ten every ten years. The important research area is the data repository. In an integrated system, the data repository must support storage, retrieval, version management, and configuration management of products used in the software development. Standards are being developed by ANSI. The success of CASE technology is dependent on industry agreement of standards. Other research areas are reverse engineering, hypertext, and the use of CASE in organizational policy support. Implementation of CASE integrated environments may take a long time because of the investment that some companies have already made in collections of standalone tools. Understandably, they may be reluctant to dispose of them in favor of a newer product. The Software Engineering Institute (SEI) is working on ways to put the best software methods into use. The current state is such that more knowledge exists about developing

good software than is actually being put into practice. This is a result of not enough attention being paid to the overall development process. The consensus is that before CASE can be widely accepted, there must be evidence that it actually improves system development (Wojtkowski and Wojtkowski 1990).

Highly Regarded Software Process Models

There are eight highly regarded software improvement methodologies and measures used in this study. A high level overview of each of these is provided. They are as follows: The Software Engineering Institute's (SEI) Capability Maturity Model (CMM), ISO 9000 (International Organization for Standardization), Malcolm Baldrige Criteria Category 2, Software Productivity Research (Capers Jones), Software Engineering Laboratory (SEL), R.S. Pressman and Associates Inc., Real Decisions (Gartner Group), and Nolan, Norton & Company.

Software Engineering Institute's (SEI) Capability Maturity Model (CMM)

The CMM for software was initially produced in November 1986 by a team at SEI under the direction and vision of Watts Humphrey. Its initial release was in September 1987. Its intent is to provide a simple tool to identify areas where organization's software processes need improvement. The CMM is the foundation for systematically building a tool set for software process continuous improvement. The SEI CMM is the premiere process model in use today. It gained its momentum from use by U.S. government contractors and has recently become popular in Europe (Paulk et al.

1993). Its primary weaknesses are its complexity to use and difficulty in tailoring. As a result, companies have found they spend more money deploying it than the payback received, given the life expectancy of the software system (Baker 1996). The CMM was intended to be a coherent, ordered set of incremental improvements, all having experienced success in the field, packaged into a roadmap that showed how effective practices could be built on one another in a logical progression (Herbsleb, Zubrow, Goldenson, Hayes and Paulk 1997). The following Figure 1 provides the components of the CMM. Each organization or project receives certification at level 1 through level 5 given their level of maturity with their software process.

Level	Focus	Key Process Area	Result
5 Optimizing	Continuous process improvement	Defect Prevention Technology innovation Process change management	Productivity & Quality
4 Managed	Product and process quality	Process measurement and analysis Quality management	
3 Defined	Engineering process	Organization process focus Organization process definition Peer reviews Training program Inter-group coordination Software product engineering Integrated software management	
2 Repeatable	Project management	Software project planning Software project tracking Software subcontract management Software quality assurance Software configuration management Requirements management	
1 Initial	Heros		Risk

Figure 1. Capability Maturity Model (CMM) developed by the Software Engineering Institute (SEI) (Paulk et al. 1993)

ISO 9000 (International Organization for Standards)

ISO 9000 was initiated in 1987 by the International Organization for Standardization. It is the most used and wide spread quality measure in Europe. Many governments require ISO 9000 certification on the processes used to create their products. There are several sub-classes of standards: ISO 9001, ISO 9002, ISO 9003, and ISO 9004. ISO 9003 is the model for quality assurance in final inspection and testing, and it is this standard that is used to verify conformance to requirements. Thus many companies worldwide use ISO 9003 as their success measure for a software project (Hayes 1994).

Malcolm Baldrige Criteria, Category 2

The Malcolm Baldrige Award was introduced in 1988. Its intent is to promote total quality management (TQM) as an increasingly important approach for improving competitiveness of American companies. The criteria that make up the Baldrige Award focus on a strong balance between business results and customer satisfaction. Category 2 focuses on information and analysis. A set of criteria, within category 2, assesses the management of information and data, competitive comparisons and benchmarking, and analysis and uses of company-level data. The purpose of category 2 is to assess the types of data collected and the process by which data are analyzed and used to make decisions. The detailed criteria pursue in-depth data and information quality, integration, and availability (Brown 1996).

Software Productivity Research (SPR) - Capers Jones

Initially published in 1986, the SPR is made up of about 300 multiple choice questions. The purpose of the SPR is to place software development groups, contractors, and outsource vendors on a five-plateau excellence scale as shown in Figure 2.

SPR Excellence Scale	Meaning	Frequency of Occurrence
1 = Excellent	State of the art	2.0%
2 = Good	Superior to most companies	18.0%
3 = Average	Normal in most factors	56.0%
4 = Poor	Deficient in some factors	20.0%
5 = Very Poor	Deficient in most factors	4.0%

Figure 2. SPR Excellence Scale (Capers Jones)

The SPR assessments tend to produce a more normal bell-shaped distribution of results than do the SEI assessments. This is due in part to evaluations done within like industries. The most notable difference between the SEI and SPR methods is that the SPR assessment approach also collects baseline data on productivity, quality, schedules, costs, staffing levels, and other quantifiable factors as organizations make improvements. The SPR is also available in five languages (Jones 1994).

Software Engineering Laboratory (SEL)

The SEL pioneered its work nearly a decade before the SEI was founded. SEL was established in 1976, with the goal of reducing the defect rate of delivered software, the cost of software to support flight projects, and the average time to produce mission-support software. For over 20 years the SEL has worked to understand, assess, and improve software and the software development process within the production environment. The SEL is a cooperative effort of NASA/Goddard's FDD, the University

of Maryland Department of Computer Science, and Computer Sciences Corporation's Flight Dynamics Technology Group. Their current focus is threefold: (1) Understand baseline processes and product characteristics, such as cost reliability, software size, reuse levels, and error classes; (2) Assess improvements that have been incorporated into development projects (by measuring the impact of available technologies on the software process one can determine which technologies are beneficial to the environment and how the technologies should be refined to match the process with the environment); (3) Package and infuse improvements into the standard SEL process and update and refine standards, handbooks, training materials, and development support tools (Basil et al. 1995).

R. S. Pressman and Associates Inc.

R. S. Pressman and Associates market a tool called Process Advisor. Process Advisor is a self-directed system for software process improvement. The R.S. Pressman and Associates philosophy is "simple process improvement will only succeed if you do it yourself ... if you buy into the cultural and technological changes that must be made." Their tool acts as a personal consultant for software process improvement. It integrates a detailed workbook, automated tools, video sessions, and textbook for guidance through the software process improvement cycle (Pressman 1996).

Real Decisions (Gartner Group)

Real Decisions is a benchmarking company within the Gartner Group Corporation. They use a set of criteria to measure software process improvement. They work directly with the IT department, development, project management, and operations

teams. They will assess your software process and provide you with feedback and results of comparisons to other companies in like industries (Sharp 1996).

Nolan, Norton & Company (NNC)

Since 1981, NNC has been collecting and reporting operational and demographic data and developing statistical norms for mainframe data centers. They have a set of metrics and criteria that focus on workload, service levels, personnel, and technology costs. Their clientele is primarily made up of Fortune 1000 organizations. Their information technology (IT) strategy and management services provide direction on the development and planning of IT with linkage into the business processes. Their methodologies support various infrastructures and architectures as do most all of the approaches discussed (Nolan and Norton 1996).

System Success Theory

There have been many studies that have addressed the topic of, and the factors that contribute to, the success of an IS. This study looks specifically at the process as the key factor contributing to IS success. In a 1991 study senior IS executives ranked improving the quality of the software development process ninth in importance behind information architecture, data resource effectiveness, strategic planning, IS human resources, new technologies, responsiveness, alignment with enterprises, and IS for competitive advantage (Niederman, Branchequ and Wetherbe 1991). Their ranking emphasizes business goals, but the importance of the software development process is included in the top ten.

To define the dependent construct in this study, system success, one must review earlier attempts at this. Early studies considered accuracy and effectiveness as information system success measures, along with the level of conveying the information to the receiver as it was intended (Shannon and Weaver 1949). Mason (1978) emphasized "influence" as the key determinant of system success and defined it as the level of information received in a hierarchy of events of an information system (or the influence on the recipient of the information). Several organizations and categorizations of system success have been proposed over the past twenty years. Zmud (1978) considered the success of an MIS to be made up of three categories: user performance, MIS usage, and user satisfaction. System quality and system acceptance later evolved as areas of system success (Ives and Olson 1984). Others have emphasized success as a measure of how well the information system organization performs (Singleton, McLean and Altman 1988). The most profound research on system success to date is that of Delone and McLean (1992). They performed an extensive assessment of this topic and produced a taxonomy which posits six major dimensions or categories of IS success: system quality, information quality, use, user satisfaction, individual impact, and organizational impact (Delone and McLean 1996). Their findings serve as the dependent construct for this study.

Discriminant Analysis

Some components of discriminant analysis were proposed in the 1920's. Karl Pearson introduced the coefficient of racial likeness (CRL), which is an intergroup

distanced index. Pearson's CRL was examined extensively by G. M. Morant. P. C. Mahalanobis introduced another distance index in the 1930's. The foundation of what is known as discriminant analysis today was proposed by R. A. Fisher in the 1930's. He proposed multivariable intergroup distance as a linear combination of variables. Discriminant analysis has been categorized into two methods, predictive discriminant analysis (PDA) and descriptive discriminant analysis (DDA). DDA addresses grouping variable effects on the multiple outcome variables, or more specifically, grouping separation or group differences with respect to outcome variables. PDA addresses how accurately group membership can be predicted. Discriminant analysis may be performed on two groups or multiple groups. Initial study of discriminant analysis involved applications in biological and medical sciences. Discriminant analysis was utilized in organizational settings during the 1950's and 1960's, but its use has been limited in applied research settings over the past two decades (Huberty 1994).

Discriminant analysis is still a popular and respected management science technique. It has even recently been used in areas that contain similarities to this research, software quality and customer satisfaction. Kathryn Dansky and Diane Brannon applied discriminant analysis to determine the components of customer satisfaction in the health care profession (Dansky and Brannon 1996). Koshgoftaar et al. (1996) predicted the quality of telecommunications systems modules by applying discriminant analysis to failed modules.

Summary

This chapter has explored the prior research in the areas of systems analysis and design and information systems success, along with their evolution over that past 25 years. Eight highly regarded software process models were presented and discussed. A brief overview of discriminant analysis was presented. The next chapter takes this theory, presents the theoretical framework and progresses into the research methodology.

CHAPTER 3

THEORETICAL FRAMEWORK AND RESEARCH METHODOLOGY

Theoretical Framework

The theoretical framework in this study combines factors from MIS (Management Information Systems) frameworks and MIS success and development process frameworks. Software project implementation success is the dependent variable and a primary construct of interest. The constructs and variables that contribute to the success of a software implementation project are large in number and diverse relative to project type. As supported in the literature and discussed in the previous chapter, the software process methodology is a crucial predictor impacting a software implementation project's success or failure. Other variables to be included in the theoretical framework are external and organizational environment (Ives, Hamilton and Davis 1980). Variables that fall within the umbrella (figuratively speaking) of system process are personnel, technology (tools), budget, scope of the project, and requirements. Figure 3 provides a pictorial overview of the theoretical framework used in this research.

The items for predicting the success of a software implementation project will be taken form the software process criteria. The software process methodologies included in this research have many similarities with each other. They are all driven by similar issues and are intuitively correlated. They differ in their approach, however, and in the case of ISO (International Organization for Standardization) 9000 and the SEI CMM (Software

Engineering Institute Capability Maturity Model), they differ in their underlying philosophies. ISO 9000 focuses on the minimal requirements for a quality system, while the CMM emphasizes the need for continuous process improvement (Paulk 1995). As mentioned in the previous chapter, the ISO 9000 and SEI CMM are the most comprehensive models, and contain almost everything the other models possess and more.

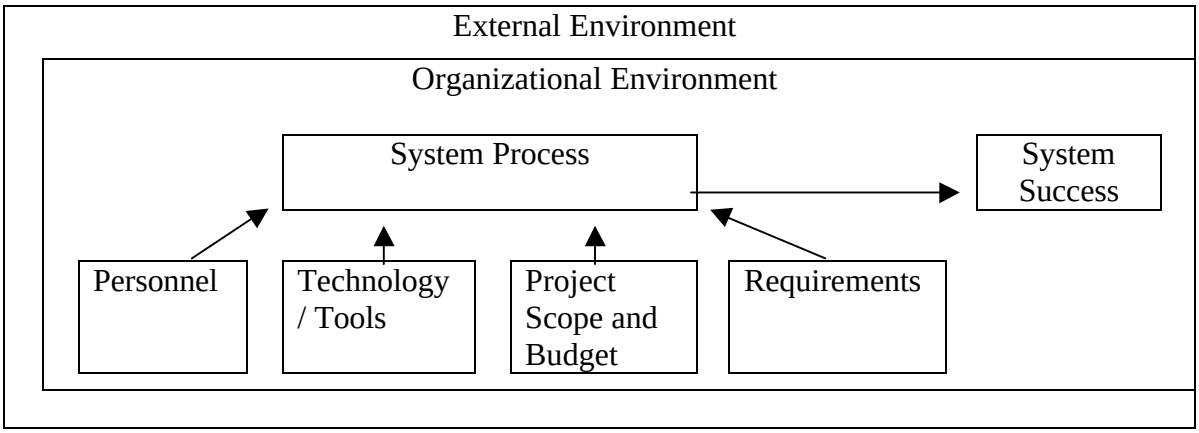

Figure 3. Theoretical Framework: A Consolidation and Mapping of the Criteria from the Software Process Models

The consolidation and development of the criteria will be primarily based on the Key Process Areas (KPAs) and clauses within the ISO 9000 clauses and CMM KPAs. Mark Paulk (1995) mapped the 20 clauses that make up ISO 9001 to the CMM key practices. Table 2 shows this mapping. "Every CMM key process area is at least weakly related to ISO 9001 in some way" (Paulk 1995, p. 82). Oskarsson and Glass (1996) believe that an organization that is certified at level 2 for the CMM would be able to fulfill most of the requirements in ISO 9001 and vice-versa. They believe that ISO 9001 is a better starting point for most projects than the CMM, but that the CMM is a better long term model for continuous improvement. They also document only four main

differences between ISO 9001 and the CMM. ISO 9001 is intended for diverse industries, not just software-specific as is the CMM. The CMM is more detailed and specific. The CMM goes into much more detail in assessing supplier's software abilities. CMM also focuses on the software development process, whereas ISO 9001 focuses on a customer-supplier relationship.

Table 2. Summary Mapping Between ISO 9001 and the CMM (Paulk 1995)

ISO 9001 Clause	CMM Strong Relationship	CMM Judgmental Relationship
4.1: Management responsibility	Commitment to perform Software project planning Software project tracking and oversight Software quality assurance	Ability to perform Verifying implementation Software quality management
4.2: Quality systems	Verifying implementation Software project planning Software quality assurance	Organization process definition
4.3: Contract review	Requirements management Software project planning	Software subcontract management
4.4: Design control	Software project planning Software project tracking and oversight Software configuration management Software product engineering	Software quality management
4.5: Document and data control	Software configuration management Software product engineering	
4.6: Purchasing	Software subcontract management	
4.7: Control of customer-supplied product		Software subcontract management
4.8: Product identification and traceability	Software configuration management Software product engineering	
4.9: Process control	Software project planning Software quality assurance Software product engineering	Quantitative process management Technology change management
4.10: Inspection and testing	Software product engineering Peer reviews	

ISO 9001 Clause	CMM Strong Relationship	CMM Judgmental Relationship
4.11: Control of inspection, measuring, and test equipment	Software product engineering	
4.12: Inspection and test status	Software configuration management Software product engineering	
4.13: Control of nonconforming product	Software configuration management Software product engineering	
4.14: corrective preventive action	Software quality assurance Software configuration management	Defect prevention
4.15: Handling, storage, packaging, preservation, and delivery		Software configuration management Software product engineering
4.16: Control of quality records	Software configuration management Software product engineering Peer reviews	
4.17: Internal quality audits	Verifying implementation Software quality assurance	
4.18: Training	Ability to perform Training program	
4.19: Servicing		
4.20: Statistical techniques	Measurement and analysis	Organization process definition Quantitative process management Software quality management

Theoretical Link to Research Framework

This section provides an overview of each of the independent constructs. The overview is a consolidation of material from the highly regarded models discussed in chapter 2 of this research. The theory behind the independent construct (software

process) is organized in a manner that represents the key contributions to this sub-discipline within the MIS field and is in a format more easily convertible to measurable variables. The creation of measurable variables of high construct validity is challenging in field research, yet it is an indispensable component of internal validity (Mitchell 1985). The independent construct questionnaire is drawn from the following section.

As with any mapping and consolidation of similar details, different terminology may be used to define a specific item. A consolidation of the software process criteria from the highly regarded models used in this research provides a structure very much like the SEI CMM (Paulk et al. 1993). Thus, the basic outline of the SEI CMM KPAs serves as an appropriate organization structure for preliminary consolidation and categorizing various software criteria (Pryor and McGuire 1997). The following section categorizes and summarizes the criteria groupings that make and serve as a tailored taxonomy for the independent construct. Each construct (category) is briefly explained, along with its role in the software implementation process and what is involved to deploy it. The instrument used to measure the independent construct and variables is developed from this. The instrument is located in appendix A. The following explanations (which are a consolidation from the highly regarded models used in this study) are critical in understanding the significance of the independent variables and constructs used in this research.

Software Project Planning

The purpose of software project planning is to establish reasonable plans for performing the software implementation and for managing software projects. Software

project planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work (Paulk et al. 1993). The plan should include a definition of the process to be used, descriptions of phases, a listing of the players involved, and identification of the tools and methods (Oskarsson and Glass 1996).

The software planning begins with a statement of the work to be performed and other constraints and goals that define and bound the software project (those established by the practices of the requirement's management process). The software planning process includes steps to estimate the size of the software work products and the resources needed, to produce a schedule, identify and assess software risks, and negotiate commitments. Iterating through these steps may be necessary to establish the plan for the software project, i.e., the software development plan. This plan provides the basis for performing and managing the software project's activities and addresses the commitments to the software project's customer according to the resources, constraints, and capabilities of the software project (Paulk et al. 1993).

To meet each planning objective, three items must be confirmed. Are the software estimates that are used in planning and tracking the software project documented? Are software project activities and commitments planned and documented? Do the affected groups and individuals agree to their commitments related to the software project?

Software Project Tracking

The purpose of software project tracking is to provide adequate visibility into actual progress so that management can take effective actions when the software project's performance deviates significantly from the software plans. Software project tracking involves tracking and reviewing the software accomplishments and results against documented estimates, commitments, and plans, and making adjustments as needed (Paulk et al. 1993).

A documented plan for the software project, i.e., the software development plan, is used as the basis for tracking the software activities, communicating status, and revising plans. Progress is primarily determined by comparing the actual software size, effort, cost, and schedule to the plan when selected software work products are completed and when milestones are accomplished. When it is determined that the software project's plans are not being met, corrective actions are taken. These actions may include revising the software development plan to reflect the actual accomplishments and re-planning the remaining work or taking actions to improve the performance (Paulk et al. 1993).

To meet the objectives for project tracking and oversight, the following four items must be validated. Are actual results and performances tracked against the software plans? Are corrective actions taken and managed to closure when actual results and performance deviate significantly from the software plans? Are changes to software commitments agreed to by the affected groups and individuals?

Software Subcontract Management

The purpose of software subcontract management is to select qualified software subcontractors and manage them effectively. Software subcontract management involves selecting a software subcontractor, establishing commitments with the subcontractor, and tracking and reviewing the subcontractor's performance and results. These practices cover the management of a software subcontract, as well as the management of the software component of a subcontract that includes software, hardware, and possibly other system components (Paulk et al. 1993). Selection of contractors is critical. Contractors with experience using a sound methodology are preferable. Contractors may not be certified to ISO 9001, if they only supply the manpower (Oskarsson and Glass 1996). The subcontractor is selected based on its ability to perform the work. Many factors contribute to the decision to subcontract a portion of the prime contractor's work. Subcontractors may be selected based on strategic business alliances, as well as technical considerations. The practices of this key process area address the traditional acquisition process associated with subcontracting a defined portion of the work to another organization. When subcontracting, a documented agreement covering the technical and non-technical, e.g., delivery dates, requirements are established and used as the basis for managing the subcontract. The work to be done by the subcontractor and the plans for the work are documented. The standards that are to be followed by the subcontractor are compatible with the contractor's work standards. The contractor supervisor ensures that these planning, tracking, and oversight activities are performed appropriately and that the software products delivered by the subcontractor satisfy their acceptance criteria. The

contractor works with the subcontractor to manage their product and process interfaces (Paulk et al. 1993).

There are four objectives of software contract management as follows. Are the subcontractors qualified? Do the contractor supervisor and the software subcontractor(s) agree to their commitments to each other? Does the contractor supervisor maintain ongoing communications with the software subcontractor? Does the contractor supervisor track the software subcontractor's actual results and performance against its commitments?

Software Quality and Assurance

The purpose of software quality assurance is to provide management with appropriate visibility into the process being used by the software project and of the products being built. Software quality assurance involves reviewing and auditing the software products and activities to verify that they comply with the applicable procedures and standards and providing the software project and other appropriate managers with the results of these reviews and audits. The software quality assurance group works with the software project during its early stages to establish plans, standards, and procedures that will add value to the software project and satisfy the constraints of the project and the organization's policies. By participating in establishing the plans, standards, and procedures, the software quality assurance group helps ensure they fit the project's needs and verifies that they will be usable for performing reviews and audits throughout the software life cycle. The software quality assurance group reviews project activities and audits software work products throughout the life cycle and provides management with

visibility as to whether the software project is adhering to its established plans, standards, and procedures. Compliance issues are first addressed within the software project and resolved there if possible. For issues not resolvable within the software project, the software quality assurance group escalates the issue to an appropriate level of management for resolution. This process covers the practices for the group performing the software quality assurance function (Paulk et al. 1993).

Software quality assurance objectives are as follows. Are the projects' software quality assurance activities planned? Is adherence of software products and activities to the applicable standards, procedures, and requirements objectively verified? Are affected groups and individuals informed of software quality assurance activities and results? Are noncompliance issues that cannot be resolved within the software project addressed by senior management?

Software Configuration Management

The purpose of software configuration management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle. Software configuration management involves identifying the selected software work products and their descriptions at a given point in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software life cycle. The work products placed under software configuration management include the software products that are delivered to the customer, e.g., the requirements document and the code, and the items that are identified with or required to create these software products, e.g., the compiler. A

software baseline library is established containing the software baselines as they are developed. Changes to baselines and the release of software products built from the software baseline library are systematically controlled via the change control and configuration auditing functions of software configuration management. This key process area covers the practices for performing the software configuration management function. The practices identifying specific configuration items/units are contained in the key process areas that describe the development and maintenance of each configuration item (Paulk et al. 1993).

The objectives for the software configuration management activities are as follows. Are software configuration management activities planned? Are selected software work products identified, controlled, and available? Are changes to identified software work products controlled? Are affected groups and individuals informed of the status and content of software baselines?

Requirements Management

The purpose of requirements management is to establish a common understanding between the customer and the customer's requirements that will be addressed by the software project. Requirements management also involves establishing and maintaining an agreement with the customer on the requirements for the software implementation project. This agreement may be documented in a specification that includes functionality, performance, safety, reliability, security, privacy, interfaces, and other customer needs (Oskarsson and Glass 1996). This agreement is generally referred to as the "system requirements" or simply the “requirements.” The customer may be anyone

or any group who will be serving as or representing the end user of the system. The agreement on requirements covers both the technical and non-technical (e.g., delivery dates) requirements. The agreement forms the basis for estimating, planning, performing, and tracking the software project's activities throughout the software life cycle (Paulk et al. 1993).

The allocation of the system requirements to software, hardware, and other system components (e.g., humans) may be performed by a group external to the software implementation group. Within the constraints of the project, appropriate steps must be taken to ensure that the system requirements are documented and controlled. To achieve this control, the software group reviews the initial and revised system requirements allocated to resolve issues before they are incorporated into the software project. Whenever the system requirements allocated to software are changed, the affected software plans, work products, and activities are adjusted to remain consistent with the updated requirements (Paulk et al. 1993).

Objectives for the management of the requirements involve the following two items. Do the agreed upon system requirements establish a controllable baseline for project use? Are the projects software plans, products, and activities kept consistent with the system requirements allocated to software?

Organization Process Focus

The purpose of organization process focus is to establish the organizational responsibility for software process activities that improve the organization's overall software process capability. Organization Process Focus involves developing and

maintaining an understanding of the organization's and projects' software processes and coordinating the activities to assess, develop, maintain, and improve these processes. The organization provides the long-term commitments and resources to coordinate the development and maintenance of the software processes across current and future software projects. A group accountable to management must be responsible for the organization's software process activities. It is specifically responsible for the development and maintenance of the organization's standard software process and related process assets, and it coordinates the process activities with the software projects (Paulk et al. 1993).

For each organization process focus objective to be met, the following items must be validated. Are the software process development and improvement activities coordinated across the organization? Are the strengths and weaknesses of the software processes used and identified relative to a process standard? Are the organization-level process development and improvement activities planned?

Organization Process Definition

The purpose of organization process definition is to develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for cumulative, long-term benefits to the organization. Organization process definition involves developing and maintaining the organization's standard software process, along with related process assets, such as descriptions of software life cycles, process tailoring guidelines and criteria, the organization's software process database, and a library of software process-related documentation. These assets may be

collected in many ways, depending on the organization's implementation of organization process definition. The descriptions of the software life cycles may be an integral part of the organization's standard software process or parts of the library of software process-related documentation may be stored in the organization's software process database. The organization's software process assets are available for use in developing, implementing, and maintaining the projects' defined software processes (Paulk et al. 1993).

To ensure that each objective is achieved for organization process definition, the following two items must be validated. Is a standard software process for the organization developed and maintained? Is this information collected, reviewed, and made available?

Peer Reviews

The purpose of peer reviews is to remove defects from the software work products early and efficiently. A secondary purpose is to develop a better understanding of the software work products and of defects that might be prevented. Peer reviews involve a methodical examination of software work products by the producers' peers to identify defects and areas where changes are needed. The specific products that will undergo a peer review are identified in the project's defined software process and scheduled as part of the software project planning activities (Paulk et al. 1993). With the CMM, peer reviews refer to another human reviewing a software product. However, in ISO they also include a list of tools that are classified as “code analyzers.” There are several classifications of code analyzers, each with a unique purpose. Cross-reference/browsers help determine the interrelationships in the code. Call structure

generators help define the overall logic structure. Performance analyzers determine which modules (parts) of the code consume unnecessary resources. Metrics analyzers identify complexities within the code. Auditors specifically address areas that do not conform to standards. And requirement's tracers ensure the linkage between requirements and the delivered system (Oskarsson and Glass 1996).

To ensure the objectives of peer reviews are met, the following two items must be accomplished. Are peer review activities, i.e., inspections and/or walkthroughs, planned? And are defects in the software work products identified and removed?

Training Programs

The purpose of training programs is to develop the skills and knowledge of individuals so they can perform their roles effectively and efficiently. Training programs involve identifying the training needed by the organization, projects, and individuals, then developing or procuring training to address the identified needs. Each software project evaluates its current and future skill needs and determines how these skills will be obtained. Some skills are effectively and efficiently imparted through informal vehicles, e.g., on-the-job training and informal mentoring, whereas other skills need more formal training vehicles, e.g., classroom training and guided self-study, to be effectively and efficiently imparted. The appropriate vehicles are selected and used. This key process area covers the practices for the group performing the training function. The practices identifying the specific training topics, i.e., knowledge or skill needed, are contained in the Ability to Perform common feature of the individual key process areas (Paulk et al. 1993). Additionally, the type of training to be delivered should be agreed upon with the

customer, and a method should be in place to provide feedback on the satisfaction level with the training (Oskarsson and Glass 1996).

The objectives of the training program's process are as follows. Are training activities planned? Is training for developing the skills and knowledge needed to perform software management and technical roles provided? Do individuals in the software engineering group and software-related groups receive the training necessary to perform their roles?

Inter-group Coordination

The purpose of Inter-group Coordination is to establish a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently. Inter-group Coordination involves the software engineering group's participation with other project engineering groups to address system-level requirements, objectives, and issues. Representatives of the project's engineering groups participate in establishing the system-level requirements, objectives, and plans by working with the customer and end users, as appropriate. These requirements, objectives, and plans become the basis for all engineering activities. The technical working interfaces and interactions between groups are planned and managed to ensure the quality and integrity of the entire system. Technical reviews and interchanges are regularly conducted with representatives of the project's engineering groups to ensure that all engineering groups are aware of the status and plans of all the groups, and that system and inter-group issues receive appropriate attention (Paulk et al. 1993).

The objectives of inter-group coordination are as follows. Are the end users or their representatives in agreement on the requirements? Are the commitments between the groups agreed to by those affected? Do the project groups identify, track, and resolve inter-group issues?

Software Product Engineering

The purpose of software product engineering is to perform consistently a well-defined process that integrates all the software activities to produce correct, consistent software products effectively and efficiently. Software product engineering involves performing the engineering tasks to build and maintain the software using the project's defined software process and appropriate methods and tools. The tasks include many items, such as analyzing the system requirements allocated to software, developing the software requirements, and developing the software architecture. They also include designing the software, implementing the software in the code, integrating the software components, and testing the software to verify that it satisfies the specified requirements, i.e., the system requirements allocated to software and the software requirements. Additionally, documentation is required to perform the software engineering tasks, e.g., software requirements document, software design document, test plan, and test procedures. This documentation should be developed and reviewed to ensure that each task addresses the results of predecessor tasks and the results produced are appropriate for the subsequent tasks, including the tasks of operating and maintaining the software. When changes are approved, affected software work products, plans, commitments, processes, and activities are revised to reflect the approved changes (Paulk et al. 1993).

The two objectives of software product engineering are as follows. Are the software engineering tasks defined, integrated, and consistently performed to produce the software? Are the software work products kept consistent with each other?

Integrated Software Management

The purpose of integrated software management is to integrate the software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets, which are described in organization process definition. Integrated software management involves developing the project's defined software process and managing the software project using this defined software process. The project's defined software process is tailored from the organization's standard software process to address the specific characteristics of the project. The software development plan is based on the project's defined software process and describes how the activities of the project's defined software process will be implemented and managed. The management of the project's size, effort, cost, schedule, staffing, and other resources is tied to the tasks of the project's defined software process. Since the projects' defined software processes are all tailored from the organization's standard software process, the software projects can share process data and lessons learned. The basic practices for estimating, planning, and tracking a software project are described in the Software Project Planning and Software Project Tracking and Oversight key process areas. They focus on recognizing problems when they occur and adjusting the plans and/or performance to address the problems. The practices of this process build on, and are in addition to, the practices of those two key

process areas. The emphasis of integrated software management shifts to anticipating problems and acting to prevent or minimize the effects of these problems (Paulk et al. 1993).

The objectives of integrated software management are as follows. Is the project's defined software process a tailored version of the organization's overall standard software process? Is the project planned and managed according to the project's defined software process?

Variables and Surrogates

The construct Table 3 lists the constructs of interest, variables, types, and surrogates. The independent variable is project success as a categorical variable, i.e., success or failure. The success or failure category is derived from the selection of dependent variables as listed, derived from the Delone and McLean model (1992). The independent variables are selected criteria from the various software process models previously discussed. Some of the previously discussed criteria were consolidated for better organization and more concise testable constructs, e.g., software project planning and tracking were combined. The consolidated process models criteria resulted in 11 independent constructs.

Table 3. Construct and Variable Table

Dependent Construct	Question Item	Dependent Variable Type	Survey Question or Surrogate
Software Implementation Project Success		Discrete: 0 = Failed 1 = Success	Compilation of surrogate variable questions 1-15
	Completed and Delivered On Time	Continuous	1
	Percent of Requirements Met	Continuous	2
	Cost Forecast to Actual Incurred	Continuous	3
	Data Accuracy and Currency	Continuous	4
	Ease of Use / Learning	Continuous	5
	Accessibility	Continuous	6
	Usefulness	Continuous	7
	Flexibility	Continuous	8
	Reliability	Continuous	9
	Sophistication	Continuous	10
	Integration	Continuous	11
	Utilization	Continuous	12
	Acceptability	Continuous	13
	Overall Satisfaction	Continuous	14
	Business Impact	Continuous	15
Predictor	Constructs	Independent Variable Type	Survey Question or Surrogate
Software Process Criteria			
	Process and Quality Management	Continuous	33-34, 37, 39-42
	Organizational Process	Continuous	1-6, 8
	Peer Reviews	Continuous	51,52
	Training Programs	Continuous	7, 11
	Inter-group Coordination	Continuous	22-24
	Software Project Planning	Continuous	14-20, 25
	Software Project Measurement and Analysis	Continuous	26-32, 35, 36, 38

Predictor	Constructs	Independent Variable Type	Survey Question or Surrogate
	Software Subcontract Management	Continuous	67-69
	Software Quality Assurance	Continuous	13, 61-66
	Software Configuration Management	Continuous	12, 54-60
	Requirements Management	Continuous	43-50

Research Questions

This research will address the overall question: what tasks contribute to the success or failure of a software implementation project? The specific research questions assessing the contribution to the success or failure of a software implementation project are as follows.

- What activities contribute to the success of an information system implementation project?
    - Does process and quality management contribute to the success or failure of an information system implementation project?
    - Does the organizational process contribute to the success or failure of an information system implementation project?
    - Do peer reviews contribute to the success or failure of an information system implementation project?
    - Do training programs contribute to the success or failure of an information system implementation project?
    - Does inter-group coordination contribute to the success or failure of an information system implementation project?
    - Does software project planning contribute to the success or failure of an information system implementation project?

- Does software project measurement and analysis contribute to the success or failure of an information system implementation project?
- Does software subcontract management contribute to the success or failure of an information system implementation project?
- Does software quality assurance contribute to the success or failure of an information system implementation project?
- Does software configuration management contribute to the success or failure of an information system implementation project?
- Does requirements management contribute to the success or failure of an information system implementation project?

- What are the discriminating indicators of the success or failure of an information system implementation project for differing levels of quality requirements?

Hypotheses

The research questions are reworded into 12 hypotheses as listed below for testing. These 12 hypotheses are written in the alternative form. Hypotheses one through eleven represent the various constructs that the literature indicates are recommended to achieve a successful system implementation project. Hypothesis 12 seeks to identify a succinct set of activities that best discriminates between successful and unsuccessful projects.

H1: Process and quality management contributes to the success or failure of a software implementation project.

H2: Organizational process is a contributes to the success or failure of a software implementation project.

H3: Peer reviews contributes to the success or failure of a software implementation project.

H4: Training programs contributes to the success or failure of a software implementation project.

H5: Inter-group coordination at contributes to the success or failure of a software implementation project.

H6: Software project planning contributes to the success or failure of a software implementation project.

H7: Software project measurement and analysis contributes to the success or failure of a software implementation project.

H8: Software subcontract management contributes to the success or failure of a software implementation project.

H9: Software quality assurance contributes to the success or failure of a software implementation project.

H10: Software configuration management contributes to the success or failure of a software implementation project.

H11: Requirements management contributes to the success or failure of a software implementation project.

H12: A reduced set of indicators of software implementation project success, provide a better classification of success or failure than the naive prediction rule.

Research Methodology

Type of Research and Research Design

This study involves a field survey research strategy conducted in three phases, i.e., criteria development, data collection, and analyses. Each phase is explained below, including the activities that take place and their resulting outcomes.

Phase 1: Develop Criteria

The first phase results in the development of the measurement instruments, surveys for the independent and dependent constructs and variables, and all additional informational variables of interest, e.g., demographics of participants. The instrument for the independent constructs and variables was derived by the consolidation of a "best set" of criteria, resulting in 11 constructs, from the highly regarded software process models discussed earlier in this chapter. This results in 67 questions representing the questions for the 11 constructs. The constructs and their associated question numbers, from which the questionnaire was developed, are located in Table 3.

The dependent construct, system success or failure, and variables is developed from the Delone and McLean (1992) MIS list of success measures, which are represented in the construct and variable Table 3. From their work, I developed a questionnaire to measure the system quality success of a project

Expert Panel

A panel of two industry experts and two academic experts, all four with extensive quality and/or software process experience, critiqued and revised the surveys. The executive panel was instructed to address the topics of readability, clarity, consistency, reliability and construct validity, by confirming in their opinion that the survey questions measure what they purport to measure. The panel verified that each question from both the success and process surveys mapped to and represented its related construct as depicted in Table 3.

Measurement

This section explains how the survey results are numerically computed and prepared for further statistical analyses. Each criteria within each observation is given a numeric value for its level of compliance to the question, as depicted in Table 4.

Table 4. Numeric Representation of Level of Compliance

Numeric Value	Level of Compliance
0	Don't know or not applicable
1	Totally not compliant
2	Strongly not compliant
3	Not compliant
4	Somewhat not compliant
5	Neutral
6	Somewhat compliant
7	Compliant
8	Very compliant
9	Totally (100%) compliant

Dependent Variables Calculation

The results for the dependent variables will be consolidated into a discrete success or failure variable, i.e., 1 for success and 0 for failure. The dependent questionnaire results for each observation are combined in order to obtain a single discreet value of success or failure. This is accomplished by taking the average of each observation's dependent survey numerical values (1-9, excluding the 0s). Observations scoring a 5.0 or less are classified as failed. Those resulting in a score greater than 5.0 are classified as successful. A score of 5 represents a neutral response. Scores below 5.0 indicate non-compliance and are thus considered unsuccessful. Scores above 5.0 indicate compliance and are considered successful.

Independent Variables Calculation

The independent questionnaire results represent a predictor of success or failure at the aggregate level. Constructs are represented by several independent variables making up their criteria grouping. All independent variables will receive a score from zero to nine as indicated in Table 4. The numerical score for the constructs consists of summing the responses to the questions that represent the construct. Based on the previously discussed numerical interpretation of the independent and dependent variable results, the survey is now in a valid format for discriminant analysis and construct testing via an ANOVA procedure.

Phase 2: Assessment and Data Collection

The selected set of projects for this study can be considered to be a homogeneous sample since each project is an internal business system implementation. Companies and projects were identified, along with a corporate contact. Each of these projects is scored for each dependent and independent construct criterion. The participants to complete the survey for each project consist of an end-user representative of the system for the dependent survey and a project leader or software quality expert involved in the implementation for the independent survey.

The end user representative is selected by the corporate contact. And he/she assesses system success by completing the system success portion of the survey developed from the Delone and McLean model (1992). The project leader, or the software quality resource from the implementation team, also selected by the corporate contact, assesses the level of conformance (i.e., 0-9) that each criterion was deployed for

that specific project. This is accomplished by completing the criteria contribution portion of the process survey, independent construct. The scoring of each project observation is done in a one-on-one setting as directed by the corporate contact. The survey is completed for each project on-line through the participant's internet browser, e.g., MS Internet Explorer, Netscape, and then submitted electronically via the internet to the primary investigator. Each criterion is scored on a scale of 0-9 as depicted in Table 5.

Population and Subjects

The population is made up of a sample of successful and unsuccessful software implementation projects. Several companies agreed to participate in this research and appointed a contact/liaison to assist with the deployment of the surveys. The companies identified are Allied Signal, AMR, AT&T, EDS, Frito Lay, GTE, Harbinger, IusaCell, John Deere, Omnipoint, Ratheon, RBS Group, Sterling Software, Texas Instruments Inc., and Intersolve.

Data Collection

For each project the numeric score of each criterion is collected from a survey on the internet and loaded directly into an MS Excel spreadsheet, then imported into SAS, a statistical package. The depended variable is calculated in Excel before import into SAS as a discrete 0 or 1. For example, if only three independent variable criteria were used versus the actual 67, the format of the data collected may appear as follows in Table 5.

Table 5. Sample Data Layout

		Independent Variables			Dependent Variable
Company	Project Name	Criterion 1	Criterion 2	Criterion 3	Success or Failure
Ratheon	Y2K	4	3	9	0
Sterling Software	PeopleSoft	3	7	2	0
EDS	SAP	1	9	1	1
IsuaCell	Accounts Payable	9	1	9	1

Phase 3: Analyses

Discriminant analysis is used to identify items that contribute to the discrimination between the successful and failed software projects. In addition, to test for a difference between the successful and unsuccessful groups for each of the 11 constructs identified in this study an ANOVA procedure is used.

Since one of the objectives of this study is to include only those variables that improve the discriminatory power of the discriminant function at a given significance level, stepwise discriminant analysis is used. A significance level of .05 is used for the inclusion or deletion of variables that can “best” discriminate between the two groups. Thus, the resulting discrimination variables best determine the success or failure of a system. The discriminant analysis results are validated by assessing the statistical significance of the classification rate for each group and for the overall classification. The Z group will be used to test whether the classification rate is of practical importance.

The questions that make up each construct are summed to create a new variable representing each construct. Once the constructs are created from the survey data, an ANOVA procedure determines significance between successful and failed systems for

each construct. Procedures are performed to test and confirm that no assumptions are violated (Sharma 1996). In the event that the data do not come from a normal distribution, one can transform the data such that the distribution of the transformed variable is normal (Johnson and Wichern 1988). Once the normality assumption has been confirmed, the Box's M statistic provides the results of the test for checking equality of covariance matrices. The Box's M statistic provides the generalized variances given by the determinant of the covariance matrix, for both groups, i.e., failed and successful projects. If the variability of the two groups is not approximately equal, then appropriate transformations must be performed to correct this (Sharma 1996).

Summary

This chapter presented the theoretical and research frameworks used to address the research questions, defined variables and surrogates, presented the main hypotheses phrased as constructs, discussed the specific methodology employed in the research, and developed the testable hypotheses. The following chapter presents an analysis of the results.

CHAPTER 4

DATA ANALYSIS

Introduction

This chapter presents the descriptive statistics and results of hypothesis testing for this study. Discriminant analysis was performed on the data resulting in 12 discriminating items out of the initial 67 items between successful and unsuccessful projects. ANOVAs were performed on the 11 constructs providing evidence that three constructs do not significantly impact the success of an information systems implementation project and seven constructs do impact software implementation project success.

Demographics

The surveys were completed via an electronic form on the internet by 80 companies for 165 homogeneous projects over a six week period. The projects are homogeneous in that they are all internal business system applications. The number of successful projects received was 137. The number of unsuccessful projects received was 28. The participants completing the surveys were appointed by a corporate contact for each of the 80 companies. Each corporate contact identified two participants to complete the two surveys for each project. The system success survey was completed by a user of the system, or a representative knowledgeable of system success that was not biased by

the project implementation team. The system process survey was completed by a project leader or a software quality representative who was directly involved with the project implementation. Table 6 provides a listing of the companies and the number of projects that participated in the study. A desirable goal was to obtain a balance of unsuccessful and successful projects from each company. Post discussions as to the small number of unsuccessful responses, 28 out of 165, were held in person and by phone with 36 of the corporate contacts. Their explanations were similar. Completed unsuccessful projects are more difficult to find today than in the past. This is because the majority of unsuccessful projects are now identified earlier in the life cycle before completion by management. Once identified, a potentially unsuccessful system project is either corrected or stopped before additional resources and financial investments are wasted. A few companies did not wish to report their unsuccessful projects for fear that a breach in confidentiality may impact future business. The large number of successful projects (137 to 28) tends to make the analysis unbalanced. However, the Z tests conducted for practical importance of the classification rates were significant. Thus indicated that the discriminant function was of value in predicting future success of a project. These test results are discussed later in this chapter in the results section.

Table 6. Company and Project Participation

Company	# Projects
AKF	1
Alltel	1
AMR Sabre	3
AMX International	1
Andrew Corporation	2
ARCO	1

Associates	2
Atlantic Richfield	2
Bass Enterprises	2
Bell Helicopter	1
Berkson Web Design	1
BFE	1
BMC Software	1
Boeing	1
Brazo's Electric	1
Burlington Northern	1
Cambridge Technology	4
COMPUCOM	1
Computer Associates	1
Copper Software	1
CWYA	1
DalTile	1
Data Systems	1
DBSM Group	1
John Deere	2
Dell	1
Dillards	1
Ebby Halliday	5
EDS	21
Emprise Solutions	1
Ericsson	1
Ernst & Young	3
Fidelity	2
Frito-Lay	6
GTE	5
HDVest	1
Heckett Multiserv	1
Hypergraphics	1
IBM	5
ICT	1
Insight Associates	4
Interline	1
Interstate Batteries	1
InterVoice	1
IUSACELL	4
JCPenney	6
JD Edwards	1
Klein	1

Koche Industries	3
Lockheed Martin	4
Lubrizol	1
MBNA	1
MCI	1
Metro-Media Restaurant	2
Micro-Computer	1
New Mont Gold	1
Nortel	1
Northrop Gumman Aerospace	1
OCC	1
Omni Point	5
Paging Networks	2
PCS Health	3
PepsiCo	1
Pinnacle	1
Profit Pacesetters	2
Raytheon	3
SCT Corporation	1
SIAM Commercial Bank	1
Southwest Airlines	1
Sprint	1
Sterling Software	7
Summit	1
Sun Microsystems	1
Texas Instruments	2
Tricon Global Restaurants	1
TTI	1
Universal Display Fixtures	1
University of North Texas	2
University of Texas at Dallas	1
Western Wireless	1
Total Participating Projects	165

The forecasted and actual U.S. dollars invested in each project was provided by 111 projects. The mean forecasted investment is $2,440,531 and the mean actual project investment is $6,912,760. Thus, on the average these projects ran way over their budgeted financial forecast. The forecasted people month investments were provided by

140 projects. The actual people months investments required to complete the project were provided by 143 projects. The mean actual people month investment and forecasted people month investment for the participating projects is 57.09 people months and 99.57 people months respectively. Once again the expected effort was much less than the actual effort. The standard deviation and the sample variance for the four reported variables also indicate a dispersed and greater variation in the actual investments when compared to the forecasted investments. The standard deviation for the forecasted dollar investment is 7,700,472.42. The standard deviation for the actual dollar investment is 40,131,709.93. Additional descriptive statistics for the forecasted to actual financial and people month investment variables are reported in Table 7.

Table 7. Descriptive Statistics Summary of Forecasted to Actual Investments

	Forecasted $	Actual $	Forecast PM	Actual PM
Mean	2,440,531	6,912,760	57.09	99.57
Standard Error	730,896.49	3,809,133.33	11.03	36.34
Median	200,000	233,000	13	15
Mode	500,000	1,000,000	24	24
Standard Deviation	7,700,472.42	40,131,709.93	130.58	434.58
Sample Variance	5.92973E+13	1.61055E+15	17053.35	188867.04
Kurtosis	26.97	85.99	44.04	115.88
Skewness	5.01	8.95	5.77	10.33
Range	49,999,600	399,998,800	1199.5	4999.5
Minimum	400	1,200	0.5	0.5
Maximum	50,000,000	400,000,000	1200	5000
Sum	270898941	767316450	7993.1	14238.88
Count	111	111	140	143

The 11 primary independent constructs of this study are represented by the 67 questions. The following Table 8 provides the construct number, name, a brief

description and the question numbers from the process survey that represent each construct. Questions that were unanswered were given an average of the other questions within their construct grouping. For example, if question 33 was unanswered for a project it was given a value equal to the average of questions 34, 37, and 39-42. The zero responses representing not applicable were not included in the averages.

Table 8. Independent Construct Definitions and Variable Representation

#	Construct Name	Construct Description	Survey Questions
C1	Process Quality Management	Control process performance quantitatively and develop a quantitative understanding of the quality of the project's products and achieve quality goals.	33, 34, 37, 39-42
C2	Organizational Process	Establish organizational responsibilities for the software activities that improve the organization's overall process capabilities and maintain the assets across projects.	1-6, 8
C3	Peer Reviews	Remove defects from software work products early and efficiently.	51, 52
C4	Training Programs	Develop the skills and knowledge of individuals so they can perform their roles effectively and efficiently.	7, 11
C5	Inter-Group Coordination	Establish a means for the participation among interacting groups.	22-24
C6	Software Project Planning	Establish reasonable plans for performing and managing the software project.	14-20, 25
C7	Software Project Measurement and Analysis	Identify and maintain metrics that provide visibility into progress and take effective actions on deviations from the plans.	26-32, 35, 36, 38
C8	Software Subcontract Management	Select and manage qualified software subcontractors effectively.	67-69
C9	Software Quality Assurance	Provide management with visibility into the software process and products being built.	13, 61-66
C10	Software Configuration Management	Establish and maintain the integrity of the products and software project throughout the project's software life cycle.	12, 54-60
C11	Requirements Management	Establish a common understanding between the customer and the software project.	43-50

The following Table 9 provides the descriptive statistics for each of the 11 independent constructs. Table 9 contains responses from both the successful and unsuccessful observations. Here each construct may be descriptively assessed. Constructs 3, 4, 5, 8 received lower mean scores than did the other 7. This is due to fewer questions representing those constructs and not necessarily that less emphasis is placed on them from a project perspective.

Table 9. Construct Descriptive Statistics

	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9	C 10	C 11
Mean	33.33	36.76	10.32	10.21	12.99	44.24	39.02	17.25	27.46	33.15	44.47
Standard Error	1.34	1.36	0.44	0.39	0.34	1.48	1.98	0.66	1.46	1.65	1.52
Median	36	43	13	12	14	48	41.5	18	25	34	49
Mode	13	49	14	13	16	37	22	15	11	5	34
Standard Deviation	17.11	17.50	5.56	5.05	4.37	19.02	25.25	6.79	18.54	21.18	19.40
Sample Variance	292.84	306.51	30.92	25.54	19.15	361.98	637.72	46.11	344.10	448.61	376.65
Kurtosis	-1.16	-0.81	-1.21	-1.08	1.42	-0.19	-1.28	0.23	-1.38	-1.37	-0.53
Skewness	-0.26	-0.64	-0.51	-0.50	-1.39	-0.81	0.078	-0.85	0.13	-0.03	-0.64
Range	62	62	17	17	17	69	85	25	62	71	69
Minimum	1	1	1	1	1	3	1	2	1	1	3
Maximum	63	63	18	18	18	72	86	27	63	72	72
Sum	5433	6067	1641	1676	2118	7301	6322	1777	4422	5405	7249
Count	163	165	159	164	163	165	162	103	161	163	163

Hypothesis Testing

A summary of each hypothesis' result is provided. Hypotheses H1 through H11 were tested using an ANOVA procedure and the results are summarized in Table 10. Hypothesis H12 was tested with discriminant analysis and the discussion on H12 follows that of the other hypotheses.

Table 10. Summary of Results of Hypotheses H1 through H11

	Hypothesis	Mean	Significance
H1	Process and quality contributes to the success or failure of a software implementation project.	Success = 37.3 Failed = 23.3	p = .0000215 Significant
H2	Organizational process contributes to the success or failure of a software implementation project.	Success = 40.5 Failed = 30.2	p = .00131 Significant
H3	Peer reviews contributes to the success or failure of a software implementation project.	Success = 11.4 Failed = 6.8	p = .0000139 Significant
H4	Training programs contributes to the success or failure of a software implementation project.	Success = 11.1 Failed = 7.8	p = .000709 Significant
H5	Inter-group coordination contributes to the success or failure of a software implementation project.	Success = 13.6 Failed = 12.1	p = .053223 Not Significant
H6	Software project planning contributes to the success or failure of a software implementation project.	Success = 51.7 Failed = 36.9	p = .000000132 Significant
H7	Software project measurement and analysis contributes to the success or failure of a software implementation project.	Success = 50.7 Failed = 30.3	p = .0000163 Significant
H8	Software subcontract management contributes to the success or failure of a software implementation project.	Success = 11.6 Failed = 12.4	p = .708555 Not Significant
H9	Software quality assurance contributes to the success or failure of a software implementation project.	Success = 30.7 Failed = 24.5	p = .095642 Not Significant
H10	Software configuration management contributes to the success or failure of a software implementation project.	Success = 40.6 Failed = 32.0	p = .025291 Significant
H11	Requirements management contributes to the success or failure of a software implementation project.	Success = 51.8 Failed = 38.1	p = .0000247 Significant

The ANOVA procedure was used to test the hypothesis that the means of the two groups are equal. Here the means of the successful projects are compared to the means of the failed projects for each construct. The mean score is shown for the successful and

failed systems. The significance column indicates the resulting p value from the ANOVA test and the significance result. Evidence is provided from this study to indicate that three of the 11 constructs were not found to be significant. The findings indicate that the activities of inter-group coordination, software sub-contract management, and software quality assurance were not emphasized as much as the other seven activities by the successful projects participating in this study. Yet the unsuccessful projects placed as much emphasis on these three activities as they did the other seven. The evidence presented indicates that the participating successful information system implementations placed less emphasis and investment on these three activities. The detailed ANOVA results used to develop Table 10 are located in appendix B.

The twelfth hypothesis states that a set of discriminating activities may be identified to determine the outcome, success or failure, of a software implementation project.

H12: A reduced set of indicators of software implementation project success, provide a better classification of success or failure than the naive prediction rule.

Stepwise discriminant analysis was run against the 67 activities that are presented by the software methodologies discussed in the literature. A stepwise selection is a combination of forward and backward elimination procedures. It begins with no variables in the discrimination function, then either removes or adds a variable at each step. This process is continued until the variables with the most discriminating power are found (Sharma 1996). The Z test results indicate that the null hypothesis is rejected. The results of the stepwise discriminant analysis produced a set of 12 discriminating variables. The variables along with their averaged squared canonical correlation,

indicating strength of discriminating power, are presented in the stepwise rank order in Table 11. Canonical correlations are presented in Table 11 and are seen to increase to .4600 with the 12 items. All 12 variables are significant at .0001 or better. Details of the stepwise discrimination are located in appendix C.

Table 11. Stepwise Discriminant Analysis Results of Hypothesis 12

Question #	Activity	Average Squared Canonical Correlation
17	The software process used allows process tailoring	.2345
51	Peer reviews are conducted, following documented procedures.	.2792
62	Senior management periodically reviews the SQA function and the quality assurance results.	.3213
28	Profiles of software work product size are used to manage the project following documented procedures.	.3457
52	Data defects are collected, recorded, and analyzed during peer reviews and testing.	.3914
4	The organization has a sponsor (champion) who oversees (monitors) the organization's activities for software process improvement.	.4038
47	Unit and integration testing plans are defined and practiced on the project.	.4147
48	System and acceptance testing plans are performed to demonstrate that the product meets requirements.	.4301
69	Software quality assurance and configuration management activities of the software contract are monitored.	.4379
68	Activities for managing the software contract are reviewed with senior management.	.4474
23	The project team conducts internal meetings to track technical progress, plans, performance, and issues/risks against the project plans.	.4508
46	The software code (both new and modifications to existing) is developed, documented, and verified against the requirements.	.4600

The most popular validity and statistical significance indicator of the resulting discriminating variables is the classification rate (Huberty 1994). The test statistic that is

be used to evaluate the statistical and the practical significance of each group's classification rate and the overall classification rate is presented below. The box's M statistic provides the data necessary, o and n, to calculate the classification rate (Z). Box's M is calculated from the linear and quadratic discriminant function performed on the 12 discriminating variables that were identified from the results of the stepwise discriminant analysis procedure. The formula for calculating the classification rate (Z) for each group and overall is described below. The formula for calculating Z_g for the unsuccessful and success groups is defined below where n_g = number of observations in the respective group, successful or unsuccessful.

$$Z_g = \frac{(o_g - e_g)\sqrt{n_g}}{\sqrt{e_g(n_g - e_g)}}$$

$$e_g = n_g^2/n$$

o_g = the number of correct classifications for that group.

The formula for calculating the overall Z (all observations from both successful and unsuccessful projects) is defined below where n = the total number of observations.

$$Z = \frac{(o - e)\sqrt{n}}{\sqrt{e(n - e)}}$$

$$e = \text{Sum}(n_g^2)/n$$

o = the number of overall correct classifications.

The resubstitution linear discriminant function results are as follows.

$Z_{unsuccessful} = (20 - 4.751)\sqrt{165} \; / \; \sqrt{(4.751(165\text{-}4.751))} = 7.09$

$p < .0001$

$Z_{successful} = (128 - 113.75)\sqrt{165} \; / \; \sqrt{(113.75(165\text{-}113.75))} = 2.40$

$p < .008$

$Z_{overall} = (136 - 118.5)\sqrt{165} \; / \; \sqrt{(118.5(165\text{-}118.5))} = 3.03$

$p < .0012$

The cross validation linear discriminant results are as follows.

$Z_{unsuccessful} = (20 - 4.751)\sqrt{165} \; / \; \sqrt{(4.751(165\text{-}4.751))} = 7.09$

$p < .0001$

$Z_{successful} = (129 - 113.75)\sqrt{165} \; / \; \sqrt{(113.75(165\text{-}113.75))} = 2.57$

$p < .005$

$Z_{overall} = (137 - 118.5)\sqrt{165} \; / \; \sqrt{(118.5(165\text{-}118.5))} = 3.20$

$p < .0007$

Both of the linear discriminant functions, resubstitution and cross validation, report significance for all three groups, unsuccessful, successful and overall. The *p* values for unsuccessful, successful and overall are significant. All *p* values are less than

.05. Evidence indicates that the 12 identified variables representing software development activities, are the best set of discriminators between successful and unsuccessful projects. The details of the resubstitution and cross validation procedures are located in appendix D.

Summary

This chapter presented descriptive statistics of the demographic variables and the 11 constructs, along with the results of hypotheses testing and their significance. Twelve discriminating variables were identified and supported as significant discriminators between successful and unsuccessful projects. Three constructs were found not to be significant investments for the successful projects. The following chapter discusses these results.

CHAPTER 5

DISCUSSION

Introduction

This chapter discusses the results of the hypothesis testing and each hypothesis' impact on software implementation projects. Each hypothesis is defined from a practical perspective. The discussion includes the potential impact on businesses performing software implementations as a result of these findings. Initial discussion is on hypotheses 1 through 11, which were tested with ANOVA. The mean ratio is also discussed to indicate the level of emphasis placed on each construct relative to successful and unsuccessful project implementations. Following the ANOVA results, hypothesis 12, which was tested by discriminant analysis, is discussed.

ANOVA tests for the equality of means. In this study the ANOVA procedure was used to test for equality of means between successful and unsuccessful projects for the 11 constructs. In the below discussion, the statement "the hypothesis is significant" indicates that the null hypothesis "the means are equal" was rejected as indicated by the reported test statistic. The statement "the hypothesis is not significant" indicates that the null hypothesis "the means are equal" was failed to be rejected as indicated by the reported test statistic, *p*. This wording allows the practical findings for each hypothesis to be better understood relative to the purpose of this study and application of the results.

Hypothesis 1

Process and quality management contributes to the success or failure of a software implementation project. The intent of process and quality management is to control process performance quantitatively and develop a quantitative understanding of the quality of the projects' products and to achieve the organization's quality goals. This construct is widely deployed within many organizations as a result of management's need to maintain visibility and control performance. Today's emphasis on object oriented techniques suggests reuse of software work products.

This hypothesis is significant (p = .0000215). The mean ratio of successful to unsuccessful projects is 1.6 indicating that the successful projects placed more than half again more emphasis on this construct than did the unsuccessful projects.

Hypothesis 2

Organizational process contributes to the success or failure of a software implementation project. The intent of the organizational process is to establish organizational responsibilities for the software activities that improve the organization's overall process capabilities and maintain the assets across projects. Organizational level processes result in better visibility of resources that may benefit other projects within the organization. Accountability and responsibility are important in ensuring the success of a project. By defining responsibilities of personnel, they may be more effectively directed and managed.

This hypothesis is significant (p = .00131). The mean ratio of successful to unsuccessful projects is 1.34 indicating that successful projects emphasize this construct a third more than did the failed projects.

Hypothesis 3

Peer reviews contributes to the success or failure of a software implementation project. The intent of peer reviews is to remove defects from software work products early and efficiently. Having team members' review another team members code has been practiced for some time. This technique has been useful in identifying defects earlier in the process and resulted in many innovative improvements such as sharing design and coding ideas with others.

This hypothesis is significant (p = .0000139). The ratio of the means between successful and unsuccessful projects is 1.7. The successful project from this study emphasized this task 70 percent more than did the unsuccessful projects.

Hypothesis 4

Training programs contributes to the success or failure of a software implementation project. The intent of training programs is to develop the skills and knowledge of individuals so they can perform their roles effectively and efficiently. Training has been emphasized long before the invention of the computer. Where would society be today without education?

This hypothesis is significant (p = .000709). The mean ratio of successful to unsuccessful projects is 1.4 indicating that the successful projects emphasized this 40 percent more than did the unsuccessful projects.

Hypothesis 5

Inter-group coordination contributes to the success or failure of a software implementation project. The intent of inter-group coordination is to establish a means for the participation among interacting groups. Most projects require interaction from several different groups. Coordination of these activities is quite an undertaking given the number of suppliers and customers with which a given project might interact.

This hypothesis is not significant (p = .053223). The ratio of successful to unsuccessful means is 1.1 indicating that both groups place about the same amount of emphasis on this. This does not indicate that this activity should be avoided; rather project teams may wish to evaluate their current techniques for deploying this task. They may also wish to place more emphasis on other activities that have shown to be contributors to successful project implementations.

Hypothesis 6

Software project planning contributes to the success or failure of a software implementation project. Software project planning involves establishing reasonable plans for performing and managing the software project. Planning is an important part of any

activity, the more variables and diverse activities the more planning that is necessary to ensure success.

This hypothesis tested significant (p = .000000132). The mean ratio of successful to unsuccessful projects is 1.4 indicating that successful projects placed 40 percent more emphasis on this than did the unsuccessful.

Hypothesis 7

Software project measurement and analysis contributes to the success or failure of a software implementation project. Project measurement and analysis of the metrics requires identifying and maintaining indices that provide visibility into progress and take effective actions on deviations from the plans. As discussed in the literature, many metrics used in the software implementation discipline are reactive. By placing emphasis on project measurement and analysis, projects seek out proactive metrics that may provide insight to error and deviations that are correctable.

This hypothesis is significant (p = .0000163). The mean ratio of successful to unsuccessful projects is 1.7, indicating that 70 percent more emphasis is placed on project measurement and analysis by successful projects than by unsuccessful.

Hypothesis 8

Software subcontract management contributes to the success or failure of a software implementation project. Contract management involves selecting and managing qualified software subcontractors effectively. Managing contractors is not unique to

software implementation. Once businesses transitioned from the emphasis on vertical integration to establishing their succinct set of core competencies, contractors have begun to play a more significant role. However, managing that role is challenging.

This hypothesis is not significant (p = .708555). The mean ratio of successful to unsuccessful projects is .93 indicating that the unsuccessful projects placed nearly 10% more emphasis on this activity than did the successful projects. This is the one construct within this study which was emphasized more by the unsuccessful projects than the successful project implementations. Conclusions can not be drawn that contract management is not important. The evidence presented in these findings indicates that the current approach to contract management should be reevaluated and that possibly more emphasis should be placed on other activities.

Hypothesis 9

Software quality assurance contributes to the success or failure of a software implementation project. This activity provides management with visibility into the software process and products being built. SEI (Software Engineering Institute) requires that the software quality assurance function be managed independent of the project. This means that the reporting structure of the software quality analysts be different from those team members on the project. The resulting effect is that the software quality analysts are deemed as police and thus not generally accepted as part of the software project implementation team.

This hypothesis is not significant (p = .095642). The mean ratio of successful to unsuccessful projects is 1.25. The literature states that the software quality analysis function is very important. The evidence from this study does not contradict that statement. Perhaps the way it is being run is the problem. Organizations should revisit this activity and consider new approaches to deploying it.

Hypothesis 10

Software configuration management contributes to the success or failure of a software implementation project. Configuration management's purpose is to establish and maintain the integrity of the products and software project throughout the project's software life cycle. Configuration management has become increasingly more important with the increase in software development support costs. Support is the most costly phase of the software development life cycle (Turban et al. 1996).

This hypothesis is significant (p = .025291). The mean ratio for this activity is 1.3 indicating that the successful projects place a third more emphasis on this than do the unsuccessful projects.

Hypothesis 11

Requirements management contributes to the success or failure of a software implementation project. Managing requirements involves establishing a common understanding between the customer and the software project. The number of business

analysts is an indication of the importance of this activity. Identifying requirements from the customer and successfully managing them is deemed important by the literature.

This hypothesis is significant (p = .0000247). The ratio of means of successful projects to unsuccessful is 1.4 indicating that the successful projects spent 40% more emphasis on this activity.

Hypothesis 12

A reduced set of indicators of software implementation project success, provide a better classification of success or failure than the naive prediction rule. As discussed in the literature, management requires software implementation projects to perform many diverse activities. The impact of this hypothesis is perhaps more practically significant than the previous 11 hypotheses findings. Organizations require software implementation projects to perform numerous tasks, many of which are deemed unnecessary by the project team members. By reducing the list of 67 tasks down to 12 as identified by the discriminant analysis procedure, teams may benefit in the areas of quality, cycle time, budget and resources. The results of this study do not provide evidence that the other 55 tasks are unnecessary or have no impact on the success of a project. Rather these results indicate that the 12 variables identified best discriminate between the successful and the unsuccessful projects participating in this study. There may have been other activities performed by both the successful and unsuccessful projects that were done differently or inadequately that did not make this list. Though the variables that make up each construct intend to represent the performance level in

addition to the emphasis placed on each activity, many other variables are involved such as type of management, skill level, environment, etc. Interaction among the discriminating variables with the other variables may also play a key role in determining the success or failure of a project. Performing certain activities without having done other activities may also impact success. The previously discussed hypotheses, H1 – H11, should assist in addressing this subjectivity from a practical standpoint. Evidence is provided at a significance level ($p < .0007$) that these 12 activities best discriminate between the successful and unsuccessful projects. Each activity should be considered by software implementation projects as an indicator of a successful project.

The discriminating activities are briefly discussed in their ranked order by average squared canonical correlation shown in Table 11. The successful projects indicated that they allowed tailoring to their process whereas the unsuccessful did not indicate that they did this. Project tailoring is very important given that no two projects are identical in scope, personnel, and technology to name just a few differences. The successful projects indicated that peer reviews were conducted following documented procedures. Peer reviews, also a significant finding of hypothesis 3, are the primary vehicle for proactive removal of defects. Senior management periodically reviewing the software quality assurance function and the quality assurance results was identified as a higher priority of the successful projects relative to the unsuccessful projects. Forecasting the profiles of software work product size and managing the project by the profiles following documented procedures were indicated as an activity performed by the successful projects but were not emphasized by the unsuccessful projects.

Another component of peer reviews made the discriminating list. The successful projects collected, recorded, and analyzed defects during peer reviews and testing. This is the third occurrence of successful projects placing emphasis on peer reviews reported from these findings, hypothesis 3, discriminating variable 2 and discriminating variable 5. The participating successful projects have organization sponsors (champions) who oversee (monitor) the organization's activities for software process improvement. The unsuccessful projects did not place as much emphasis on identifying an overseeing sponsor. The successful projects placed more emphasis on unit and integration testing plans and on demonstrating that the product meets the requirements by system and acceptance testing than did the unsuccessful projects. The successful projects indicated that they monitored their software quality assurance and configuration management activities more so than did the unsuccessful projects. Activities for managing the software contract are reviewed with senior management by the successful projects. The unsuccessful projects did not indicate that their project teams conducted internal meetings to track technical progress, plans, performance, issues, and/or risks against the project plans as did the successful projects. The successful projects indicated that their software code (both new and modifications to existing) is developed, documented, and verified against the requirements.

Summary

This chapter provided a discussion of the hypothesis results and compared those activities on which the successful projects placed a higher emphasis than did the unsuccessful projects participating in the study. Discussion was provided on the degree

to which conclusions may be drawn. Some activities were not indicated to be discriminators but may still be important to the success of a project. It may be a matter of how they are defined and deployed by the organization for a specific project. Thus, conclusions that a specific activity is or is not necessary to achieve a successful project should be made with caution.

CHAPTER 6

CONCLUSIONS, LIMITATIONS, AND FUTURE RESEARCH

Conclusions

The purpose of this research was to identify from the currently popular software system implementation activities, what activities impact the success of a software system implementation project and what activities do not. This was accomplished by documenting the current activities recommended by the highly regarded models, collecting performance data on these activities for successful and unsuccessful software system implementation projects, and discriminating between the successful and unsuccessful reported activities. The study focused on 12 hypotheses, 67 software implementation performance activities, one dependent construct, and 11 independent constructs. Evidence was found indicating what activities received more emphasis by successful software system implementation projects than by unsuccessful projects.

Twelve hypotheses were tested resulting in four findings. The findings identified three constructs that received less emphasis from the successful projects than they did from the participating unsuccessful projects. This evidence would indicate that more attention should be given to the other eight constructs and/or a reevaluation should be done of the techniques to define and deploy the three non-significant constructs. The fourth profound finding is the succinct set of software implementation criteria identified by the discriminant analysis procedures.

Assumptions

The development of the independent construct questionnaire is based on the list of highly regarded methodologies discussed in the literature. Some scholars may argue that other sound methodologies have been omitted. However, in that the commonality of the methodologies is the basis of the independent construct's survey content, it is unlikely that the independent construct questionnaire would be greatly modified, even if additional methodologies were included. Mapping of diverse methodologies in the literature supported this (Paulk 1995). More specifically, the general categories (11 constructs) for a software implementation project are similar, regardless of the methodology being used. If changes were to occur they would most likely involve the splitting of a construct into two or combining two constructs into one concise construct.

This research is based on the assumption that the criteria developed from the literature and evaluated by the expert panel contain the "best set" of indicators that contribute to a software project's success or failure. More specifically, the highly regarded software process models already contain the critical success criteria and the expert panel validated the criteria from that composite. The dependent construct assumption is that the success measures selected from the Delone and McLean model (1992) resulted in an accurate measure of a software implementation project's success or failure. This model is widely accepted, as supported in the literature discussed in chapter 2. Data collection assumptions are that the participating companies complied with their

agreement on selecting homogenous internal business system projects, using qualified and knowledgeable personnel to complete the survey, and that their survey responses were as accurate as possible to the best of their knowledge.

Limitations

As with any field research, only partial control is possible and there is limited ability to accommodate extraneous variables (Buckley et al. 1976). Internal validity is not as strong with field research as in the lab. This is based on the acquisition of field data versus data collected from a laboratory experiment. With field data the possibility and risk of extraneous variables and interactions exist. The significance testing indicates that the internal and external validity of this research have no major threats.

Validity threats categorized by Cook and Campbell (1979) are as follows: history, maturation, testing instrumentation, statistical regression, selection, mortality, interactions with selection, ambiguity about the direction of causal influence, diffusion or imitation of treatments, compensatory equalization of treatments, compensatory rivalry, resentful demoralization (Cook and Campbell 1979). History and maturation are not a threat due to the narrow time elapse, six weeks, of data collection. The survey instruments were validated by the expert panel, which reduces the possibility of testing instrumentation validity threats. Statistical threats were addressed and reported by the significance testing appropriate for each test statistic. The remaining threats listed by Cook and Campbell (1979) were addressed by the deployment of a survey and data collection over the internet. The data collected also has high internal validity given that

the data were loaded directly from the participants' survey response electronically into the testing software package, eliminating data entry error. External validity is very strong in that the practical implications of this study may be generalized across organizations implementing homogenous, internal business systems, software.

No assumptions were violated. The content validity of the set of 67 criteria is supported and agreed upon by an expert panel of academic and practitioners very knowledgeable on this topic. Statistical methods are not 100 percent conclusive, but most do contain a high level of objectivity. These results can help quantify uncertainty, but cannot eliminate it (Huberty 1994). The data collected were not as parsimonious as expected. The total sample consisted of 165 observations. The balance of the sample was 28 unsuccessful to 137 successful projects. Even though this balance appears from a face validity perspective to be weak, the Z tests that were discussed in chapter 4 indicate that it produced significant results.

Future Research

A thorough empirical testing of these findings would be valuable from a research and practical perspective. Empirical testing could be carried out by identifying homogenous internal business systems projects willing to apply these results in a controlled environment with emphasis on the eight significant constructs and 12 software project implementation activities. The amount of emphasis each project actually placed on each construct should also be evaluated in more detail.

A more thorough assessment of process models and alternative development methodologies such as RAD, JAD, and prototyping should be done along with a verification of their success.

A true customizable, tailored technique should achieve better results than a single process model solution for each software implementation project. Future research should investigate the creation of a flexible software process model. From these findings a tailorable, more flexible, process model may be deployed that would result in optimal pay back given a solution's investment and life expectancy. Additional research should also include the overall impact of specific technologies. Tools and development environments such as CASE and PowerBuilder should be compared to development with traditional programming languages.

A new paradigm in software design methodologies is object oriented. Software life cycle models used for traditional data centric architectures and designs may or may not be adequate to support the future of object oriented philosophies where software objects and data are independent, encapsulated and portable.

APPENDIX A

PROCESS AND SUCCESS SURVEYS

Instructions

Your responses will be kept anonymous. Only the University of North Texas researchers will ever see individual project results. Only group summaries will be reported to anyone.

There are no right or wrong answers. We just need your honest opinion.

Company Name:
Project Name:

The term "this product" refers to the project software solution being surveyed.

A scale of 0 to 9 will be used. Please select the ONE answer that best describes your response to the statement. Select 0 if the question is not applicable or you don't know the answer. Think of the scale from 1 to 9 as a continuum from total non-compliance to total compliance with the statement provided. For a neutral response, select 5.

0 = This question is not applicable or I don't know the answer.
1 = totally NOT compliant
2 = strongly NOT compliant
3 = NOT compliant
4 = somewhat NOT compliant
5 = neutral (neither compliant or NOT compliant)
6 = somewhat compliant
7 = compliant
8 = strongly compliant
9 = totally compliant

System Satisfaction Questions	**0 1 2 3 4 5 6 7 8 9**
1. This product was completed and delivered on time.	O O O O O O O O O O
2. This product met 100% of your requirements.	O O O O O O O O O O
3. The total cost of this is product was exactly as forecasted.	O O O O O O O O O O
4. The data within this product is accurate and current.	O O O O O O O O O O
5. This product is easy to learn and use.	O O O O O O O O O O
6. This product is very convenient to access.	O O O O O O O O O O
7. This product is very useful.	O O O O O O O O O O
8. This product is very flexible.	O O O O O O O O O O
9. This product is very reliable.	O O O O O O O O O O
10. This product contains a high level of sophistication.	O O O O O O O O O O
11. This product is integrated with other systems as it should be.	O O O O O O O O O O

12. This product has high utilization.	O O O O O O O O O O
13. This product has very acceptable response time.	O O O O O O O O O O
14. The users are extremely satisfied with this product.	O O O O O O O O O O
15. This product has a significant impact on the business.	O O O O O O O O O O

Instructions

Your responses will be kept anonymous. Only the University of North Texas researchers will ever see individual project results. Only group summaries will be reported to anyone.

There are no right or wrong answers. We just need your honest opinion.

1. Company Name:
2. Project Name:
3. What was the total forecasted cost of the project?
4. What was the actual total cost of the project?
5. What were the number of people months forecasted for this project?
6. What was the actual number of people months required to complete this project?
7. What technologies (e.g., CASE) or languages were used to develop/implement this project?

A scale of 0 to 9 will be used. Please select the ONE answer that best describes your response to the statement. Select 0 if the question is not applicable or you don't know the answer. Think of the scale from 1 to 9 as a continuum from total non-compliance to total compliance with the statement provided. For a neutral response, select 5.

0 = This question is not applicable or I don't know the answer.
1 = totally NOT compliant
2 = strongly NOT compliant
3 = NOT compliant
4 = somewhat NOT compliant
5 = neutral (neither compliant or NOT compliant)
6 = somewhat compliant
7 = compliant
8 = strongly compliant
9 = totally compliant

Organization Software Process	**0 1 2 3 4 5 6 7 8 9**
1. Your organization has a written policy that covers coordination of software process activities across the organization.	O O O O O O O O O O
2. Your organization has a group/team responsible for defining the organization's software process activities.	O O O O O O O O O O
3. Your organization has a sponsor (champions) for the organization's activities for software process improvement.	O O O O O O O O O O
4. Your organization has a sponsor (champion) who oversees (monitors) the organization's activities for software process improvement?	O O O O O O O O O O
5. Your software process was thoroughly assessed at the organization level and the project level.	O O O O O O O O O O
6. Your organization has a documented set of standards for the software process it uses and has a group who maintains it.	O O O O O O O O O O

		0 1 2 3 4 5 6 7 8 9
7.	Your project personnel received training and on-going communication of the organization's software process related activities and their roles/responsibilities in the activities.	O O O O O O O O O O
8.	Your organization has a software process database or spreadsheet containing relevant information for each project.	O O O O O O O O O O
9.	Your organization has access to a library which contains software process-related documentation.	O O O O O O O O O O
10.	Your organization has a process or program to introduce, monitor and evaluate new processes, methods, and tools.	O O O O O O O O O O
11.	Your organization has requirements and recommendations for software process related training.	O O O O O O O O O O
12.	Your organization has a policy for implementing SCM (Software Configuration Management).	O O O O O O O O O O
13.	Your organization has a policy for implementing SQA (Software Quality Assurance).	O O O O O O O O O O

0 = This question is <u>not applicable</u> or I <u>don't know</u> the answer.
1 = <u>totally NOT compliant</u>
2 = strongly NOT compliant
3 = NOT compliant
4 = somewhat NOT compliant
5 = <u>neutral</u> (neither compliant or NOT compliant)
6 = somewhat compliant
7 = compliant
8 = strongly compliant
9 = <u>totally compliant</u>

<u>Project Software Process</u>		**0 1 2 3 4 5 6 7 8 9**
14.	Your software process includes a phase for software project planning, documentation and use.	O O O O O O O O O O
15.	Your project team reviews preliminary requirements and then ensures these requirements are the basis for plans, software work products, and activities.	O O O O O O O O O O
16.	Your process ensures the software life cycle plan is followed.	O O O O O O O O O O
17.	Your process allows process tailoring.	O O O O O O O O O O
18.	Your process provides procedures for revising the software project plan and other project plans.	O O O O O O O O O O
19.	Plans for the project's facilities and support tools are prepared and documented.	O O O O O O O O O O
20.	Your process includes a procedure to conduct formal reviews with senior management prior to making or changing project commitments and at selected milestones throughout the project's life cycle.	O O O O O O O O O O

22. Your project periodically reviews accomplishments and plans, and determines the actions to meet the current and forecasted needs of the business, customer, and end users.	O O O O O O O O O O
23. Your project team conducts internal meetings to track technical progress, plans, performance, and issues/risks against the project plans.	O O O O O O O O O O
24. Intergroup commitments and critical dependencies are identified, negotiated, communicated and tracked.	O O O O O O O O O O
25. Software work products that are needed to establish and maintain control of the software project are identified and documented.	O O O O O O O O O O
26. Original and revised estimates for the size of the software work products are derived and recorded following documented procedures.	O O O O O O O O O O
27. Profiles of software work product size (actual versus planned) are documented and updated over time following document procedures.	O O O O O O O O O O
28. Profiles of software work product size are used to manage the project following documented procedures.	O O O O O O O O O O
29. Original and revised estimates for the project's critical computer resources are derived and recorded following documented procedures.	O O O O O O O O O O
30. Profiles of critical computer resource metrics (actual versus planned) are documented and updated over time by following documented procedures.	O O O O O O O O O O
31. Profiles of critical computer resource metrics are used to manage the project by following documented procedures.	O O O O O O O O O O
32. Original and revised estimates for the project's effort (or labor) are derived and recorded following documented procedures.	O O O O O O O O O O
33. Profiles of the project's effort (actual versus planned) are documented and updated over time following documented procedures.	O O O O O O O O O O
34. Profiles of the project's effort are used to manage the project by following documented procedures.	O O O O O O O O O O
35. Original and revised estimates for the software project's cost are derived and recorded following documented procedures.	O O O O O O O O O O
36. Profiles of the project's cost (actual versus planned) are documented and updated over time following documented procedures.	O O O O O O O O O O
37. Profiles of the project's cost are used to manage the project following documented procedures.	O O O O O O O O O O
38. Original and revised estimates for the high level milestones and the detailed schedule are derived and recorded following documented procedures.	O O O O O O O O O O

39.	Profiles of the high level milestones and a detailed schedule (actuals versus planned, by phase) are documented and updated over time following documented procedures.	O O O O O O O O O O
40.	Critical dependencies and critical paths of the milestones and detailed schedule are managed following documented procedures.	O O O O O O O O O O
41.	Risks associated with cost, resource, schedule, and technical aspects of the project are identified, assessed, and tracked.	O O O O O O O O O O
42.	Risks are used to make decisions during the project execution following documented procedures.	O O O O O O O O O O
43.	Detailed software requirements are developed, recorded, verified and reviewed following documented procedures.	O O O O O O O O O O
44.	Changes to requirements are reviewed and incorporated into the software project following documented procedures.	O O O O O O O O O O
45.	The software design (new and changed) is developed, documented, and verified against the requirements.	O O O O O O O O O O
46.	The software code (new and changed) is developed, documented, and verified against the requirements and design.	O O O O O O O O O O
47.	Unit and integration testing plans are defined and practiced on the project.	O O O O O O O O O O
48.	System and acceptance testing plans are performed to demonstrate that the product meets requirements.	O O O O O O O O O O
49.	Project documentation is developed, verified, and maintained (e.g., user documentation and support documentation).	O O O O O O O O O O
50.	Key project documents are maintained for consistency of content.	O O O O O O O O O O
51.	Peer reviews are conducted, following documented procedures.	O O O O O O O O O O
52.	Data defects are collected, recorded, and analyzed during peer reviews and testing.	O O O O O O O O O O

0 = This question is not applicable or I don't know the answer.
1 = totally NOT compliant
2 = strongly NOT compliant
3 = NOT compliant
4 = somewhat NOT compliant
5 = neutral (neither compliant or NOT compliant)
6 = somewhat compliant
7 = compliant
8 = strongly compliant
9 = totally compliant

Software Configuration Management	**0 1 2 3 4 5 6 7 8 9**
54. Resources (people, tools, and methods) are utilized to perform the SCM (software configuration management) function.	O O O O O O O O O O
55. Senior management periodically reviews SCM activities and results.	O O O O O O O O O O
56. Project team members and SCM analysts are educated on the role, responsibilities, authority, and value of SCM.	O O O O O O O O O O
57. Changes to project software work products are managed and controlled.	O O O O O O O O O O
58. Your project has and maintains a configuration management repository.	O O O O O O O O O O
59. Products are created from the baseline repository and their release is controlled.	O O O O O O O O O O
60. Software baseline audits are conducted.	O O O O O O O O O O

0 = This question is not applicable or I don't know the answer.
1 = totally NOT compliant
2 = strongly NOT compliant
3 = NOT compliant
4 = somewhat NOT compliant
5 = neutral (neither compliant or NOT compliant)
6 = somewhat compliant
7 = compliant
8 = strongly compliant
9 = totally compliant

Software Quality Assurance	**0 1 2 3 4 5 6 7 8 9**
61. Your project has an assigned SQA analyst that reports outside of the project leader/manager's organization.	O O O O O O O O O O
62. Senior management periodically reviews the SQA function and the quality assurance results.	O O O O O O O O O O
63. Software project members and SQA analysts are educated on the role, responsibilities, authority, and value of the SQA function.	O O O O O O O O O O
64. Your project has an SQA plan which is monitored.	O O O O O O O O O O
65. The project SQA plan includes a SQA analyst involvement in project process definition, monitoring adherence to the process, and status reporting to the project.	O O O O O O O O O O
66. Your project documents and handles quality issues and deviations are identified in the software work products.	O O O O O O O O O O

0 = This question is not applicable or I don't know the answer.
1 = totally NOT compliant
2 = strongly NOT compliant
3 = NOT compliant
4 = somewhat NOT compliant
5 = neutral (neither compliant or NOT compliant)
6 = somewhat compliant
7 = compliant
8 = strongly compliant
9 = totally compliant

Software Subcontract Management	**0 1 2 3 4 5 6 7 8 9**
67. Your project plans, manages, and reviews subcontractor work.	O O O O O O O O O O
68. Activities for managing the software subcontract are reviewed with senior management.	O O O O O O O O O O
69. Software quality assurance and configuration management (CM) activities of the software subcontract are monitored.	O O O O O O O O O O

APPENDIX B

ANOVA RESULTS OF HYPOTHESES 1 – 11

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
C1	137	5104	37.25547	242.574
C1f	28	653	23.32143	200.1521

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	4513.871	1	4513.871	19.16335	2.14E-05	3.899146
Within Groups	38394.17	163	235.547			
Total	42908.04	164				

Construct: Process Quality Management

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
C2	137	5554	40.54015	223.7649
c2f	28	846	30.21429	272.0265

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	2478.832	1	2478.832	10.69573	0.00131	3.899146
Within Groups	37776.74	163	231.7592			
Total	40255.58	164				

Construct: Organizational Process

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
C3	137	1559	11.37956	24.86958
c3f	28	190	6.785714	22.1746

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	490.6229	1	490.6229	20.08842	1.39E-05	3.899146
Within Groups	3980.977	163	24.42317			
Total	4471.6	164				

Construct: Peer Reviews

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
C4	137	1520	11.09489	20.85122
c4f	28	219	7.821429	21.18915

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	249.1204	1	249.1204	11.91553	0.000709	3.899146
Within Groups	3407.874	163	20.9072			
Total	3656.994	164				

Construct: Training Programs

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
C5	137	1860	13.57664	13.7018
c5f	28	339	12.10714	10.91402

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	50.20345	1	50.20345	3.791794	0.053223	3.899146
Within Groups	2158.124	163	13.24002			
Total	2208.327	164				

Construct: Inter-group Coordination

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
C6	137	7078	51.66423	175.2394
c6f	28	1033	36.89286	122.6177

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	5072.67	1	5072.67	30.46229	1.32E-07	3.899146
Within Groups	27143.23	163	166.5229			
Total	32215.9	164				

Construct: Software Project Planning

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
C7	137	6951	50.73723	526.401
c7f	28	847	30.25	333.1574

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	9758.004	1	9758.004	19.73741	1.63E-05	3.899146
Within Groups	80585.79	163	494.3914			
Total	90343.79	164				

Construct: Software Project Measurement and Analysis

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
C8	137	1592	11.62044	104.7666
c8f	28	347	12.39286	69.50661

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	13.87078	1	13.87078	0.140214	0.708555	3.899146
Within Groups	16124.94	163	98.92602			
Total	16138.81	164				

Construct: Software SubContract Management

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
C9	137	4203	30.67883	321.2343
c9f	28	687	24.53571	267.369

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	877.3489	1	877.3489	2.809208	0.095642	3.899146
Within Groups	50906.83	163	312.3119			
Total	51784.18	164				

Construct: Software Quality Assurance

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
C10	137	5563	40.60584	347.5935
c10f	28	895	31.96429	305.369

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	1736.114	1	1736.114	5.097235	0.025291	3.899146
Within Groups	55517.68	163	340.5993			
Total	57253.79	164				

Construct: Software Configuration Management

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
C11	137	7090	51.75182	239.5409
c11f	28	1068	38.14286	171.9788

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	4305.712	1	4305.712	18.85579	2.47E-05	3.899146
Within Groups	37220.99	163	228.3496			
Total	41526.7	164				

Construct: Requirements Management

APPENDIX C

DISCRIMINANT ANALYSIS STEPWISE RESULTS

DISCRIMINANT ANALYSIS STEPWISE RESULTS

21:03 Thursday, May 20, 1999

Discriminant Analysis

165 Observations	164 DF Total
67 Variables	163 DF Within Classes
2 Classes	1 DF Between Classes

Class Level Information

DV	Frequency	Weight	Proportion	Prior Probability
0	28	28.0000	0.169697	0.169697
1	137	137.0000	0.830303	0.830303

Discriminant Analysis Within Covariance Matrix Information

DV	Covariance Matrix Rank	Natural Log of the Determinant of the Covariance Matrix
0	18	-712.31356
1	67	24.81636
Pooled	67	28.88530

Discriminant Analysis Test of Homogeneity of Within Covariance Matrices

Notation: K = Number of Groups

P = Number of Variables

N = Total Number of Observations - Number of Groups

N(i) = Number of Observations in the i'th Group - 1

$$V = \frac{\prod |\text{Within SS Matrix(i)}|^{N(i)/2}}{|\text{Pooled SS Matrix}|^{N/2}}$$

$$RHO = 1.0 - \left(\sum \frac{1}{N(i)} - \frac{1}{N} \right) \frac{2P^2 + 3P - 1}{6(P+1)(K-1)}$$

$$DF = .5(K-1)P(P+1)$$

Under null hypothesis: $-2\ RHO \ln \left(\frac{N^{PN/2}}{\prod N(i)^{PN(i)/2}} V \right)$

is distributed approximately as chi-square(DF)

Test Chi-Square Value = 2867.889381 with 2278 DF Prob > Chi-Sq = 0.0001

Since the chi-square value is significant at the 0.1 level,
the within covariance matrices will be used in the discriminant function.

Reference: Morrison, D.F. (1976) Multivariate Statistical Methods p252.

Discriminant Analysis Pairwise Generalized Squared Distances Between Groups

$$D^2(i|j) = (\bar{X}_i - \bar{X}_j)' \, COV_j^{-1} \, (\bar{X}_i - \bar{X}_j) + \ln |COV_j| - 2 \ln PRIOR_j$$

Generalized Squared Distance to DV

From DV	0	1
0	-708.76608	40.36620
1	62328784	25.18829

Discriminant Analysis Classification Summary for Calibration Data: WORK.NEW

Resubstitution Summary using Quadratic Discriminant Function

Generalized Squared Distance Function:

$$D_j^2(X) = (X-\bar{X}_j)' \, COV_j^{-1} \, (X-\bar{X}_j) + \ln |COV_j| - 2 \ln PRIOR_j$$

Posterior Probability of Membership in each DV:

$$Pr(j|X) = \exp(-.5 \, D_j^2(X)) \, / \, SUM_k \exp(-.5 \, D_k^2(X))$$

Number of Observations and Percent Classified into DV:

From DV	0	1	Total
0	28	0	28
	100.00	0.00	100.00
1	4	133	137
	2.92	97.08	100.00
Total	32	133	165
Percent	19.39	80.61	100.00
Priors	0.1697	0.8303	

Error Count Estimates for DV:

	0	1	Total
Rate	0.0000	0.0292	0.0242

Priors 0.1697 0.8303

Discriminant Analysis Classification Results for Calibration Data: WORK.NEW

Cross-validation Results using Quadratic Discriminant Function

Generalized Squared Distance Function:

$$D_j^2(X) = (X-\bar{X}_{(X)j})' \, COV_{(X)j}^{-1} \, (X-\bar{X}_{(X)j}) + \ln |COV_{(X)j}| - 2 \ln PRIOR_j$$

Posterior Probability of Membership in each DV:

$$Pr(j|X) = \exp(-.5 \, D_j^2(X)) \, / \, SUM_k \exp(-.5 \, D_k^2(X))$$

Obs	From DV	Classified into DV	Posterior Probability of Membership in DV: 0	1
2	0	1 *	0.0000	1.0000
29	0	1 *	0.0000	1.0000
64	0	1 *	0.0000	1.0000
78	0	1 *	0.0000	1.0000
82	0	1 *	0.0000	1.0000
86	0	1 *	0.0000	1.0000
95	0	1 *	0.0000	1.0000
96	0	1 *	0.0000	1.0000
98	1	0 *	1.0000	0.0000
106	0	1 *	0.0000	1.0000
114	0	1 *	0.0000	1.0000
121	0	1 *	0.0000	1.0000
127	1	0 *	1.0000	0.0000
129	1	0 *	1.0000	0.0000
130	1	0 *	1.0000	0.0000
135	0	1 *	0.0000	1.0000
138	0	1 *	0.0000	1.0000
145	0	1 *	0.0000	1.0000
150	0	1 *	0.0000	1.0000
154	0	1 *	0.0000	1.0000

* Misclassified observation

Discriminant Analysis Classification Summary for Calibration Data: WORK.NEW

Cross-validation Summary using Quadratic Discriminant Function

Generalized Squared Distance Function:

$$D_j^2(X) = (X-\bar{X}_{(X)j})' \, COV_{(X)j}^{-1} \, (X-\bar{X}_{(X)j}) + \ln |COV_{(X)j}| - 2 \ln PRIOR_j$$

Posterior Probability of Membership in each DV:

$$Pr(j|X) = \exp(-.5 \, D_j^2(X)) \, / \, SUM_k \exp(-.5 \, D_k^2(X))$$

Number of Observations and Percent Classified into DV:

From DV	0	1	Total
0	12	16	28
	42.86	57.14	100.00
1	4	133	137
	2.92	97.08	100.00
Total	16	149	165
Percent	9.70	90.30	100.00
Priors	0.1697	0.8303	

Error Count Estimates for DV:

	0	1	Total
Rate	0.5714	0.0292	0.1212
Priors	0.1697	0.8303	

Stepwise Discriminant Analysis

165 Observations 67 Variable(s) in the Analysis
2 Class Levels 0 Variable(s) will be included

The Method for Selecting Variables will be: STEPWISE

Significance Level to Enter = 0.1500
Significance Level to Stay = 0.1500

Class Level Information

DV	Frequency	Weight	Proportion
0	28	28.0000	0.169697
1	137	137.0000	0.830303

Stepwise Discriminant Analysis

Stepwise Selection: Step 1

Statistics for Entry, DF = 1, 163

Variable	R**2	F	Prob > F	Tolerance
I1	0.0887	15.864	0.0001	1.0000
I2	0.0426	7.245	0.0079	1.0000
I3	0.0150	2.476	0.1176	1.0000
I4	0.0110	1.808	0.1806	1.0000
I5	0.0700	12.260	0.0006	1.0000
I6	0.0550	9.488	0.0024	1.0000
I7	0.0493	8.452	0.0042	1.0000
I8	0.0307	5.155	0.0245	1.0000
I9	0.0627	10.896	0.0012	1.0000
I10	0.0319	5.363	0.0218	1.0000
I11	0.0633	11.024	0.0011	1.0000

I12	0.0166	2.744	0.0996	1.0000
I13	0.0696	12.192	0.0006	1.0000
I14	0.0683	11.946	0.0007	1.0000
I15	0.1059	19.298	0.0001	1.0000
I16	0.0963	17.362	0.0001	1.0000
I17	0.2345	49.924	0.0001	1.0000
I18	0.2124	43.950	0.0001	1.0000
I19	0.0322	5.419	0.0211	1.0000
I20	0.0393	6.671	0.0107	1.0000
I22	0.0561	9.684	0.0022	1.0000
I23	0.0130	2.145	0.1450	1.0000
I24	0.0227	3.793	0.0532	1.0000
I25	0.0238	3.973	0.0479	1.0000
I26	0.0365	6.178	0.0139	1.0000
I27	0.0710	12.453	0.0005	1.0000
I28	0.1429	27.182	0.0001	1.0000
I29	0.0465	7.945	0.0054	1.0000
I30	0.0807	14.317	0.0002	1.0000
I31	0.0748	13.185	0.0004	1.0000
I32	0.1503	28.840	0.0001	1.0000
I33	0.0806	14.299	0.0002	1.0000
I34	0.0803	14.227	0.0002	1.0000
I35	0.0779	13.767	0.0003	1.0000
I36	0.0462	7.894	0.0056	1.0000
I37	0.0542	9.344	0.0026	1.0000
I38	0.1068	19.489	0.0001	1.0000
I39	0.1313	24.638	0.0001	1.0000
I40	0.1265	23.599	0.0001	1.0000
I41	0.0121	1.993	0.1599	1.0000
I42	0.0746	13.134	0.0004	1.0000
I43	0.1135	20.872	0.0001	1.0000
I44	0.1028	18.675	0.0001	1.0000
I45	0.1523	29.280	0.0001	1.0000
I46	0.0550	9.494	0.0024	1.0000
I47	0.0173	2.869	0.0922	1.0000
I48	0.0745	13.119	0.0004	1.0000
I49	0.0397	6.746	0.0103	1.0000
I50	0.0476	8.149	0.0049	1.0000

Stepwise Discriminant Analysis

Stepwise Selection: Step 1

Statistics for Entry, DF = 1, 163

Variable	R**2	F	Prob > F	Tolerance
I51	0.1558	30.084	0.0001	1.0000
I52	0.0501	8.601	0.0038	1.0000
I54	0.0069	1.135	0.2883	1.0000
I55	0.1064	19.399	0.0001	1.0000
I56	0.0398	6.756	0.0102	1.0000
I57	0.0149	2.463	0.1185	1.0000
I58	0.0031	0.505	0.4784	1.0000
I59	0.0044	0.723	0.3963	1.0000
I60	0.0227	3.780	0.0536	1.0000
I61	0.0006	0.106	0.7457	1.0000
I62	0.0048	0.784	0.3771	1.0000
I63	0.0065	1.067	0.3031	1.0000
I64	0.0220	3.660	0.0575	1.0000
I65	0.0238	3.979	0.0477	1.0000
I66	0.0088	1.452	0.2300	1.0000
I67	0.0032	0.528	0.4683	1.0000
I68	0.0000	0.006	0.9404	1.0000
I69	0.0005	0.075	0.7849	1.0000

Variable I17 will be entered

The following variable(s) have been entered:
I17

Multivariate Statistics

Wilks' Lambda = 0.76553270 F(1, 163) = 49.924 Prob > F = 0.0001
Pillai's Trace = 0.234467 F(1, 163) = 49.924 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.23446730

Stepwise Discriminant Analysis

Stepwise Selection: Step 2

Statistics for Removal, DF = 1, 163

Variable	R**2	F	Prob > F
I17	0.2345	49.924	0.0001

No variables can be removed

Statistics for Entry, DF = 1, 162

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0055	0.890	0.3469	0.7507
I2	0.0007	0.107	0.7438	0.8532
I3	0.0052	0.853	0.3570	0.8602
I4	0.0052	0.839	0.3609	0.8855
I5	0.0132	2.161	0.1435	0.8760
I6	0.0042	0.678	0.4114	0.8584
I7	0.0019	0.310	0.5786	0.8512
I8	0.0008	0.124	0.7252	0.8342
I9	0.0007	0.120	0.7291	0.7757
I10	0.0001	0.010	0.9200	0.8542
I11	0.0064	1.037	0.3101	0.8503
I12	0.0004	0.060	0.8060	0.9463
I13	0.0056	0.914	0.3404	0.8218
I14	0.0065	1.065	0.3036	0.8350
I15	0.0061	0.992	0.3207	0.6924
I16	0.0000	0.001	0.9733	0.5941
I18	0.0313	5.238	0.0234	0.4601
I19	0.0073	1.186	0.2777	0.9515
I20	0.0002	0.036	0.8501	0.8520
I22	0.0168	2.761	0.0985	0.9306
I23	0.0020	0.321	0.5716	0.9028
I24	0.0012	0.192	0.6617	0.9369
I25	0.0007	0.118	0.7319	0.9262
I26	0.0003	0.053	0.8181	0.8674
I27	0.0041	0.663	0.4166	0.8002
I28	0.0356	5.981	0.0155	0.7677
I29	0.0066	1.075	0.3013	0.9067
I30	0.0020	0.317	0.5742	0.7311
I31	0.0005	0.087	0.7689	0.7196
I32	0.0216	3.578	0.0603	0.6573
I33	0.0000	0.004	0.9479	0.6472
I34	0.0012	0.190	0.6638	0.7161

I35	0.0017	0.274	0.6014	0.7373
I36	0.0001	0.023	0.8799	0.7857
I37	0.0002	0.035	0.8514	0.7910
I38	0.0078	1.272	0.2610	0.7076
I39	0.0101	1.660	0.1994	0.6362

Stepwise Discriminant Analysis

Stepwise Selection: Step 2

Statistics for Entry, DF = 1, 162

Variable	Partial R**2	F	Prob > F	Tolerance
I40	0.0065	1.060	0.3049	0.6154
I41	0.0001	0.016	0.8990	0.9402
I42	0.0025	0.414	0.5209	0.7656
I43	0.0036	0.584	0.4460	0.6279
I44	0.0072	1.178	0.2794	0.7168
I45	0.0232	3.845	0.0516	0.6610
I46	0.0121	1.979	0.1614	0.9131
I47	0.0041	0.673	0.4131	0.8562
I48	0.0059	0.963	0.3279	0.8074
I49	0.0076	1.248	0.2656	0.9329
I50	0.0000	0.000	0.9919	0.7958
I51	0.0584	10.054	0.0018	0.8249
I52	0.0013	0.211	0.6465	0.8379
I54	0.0021	0.346	0.5574	0.9921
I55	0.0163	2.685	0.1032	0.7793
I56	0.0052	0.854	0.3568	0.9178
I57	0.0007	0.107	0.7437	0.9574
I58	0.0032	0.521	0.4713	0.9539
I59	0.0008	0.136	0.7132	0.9644
I60	0.0000	0.000	0.9970	0.9037
I61	0.0112	1.835	0.1774	0.9813
I62	0.0246	4.089	0.0448	0.8380
I63	0.0057	0.926	0.3373	0.9120
I64	0.0006	0.104	0.7470	0.8782
I65	0.0016	0.265	0.6077	0.8509
I66	0.0012	0.196	0.6584	0.9351
I67	0.0225	3.730	0.0552	0.9773
I68	0.0147	2.411	0.1224	0.9591
I69	0.0258	4.296	0.0398	0.9434

Variable I51 will be entered

The following variable(s) have been entered:
I17 I51

Multivariate Statistics

Wilks' Lambda = 0.72079986 F(2, 162) = 31.375 Prob > F = 0.0001
Pillai's Trace = 0.279200 F(2, 162) = 31.375 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.27920014

Stepwise Discriminant Analysis

Stepwise Selection: Step 3

Statistics for Removal, DF = 1, 162

Variable	Partial R**2	F	Prob > F
I17	0.1462	27.732	0.0001
I51	0.0584	10.054	0.0018

No variables can be removed

Statistics for Entry, DF = 1, 161

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0008	0.123	0.7263	0.6263
I2	0.0079	1.274	0.2607	0.6739
I3	0.0233	3.839	0.0518	0.7568
I4	0.0289	4.786	0.0301	0.7275
I5	0.0000	0.000	0.9871	0.6355
I6	0.0113	1.837	0.1772	0.5218
I7	0.0034	0.542	0.4626	0.6965
I8	0.0192	3.160	0.0773	0.6950
I9	0.0052	0.838	0.3615	0.6638
I10	0.0214	3.521	0.0624	0.6333
I11	0.0033	0.536	0.4653	0.5963
I12	0.0003	0.049	0.8257	0.8053
I13	0.0000	0.004	0.9483	0.7327
I14	0.0000	0.000	0.9864	0.7355
I15	0.0002	0.039	0.8445	0.6446
I16	0.0093	1.515	0.2201	0.5124
I18	0.0072	1.170	0.2811	0.3776
I19	0.0006	0.092	0.7614	0.6658
I20	0.0016	0.263	0.6090	0.7593
I22	0.0004	0.069	0.7935	0.6489
I23	0.0197	3.237	0.0739	0.7267
I24	0.0001	0.012	0.9135	0.7992
I25	0.0011	0.184	0.6688	0.7762
I26	0.0058	0.935	0.3351	0.7183
I27	0.0027	0.429	0.5132	0.6388
I28	0.0107	1.746	0.1883	0.6452
I29	0.0001	0.016	0.8983	0.7520
I30	0.0039	0.628	0.4293	0.6077
I31	0.0056	0.899	0.3445	0.6194
I32	0.0043	0.695	0.4056	0.5708
I33	0.0078	1.273	0.2610	0.5826
I34	0.0127	2.072	0.1520	0.5178
I35	0.0044	0.716	0.3987	0.6114
I36	0.0141	2.298	0.1315	0.6666
I37	0.0070	1.134	0.2886	0.6812

Stepwise Discriminant Analysis

Stepwise Selection: Step 3

Statistics for Entry, DF = 1, 161

Variable	Partial R**2	F	Prob > F	Tolerance
I38	0.0000	0.006	0.9363	0.6252
I39	0.0001	0.013	0.9078	0.5430

I40	0.0014	0.228	0.6334	0.4812
I41	0.0070	1.127	0.2899	0.7602
I42	0.0064	1.031	0.3115	0.5831
I43	0.0059	0.955	0.3300	0.4630
I44	0.0019	0.301	0.5842	0.5366
I45	0.0000	0.001	0.9798	0.4040
I46	0.0004	0.061	0.8057	0.7042
I47	0.0326	5.417	0.0212	0.6941
I48	0.0009	0.152	0.6974	0.6586
I49	0.0004	0.065	0.7983	0.6691
I50	0.0167	2.733	0.1002	0.6290
I52	0.0342	5.702	0.0181	0.4344
I54	0.0006	0.095	0.7578	0.7581
I55	0.0014	0.233	0.6302	0.6621
I56	0.0001	0.022	0.8824	0.7716
I57	0.0051	0.832	0.3632	0.7099
I58	0.0056	0.913	0.3406	0.8011
I59	0.0023	0.365	0.5467	0.8079
I60	0.0070	1.135	0.2883	0.7407
I61	0.0492	8.337	0.0044	0.7030
I62	0.0584	9.982	0.0019	0.7595
I63	0.0454	7.649	0.0063	0.6571
I64	0.0216	3.557	0.0611	0.6735
I65	0.0277	4.585	0.0337	0.6691
I66	0.0201	3.309	0.0708	0.7042
I67	0.0168	2.743	0.0996	0.7982
I68	0.0141	2.304	0.1310	0.7938
I69	0.0300	4.987	0.0269	0.7901

Variable I62 will be entered

The following variable(s) have been entered:
I17 I51 I62

Multivariate Statistics

Wilks' Lambda = 0.67871788 F(3, 161) = 25.404 Prob > F = 0.0001
Pillai's Trace = 0.321282 F(3, 161) = 25.404 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.32128212

Stepwise Discriminant Analysis

Stepwise Selection: Step 4

Statistics for Removal, DF = 1, 161

Variable	Partial R**2	F	Prob > F
I17	0.1868	36.975	0.0001
I51	0.0910	16.123	0.0001
I62	0.0584	9.982	0.0019

No variables can be removed

Statistics for Entry, DF = 1, 160

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0010	0.159	0.6909	0.5911

I2	0.0028	0.448	0.5041	0.6478
I3	0.0140	2.271	0.1338	0.7164
I4	0.0188	3.073	0.0815	0.6928
I5	0.0015	0.239	0.6255	0.6147
I6	0.0028	0.451	0.5030	0.5136
I7	0.0001	0.013	0.9096	0.6646
I8	0.0064	1.024	0.3131	0.6533
I9	0.0001	0.019	0.8907	0.6205
I10	0.0135	2.193	0.1406	0.6113
I11	0.0000	0.003	0.9554	0.5745
I12	0.0027	0.426	0.5149	0.7149
I13	0.0127	2.057	0.1535	0.5974
I14	0.0004	0.069	0.7924	0.6915
I15	0.0000	0.000	0.9982	0.6018
I16	0.0000	0.005	0.9426	0.4390
I18	0.0053	0.856	0.3563	0.3763
I19	0.0005	0.078	0.7804	0.6437
I20	0.0001	0.009	0.9240	0.7210
I22	0.0008	0.126	0.7229	0.6113
I23	0.0155	2.518	0.1146	0.6847
I24	0.0001	0.018	0.8947	0.7372
I25	0.0014	0.222	0.6381	0.7097
I26	0.0001	0.008	0.9272	0.6671
I27	0.0010	0.164	0.6862	0.5673
I28	0.0360	5.970	0.0156	0.5877
I29	0.0080	1.296	0.2566	0.7014
I30	0.0014	0.223	0.6372	0.5125
I31	0.0013	0.213	0.6448	0.4995
I32	0.0118	1.907	0.1692	0.5564
I33	0.0000	0.000	0.9949	0.5051
I34	0.0009	0.142	0.7065	0.4527
I35	0.0001	0.017	0.8955	0.5507
I36	0.0013	0.214	0.6446	0.5831

Stepwise Discriminant Analysis

Stepwise Selection: Step 4

Statistics for Entry, DF = 1, 160

Variable	Partial R**2	F	Prob > F	Tolerance
I37	0.0000	0.000	0.9856	0.6022
I38	0.0105	1.701	0.1940	0.5448
I39	0.0101	1.636	0.2027	0.4795
I40	0.0013	0.202	0.6533	0.4401
I41	0.0045	0.715	0.3989	0.7118
I42	0.0011	0.176	0.6754	0.5598
I43	0.0004	0.065	0.7987	0.4365
I44	0.0001	0.017	0.8967	0.5109
I45	0.0000	0.004	0.9497	0.4036
I46	0.0017	0.266	0.6065	0.6658
I47	0.0180	2.939	0.0884	0.6709
I48	0.0001	0.009	0.9236	0.6405
I49	0.0000	0.001	0.9783	0.6348
I50	0.0078	1.252	0.2649	0.6070
I52	0.0302	4.988	0.0269	0.4228
I54	0.0026	0.417	0.5195	0.7039
I55	0.0233	3.817	0.0525	0.5542
I56	0.0099	1.595	0.2085	0.6883
I57	0.0005	0.087	0.7678	0.6826
I58	0.0004	0.065	0.7985	0.6568
I59	0.0060	0.967	0.3268	0.6001
I60	0.0005	0.081	0.7762	0.6303

I61	0.0062	0.999	0.3192	0.4052
I63	0.0034	0.548	0.4603	0.3518
I64	0.0031	0.504	0.4788	0.3074
I65	0.0024	0.392	0.5322	0.2416
I66	0.0005	0.073	0.7876	0.5679
I67	0.0060	0.961	0.3285	0.7293
I68	0.0039	0.620	0.4323	0.7233
I69	0.0122	1.977	0.1617	0.7052

Variable I28 will be entered

The following variable(s) have been entered:
I17 I28 I51 I62

Multivariate Statistics

Wilks' Lambda = 0.65430262 F(4, 160) = 21.134 Prob > F = 0.0001
Pillai's Trace = 0.345697 F(4, 160) = 21.134 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.34569738

Stepwise Discriminant Analysis

Stepwise Selection: Step 5

Statistics for Removal, DF = 1, 160

Variable	Partial R**2	F	Prob > F
I17	0.1493	28.078	0.0001
I28	0.0360	5.970	0.0156
I51	0.0546	9.245	0.0028
I62	0.0824	14.370	0.0002

No variables can be removed

Statistics for Entry, DF = 1, 159

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0009	0.149	0.7002	0.5290
I2	0.0119	1.917	0.1681	0.5462
I3	0.0243	3.967	0.0481	0.5703
I4	0.0235	3.819	0.0524	0.5851
I5	0.0001	0.023	0.8807	0.5470
I6	0.0122	1.959	0.1635	0.4761
I7	0.0050	0.801	0.3722	0.5004
I8	0.0186	3.021	0.0841	0.5474
I9	0.0033	0.532	0.4667	0.5557
I10	0.0185	2.990	0.0857	0.5679
I11	0.0022	0.347	0.5566	0.5366
I12	0.0001	0.015	0.9016	0.5276
I13	0.0026	0.409	0.5235	0.5165
I14	0.0000	0.005	0.9461	0.5838
I15	0.0008	0.125	0.7246	0.5758
I16	0.0093	1.489	0.2241	0.3608
I18	0.0041	0.657	0.4188	0.3753
I19	0.0005	0.085	0.7704	0.5558
I20	0.0004	0.068	0.7944	0.5752
I22	0.0000	0.001	0.9779	0.5737

I23	0.0212	3.446	0.0652	0.5825
I24	0.0056	0.891	0.3467	0.5305
I25	0.0000	0.003	0.9534	0.5594
I26	0.0151	2.440	0.1202	0.4009
I27	0.0208	3.373	0.0681	0.2886
I29	0.0002	0.027	0.8686	0.4225
I30	0.0063	1.014	0.3154	0.3615
I31	0.0101	1.622	0.2046	0.3159
I32	0.0003	0.041	0.8391	0.4173
I33	0.0112	1.796	0.1821	0.3898
I34	0.0372	6.141	0.0143	0.2768
I35	0.0114	1.838	0.1771	0.3985
I36	0.0247	4.028	0.0464	0.4325

Stepwise Discriminant Analysis

Stepwise Selection: Step 5

Statistics for Entry, DF = 1, 159

Variable	Partial R**2	F	Prob > F	Tolerance
I37	0.0149	2.409	0.1226	0.4232
I38	0.0005	0.079	0.7790	0.3283
I39	0.0010	0.155	0.6939	0.2750
I40	0.0025	0.392	0.5324	0.3623
I41	0.0147	2.367	0.1259	0.5496
I42	0.0095	1.532	0.2177	0.5072
I43	0.0085	1.360	0.2452	0.3856
I44	0.0048	0.767	0.3825	0.4351
I45	0.0008	0.122	0.7273	0.3982
I46	0.0002	0.037	0.8479	0.5767
I47	0.0283	4.635	0.0328	0.5740
I48	0.0003	0.047	0.8294	0.5863
I49	0.0004	0.057	0.8110	0.5833
I50	0.0158	2.561	0.1115	0.5693
I52	0.0354	5.833	0.0169	0.4025
I54	0.0006	0.103	0.7488	0.4984
I55	0.0029	0.469	0.4946	0.3709
I56	0.0002	0.028	0.8664	0.4620
I57	0.0019	0.303	0.5829	0.5817
I58	0.0009	0.146	0.7027	0.5485
I59	0.0016	0.255	0.6143	0.5628
I60	0.0012	0.190	0.6632	0.5383
I61	0.0088	1.418	0.2355	0.3927
I63	0.0086	1.378	0.2422	0.3422
I64	0.0001	0.009	0.9243	0.2735
I65	0.0002	0.024	0.8772	0.2168
I66	0.0010	0.159	0.6903	0.5364
I67	0.0105	1.680	0.1967	0.5795
I68	0.0082	1.310	0.2542	0.5768
I69	0.0222	3.613	0.0591	0.5693

Variable I34 will be entered

The following variable(s) have been entered:
I17 I28 I34 I51 I62

Multivariate Statistics

Wilks' Lambda = 0.62997227 F(5, 159) = 18.678 Prob > F = 0.0001
Pillai's Trace = 0.370028 F(5, 159) = 18.678 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.37002773

Stepwise Discriminant Analysis

Stepwise Selection: Step 6

Statistics for Removal, DF = 1, 159

Variable	Partial R**2	F	Prob > F
I17	0.1714	32.899	0.0001
I28	0.0710	12.151	0.0006
I34	0.0372	6.141	0.0143
I51	0.0817	14.150	0.0002
I62	0.0610	10.326	0.0016

No variables can be removed

Statistics for Entry, DF = 1, 158

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0000	0.003	0.9545	0.2717
I2	0.0046	0.730	0.3941	0.2621
I3	0.0121	1.941	0.1655	0.2567
I4	0.0146	2.340	0.1281	0.2667
I5	0.0002	0.031	0.8604	0.2768
I6	0.0033	0.517	0.4733	0.2528
I7	0.0024	0.388	0.5343	0.2730
I8	0.0104	1.667	0.1985	0.2659
I9	0.0019	0.301	0.5840	0.2751
I10	0.0170	2.739	0.0999	0.2763
I11	0.0033	0.517	0.4730	0.2761
I12	0.0001	0.018	0.8925	0.2768
I13	0.0015	0.238	0.6266	0.2756
I14	0.0002	0.033	0.8569	0.2762
I15	0.0005	0.087	0.7687	0.2766
I16	0.0018	0.285	0.5945	0.2530
I18	0.0026	0.413	0.5212	0.2753
I19	0.0001	0.009	0.9264	0.2749
I20	0.0008	0.124	0.7255	0.2765
I22	0.0000	0.000	0.9996	0.2768
I23	0.0147	2.350	0.1273	0.2709
I24	0.0015	0.241	0.6245	0.2665
I25	0.0003	0.045	0.8314	0.2735
I26	0.0048	0.762	0.3842	0.2516
I27	0.0057	0.909	0.3419	0.2339
I29	0.0026	0.406	0.5248	0.2497
I30	0.0003	0.046	0.8313	0.2464
I31	0.0012	0.182	0.6699	0.2409
I32	0.0040	0.637	0.4259	0.2621
I33	0.0003	0.050	0.8225	0.2163
I35	0.0000	0.000	0.9892	0.1931
I36	0.0030	0.478	0.4903	0.1807

Stepwise Discriminant Analysis

Stepwise Selection: Step 6

Statistics for Entry, DF = 1, 158

Variable	Partial R**2	F	Prob > F	Tolerance
I37	0.0016	0.258	0.6125	0.2207
I38	0.0013	0.212	0.6460	0.2529
I39	0.0002	0.039	0.8437	0.2592
I40	0.0005	0.079	0.7789	0.2403
I41	0.0059	0.944	0.3328	0.2597
I42	0.0015	0.244	0.6222	0.2490
I43	0.0039	0.619	0.4325	0.2693
I44	0.0012	0.197	0.6574	0.2676
I45	0.0008	0.126	0.7233	0.2768
I46	0.0000	0.000	0.9911	0.2753
I47	0.0288	4.683	0.0320	0.2768
I48	0.0009	0.145	0.7039	0.2756
I49	0.0000	0.000	0.9824	0.2746
I50	0.0102	1.628	0.2039	0.2710
I52	0.0339	5.543	0.0198	0.2764
I54	0.0009	0.149	0.7004	0.2549
I55	0.0033	0.528	0.4684	0.2767
I56	0.0011	0.170	0.6804	0.2742
I57	0.0016	0.260	0.6106	0.2767
I58	0.0000	0.000	0.9932	0.2702
I59	0.0014	0.219	0.6406	0.2767
I60	0.0001	0.009	0.9243	0.2638
I61	0.0056	0.885	0.3482	0.2734
I63	0.0066	1.052	0.3066	0.2755
I64	0.0002	0.025	0.8754	0.2706
I65	0.0001	0.023	0.8804	0.2134
I66	0.0013	0.201	0.6543	0.2767
I67	0.0057	0.899	0.3446	0.2704
I68	0.0040	0.635	0.4268	0.2705
I69	0.0123	1.976	0.1618	0.2632

Variable I52 will be entered

The following variable(s) have been entered:

I17 I28 I34 I51 I52 I62

Multivariate Statistics

Wilks' Lambda = 0.60862034 F(6, 158) = 16.934 Prob > F = 0.0001
Pillai's Trace = 0.391380 F(6, 158) = 16.934 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.39137966

Stepwise Discriminant Analysis

Stepwise Selection: Step 7

Statistics for Removal, DF = 1, 158

Variable	Partial R**2	F	Prob > F
I17	0.1868	36.285	0.0001
I28	0.0750	12.811	0.0005
I34	0.0357	5.848	0.0167
I51	0.1119	19.910	0.0001
I52	0.0339	5.543	0.0198
I62	0.0596	10.009	0.0019

No variables can be removed

Statistics for Entry, DF = 1, 157

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0023	0.362	0.5484	0.2703
I2	0.0081	1.278	0.2600	0.2611
I3	0.0113	1.789	0.1829	0.2565
I4	0.0204	3.271	0.0724	0.2658
I5	0.0001	0.014	0.9065	0.2763
I6	0.0042	0.661	0.4175	0.2522
I7	0.0030	0.476	0.4911	0.2725
I8	0.0047	0.748	0.3885	0.2659
I9	0.0020	0.314	0.5758	0.2747
I10	0.0086	1.354	0.2463	0.2761
I11	0.0012	0.190	0.6631	0.2756
I12	0.0002	0.032	0.8575	0.2764
I13	0.0001	0.017	0.8972	0.2754
I14	0.0004	0.058	0.8094	0.2758
I15	0.0000	0.001	0.9717	0.2762
I16	0.0029	0.462	0.4977	0.2522
I18	0.0059	0.933	0.3354	0.2746
I19	0.0003	0.047	0.8285	0.2747
I20	0.0008	0.124	0.7250	0.2760
I22	0.0000	0.000	0.9910	0.2763
I23	0.0088	1.397	0.2389	0.2708
I24	0.0008	0.133	0.7153	0.2663
I25	0.0008	0.122	0.7274	0.2732
I26	0.0087	1.385	0.2410	0.2502
I27	0.0055	0.867	0.3532	0.2337
I29	0.0030	0.477	0.4910	0.2494
I30	0.0013	0.198	0.6572	0.2452
I31	0.0010	0.155	0.6944	0.2407
I32	0.0034	0.539	0.4641	0.2616
I33	0.0011	0.174	0.6768	0.2151
I35	0.0000	0.000	0.9826	0.1929

Stepwise Discriminant Analysis

Stepwise Selection: Step 7

Statistics for Entry, DF = 1, 157

Variable	Partial R**2	F	Prob > F	Tolerance
I36	0.0023	0.357	0.5510	0.1806
I37	0.0003	0.046	0.8296	0.2206
I38	0.0011	0.172	0.6788	0.2524
I39	0.0000	0.000	0.9871	0.2576
I40	0.0041	0.643	0.4237	0.2399
I41	0.0031	0.487	0.4863	0.2597
I42	0.0001	0.023	0.8801	0.2489
I43	0.0040	0.635	0.4269	0.2689
I44	0.0011	0.167	0.6833	0.2672
I45	0.0000	0.000	0.9985	0.2764
I46	0.0010	0.152	0.6969	0.2745
I47	0.0171	2.737	0.1000	0.2764
I48	0.0001	0.021	0.8842	0.2748

I49	0.0000	0.000	0.9893	0.2742
I50	0.0092	1.460	0.2288	0.2707
I54	0.0003	0.045	0.8314	0.2540
I55	0.0014	0.213	0.6449	0.2762
I56	0.0000	0.007	0.9357	0.2734
I57	0.0042	0.661	0.4176	0.2762
I58	0.0003	0.053	0.8187	0.2695
I59	0.0000	0.004	0.9522	0.2763
I60	0.0008	0.121	0.7284	0.2621
I61	0.0053	0.843	0.3599	0.2731
I63	0.0029	0.450	0.5035	0.2753
I64	0.0009	0.141	0.7083	0.2683
I65	0.0008	0.121	0.7289	0.2120
I66	0.0000	0.007	0.9323	0.2762
I67	0.0037	0.590	0.4435	0.2702
I68	0.0013	0.199	0.6561	0.2705
I69	0.0086	1.365	0.2445	0.2632

Variable I4 will be entered

The following variable(s) have been entered:

I4 I17 I28 I34 I51 I52 I62

Multivariate Statistics

Wilks' Lambda = 0.59619936 F(7, 157) = 15.191 Prob > F = 0.0001
Pillai's Trace = 0.403801 F(7, 157) = 15.191 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.40380064

Stepwise Discriminant Analysis

Stepwise Selection: Step 8

Statistics for Removal, DF = 1, 157

Variable	Partial R**2	F	Prob > F
I4	0.0204	3.271	0.0724
I17	0.1974	38.610	0.0001
I28	0.0715	12.097	0.0007
I34	0.0257	4.141	0.0435
I51	0.1267	22.788	0.0001
I52	0.0396	6.473	0.0119
I62	0.0543	9.008	0.0031

No variables can be removed

Statistics for Entry, DF = 1, 156

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0000	0.002	0.9660	0.2640
I2	0.0008	0.129	0.7201	0.2586
I3	0.0007	0.113	0.7377	0.1997
I5	0.0029	0.450	0.5034	0.2643
I6	0.0000	0.006	0.9387	0.2515
I7	0.0001	0.013	0.9089	0.2650
I8	0.0019	0.297	0.5864	0.2591
I9	0.0008	0.120	0.7298	0.2650

I10	0.0034	0.533	0.4665	0.2657
I11	0.0000	0.002	0.9644	0.2627
I12	0.0001	0.021	0.8842	0.2658
I13	0.0015	0.239	0.6258	0.2632
I14	0.0001	0.020	0.8879	0.2638
I15	0.0010	0.149	0.7003	0.2649
I16	0.0038	0.587	0.4446	0.2421
I18	0.0112	1.771	0.1852	0.2623
I19	0.0000	0.000	0.9870	0.2632
I20	0.0046	0.726	0.3955	0.2657
I22	0.0001	0.015	0.9019	0.2658
I23	0.0121	1.914	0.1685	0.2591
I24	0.0024	0.381	0.5378	0.2537
I25	0.0002	0.036	0.8496	0.2617
I26	0.0120	1.896	0.1705	0.2385
I27	0.0063	0.991	0.3210	0.2250
I29	0.0023	0.364	0.5470	0.2393
I30	0.0012	0.188	0.6654	0.2371
I31	0.0005	0.079	0.7786	0.2344
I32	0.0029	0.446	0.5052	0.2512
I33	0.0011	0.167	0.6833	0.2089
I35	0.0008	0.132	0.7165	0.1923

Stepwise Discriminant Analysis

Stepwise Selection: Step 8

Statistics for Entry, DF = 1, 156

Variable	Partial R**2	F	Prob > F	Tolerance
I36	0.0006	0.100	0.7518	0.1798
I37	0.0000	0.000	0.9862	0.2174
I38	0.0019	0.297	0.5864	0.2453
I39	0.0000	0.000	0.9836	0.2502
I40	0.0026	0.402	0.5269	0.2286
I41	0.0042	0.658	0.4184	0.2489
I42	0.0005	0.080	0.7781	0.2455
I43	0.0034	0.533	0.4663	0.2595
I44	0.0002	0.024	0.8776	0.2596
I45	0.0008	0.120	0.7295	0.2653
I46	0.0003	0.051	0.8215	0.2648
I47	0.0183	2.902	0.0905	0.2658
I48	0.0000	0.000	0.9859	0.2649
I49	0.0008	0.130	0.7186	0.2616
I50	0.0145	2.294	0.1319	0.2580
I54	0.0001	0.012	0.9112	0.2434
I55	0.0008	0.118	0.7315	0.2655
I56	0.0000	0.002	0.9611	0.2622
I57	0.0045	0.709	0.4009	0.2656
I58	0.0008	0.119	0.7303	0.2584
I59	0.0001	0.013	0.9111	0.2658
I60	0.0021	0.323	0.5707	0.2497
I61	0.0003	0.046	0.8296	0.2655
I63	0.0013	0.205	0.6511	0.2654
I64	0.0008	0.123	0.7264	0.2631
I65	0.0007	0.102	0.7493	0.2119
I66	0.0008	0.127	0.7217	0.2658
I67	0.0053	0.833	0.3628	0.2589
I68	0.0024	0.379	0.5393	0.2589
I69	0.0099	1.559	0.2136	0.2527

Variable I47 will be entered

The following variable(s) have been entered:

I4 I17 I28 I34 I47 I51 I52 I62

Multivariate Statistics

Wilks' Lambda = 0.58531072 F(8, 156) = 13.816 Prob > F = 0.0001
Pillai's Trace = 0.414689 F(8, 156) = 13.816 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.41468928

Stepwise Discriminant Analysis

Stepwise Selection: Step 9

Statistics for Removal, DF = 1, 156

Variable	Partial R**2	F	Prob > F
I4	0.0215	3.433	0.0658
I17	0.2067	40.650	0.0001
I28	0.0794	13.450	0.0003
I34	0.0260	4.165	0.0430
I47	0.0183	2.902	0.0905
I51	0.1342	24.176	0.0001
I52	0.0268	4.302	0.0397
I62	0.0449	7.330	0.0075

No variables can be removed

Statistics for Entry, DF = 1, 155

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0002	0.025	0.8756	0.2640
I2	0.0032	0.495	0.4828	0.2582
I3	0.0003	0.045	0.8328	0.1986
I5	0.0035	0.545	0.4613	0.2643
I6	0.0002	0.031	0.8602	0.2515
I7	0.0000	0.006	0.9397	0.2650
I8	0.0009	0.133	0.7154	0.2590
I9	0.0000	0.000	0.9922	0.2650
I10	0.0019	0.300	0.5844	0.2657
I11	0.0012	0.185	0.6680	0.2625
I12	0.0005	0.076	0.7832	0.2658
I13	0.0018	0.280	0.5975	0.2632
I14	0.0008	0.122	0.7276	0.2637
I15	0.0005	0.085	0.7716	0.2647
I16	0.0020	0.304	0.5822	0.2418
I18	0.0143	2.245	0.1361	0.2622
I19	0.0008	0.124	0.7252	0.2631
I20	0.0017	0.258	0.6119	0.2657
I22	0.0008	0.126	0.7228	0.2658
I23	0.0046	0.715	0.3993	0.2583
I24	0.0010	0.162	0.6882	0.2535
I25	0.0018	0.279	0.5984	0.2616
I26	0.0070	1.097	0.2966	0.2375
I27	0.0027	0.419	0.5183	0.2235
I29	0.0070	1.089	0.2984	0.2380
I30	0.0004	0.068	0.7951	0.2369
I31	0.0005	0.076	0.7826	0.2344

I32	0.0073	1.144	0.2864	0.2506
I33	0.0007	0.105	0.7463	0.2088

Stepwise Discriminant Analysis

Stepwise Selection: Step 9

Statistics for Entry, DF = 1, 155

Variable	Partial R**2	F	Prob > F	Tolerance
I35	0.0021	0.319	0.5728	0.1917
I36	0.0007	0.113	0.7376	0.1798
I37	0.0000	0.000	0.9840	0.2174
I38	0.0037	0.575	0.4494	0.2451
I39	0.0002	0.029	0.8644	0.2500
I40	0.0061	0.950	0.3312	0.2276
I41	0.0000	0.006	0.9369	0.2455
I42	0.0038	0.589	0.4439	0.2442
I43	0.0006	0.099	0.7531	0.2592
I44	0.0005	0.072	0.7883	0.2593
I45	0.0052	0.812	0.3689	0.2652
I46	0.0116	1.818	0.1796	0.2643
I48	0.0264	4.197	0.0422	0.2484
I49	0.0031	0.488	0.4860	0.2599
I50	0.0060	0.939	0.3341	0.2570
I54	0.0009	0.136	0.7131	0.2431
I55	0.0003	0.040	0.8419	0.2655
I56	0.0006	0.097	0.7559	0.2621
I57	0.0000	0.003	0.9532	0.2656
I58	0.0002	0.030	0.8624	0.2578
I59	0.0008	0.122	0.7275	0.2658
I60	0.0005	0.084	0.7724	0.2493
I61	0.0001	0.020	0.8880	0.2655
I63	0.0002	0.035	0.8528	0.2654
I64	0.0007	0.114	0.7365	0.2631
I65	0.0004	0.062	0.8033	0.2115
I66	0.0001	0.008	0.9286	0.2658
I67	0.0045	0.708	0.4014	0.2589
I68	0.0021	0.324	0.5700	0.2589
I69	0.0118	1.845	0.1763	0.2526

Variable I48 will be entered

The following variable(s) have been entered:

I4 I17 I28 I34 I47 I48 I51 I52 I62

Multivariate Statistics

Wilks' Lambda = 0.56987972 F(9, 155) = 12.999 Prob > F = 0.0001
Pillai's Trace = 0.430120 F(9, 155) = 12.999 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.43012028

Stepwise Discriminant Analysis

Stepwise Selection: Step 10

Statistics for Removal, DF = 1, 155

Variable	Partial R**2	F	Prob > F
I4	0.0165	2.608	0.1084
I17	0.1748	32.831	0.0001
I28	0.0834	14.100	0.0002
I34	0.0219	3.467	0.0645
I47	0.0441	7.158	0.0083
I48	0.0264	4.197	0.0422
I51	0.1123	19.609	0.0001
I52	0.0299	4.784	0.0302
I62	0.0427	6.913	0.0094

No variables can be removed

Statistics for Entry, DF = 1, 154

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0004	0.064	0.8002	0.2479
I2	0.0019	0.297	0.5865	0.2468
I3	0.0016	0.241	0.6239	0.1951
I5	0.0034	0.525	0.4700	0.2484
I6	0.0001	0.013	0.9105	0.2482
I7	0.0001	0.008	0.9294	0.2484
I8	0.0008	0.123	0.7263	0.2484
I9	0.0000	0.007	0.9334	0.2479
I10	0.0014	0.214	0.6440	0.2479
I11	0.0016	0.248	0.6191	0.2482
I12	0.0001	0.020	0.8876	0.2473
I13	0.0011	0.165	0.6852	0.2474
I14	0.0002	0.034	0.8547	0.2467
I15	0.0000	0.004	0.9483	0.2453
I16	0.0009	0.133	0.7157	0.2409
I18	0.0099	1.541	0.2164	0.2438
I19	0.0009	0.141	0.7076	0.2484
I20	0.0041	0.630	0.4287	0.2440
I22	0.0000	0.005	0.9427	0.2435
I23	0.0091	1.416	0.2360	0.2424
I24	0.0044	0.686	0.4087	0.2387
I25	0.0012	0.186	0.6669	0.2478
I26	0.0087	1.358	0.2456	0.2349
I27	0.0047	0.723	0.3964	0.2201
I29	0.0022	0.345	0.5578	0.2314
I30	0.0018	0.285	0.5941	0.2328
I31	0.0021	0.332	0.5656	0.2301
I32	0.0100	1.561	0.2135	0.2459

Stepwise Discriminant Analysis

Stepwise Selection: Step 10

Statistics for Entry, DF = 1, 154

Variable	Partial R**2	F	Prob > F	Tolerance
I33	0.0042	0.646	0.4228	0.2015
I35	0.0029	0.444	0.5064	0.1911
I36	0.0000	0.001	0.9720	0.1797
I37	0.0006	0.089	0.7655	0.2172

I38	0.0030	0.461	0.4982	0.2426
I39	0.0000	0.000	0.9971	0.2467
I40	0.0066	1.026	0.3126	0.2261
I41	0.0010	0.161	0.6892	0.2411
I42	0.0011	0.172	0.6793	0.2393
I43	0.0048	0.750	0.3878	0.2325
I44	0.0011	0.175	0.6759	0.2219
I45	0.0016	0.243	0.6229	0.2376
I46	0.0067	1.032	0.3114	0.2385
I49	0.0000	0.001	0.9751	0.2164
I50	0.0090	1.401	0.2383	0.2461
I54	0.0001	0.013	0.9110	0.2391
I55	0.0000	0.001	0.9745	0.2452
I56	0.0002	0.031	0.8615	0.2472
I57	0.0002	0.036	0.8497	0.2364
I58	0.0001	0.015	0.9041	0.2432
I59	0.0000	0.000	0.9956	0.2412
I60	0.0014	0.216	0.6428	0.2463
I61	0.0005	0.076	0.7834	0.2474
I63	0.0005	0.073	0.7867	0.2480
I64	0.0002	0.031	0.8613	0.2468
I65	0.0000	0.006	0.9362	0.2060
I66	0.0002	0.031	0.8611	0.2442
I67	0.0056	0.862	0.3547	0.2481
I68	0.0027	0.411	0.5225	0.2482
I69	0.0136	2.123	0.1472	0.2480

Variable I69 will be entered

The following variable(s) have been entered:

I4 I17 I28 I34 I47 I48 I51 I52 I62 I69

Multivariate Statistics

Wilks' Lambda = 0.56213177 F(10, 154) = 11.996 Prob > F = 0.0001
Pillai's Trace = 0.437868 F(10, 154) = 11.996 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.43786823

Stepwise Discriminant Analysis

Stepwise Selection: Step 11

Statistics for Removal, DF = 1, 154

Variable	Partial R**2	F	Prob > F
I4	0.0178	2.792	0.0968
I17	0.1782	33.405	0.0001
I28	0.0848	14.261	0.0002
I34	0.0143	2.228	0.1375
I47	0.0477	7.715	0.0062
I48	0.0282	4.464	0.0362
I51	0.0968	16.496	0.0001
I52	0.0256	4.049	0.0459
I62	0.0331	5.276	0.0230
I69	0.0136	2.123	0.1472

No variables can be removed

Statistics for Entry, DF = 1, 153

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0003	0.053	0.8178	0.2475
I2	0.0030	0.463	0.4971	0.2423
I3	0.0016	0.250	0.6177	0.1951
I5	0.0039	0.599	0.4400	0.2480
I6	0.0000	0.007	0.9346	0.2366
I7	0.0001	0.018	0.8936	0.2480
I8	0.0007	0.112	0.7380	0.2444
I9	0.0000	0.001	0.9719	0.2475
I10	0.0019	0.287	0.5930	0.2476
I11	0.0023	0.352	0.5541	0.2462
I12	0.0004	0.064	0.8010	0.2470
I13	0.0004	0.068	0.7950	0.2469
I14	0.0010	0.155	0.6947	0.2465
I15	0.0008	0.125	0.7238	0.2452
I16	0.0000	0.000	0.9854	0.2357
I18	0.0107	1.659	0.1997	0.2435
I19	0.0042	0.647	0.4225	0.2479
I20	0.0018	0.282	0.5961	0.2440
I22	0.0007	0.111	0.7393	0.2434
I23	0.0065	0.994	0.3203	0.2423
I24	0.0021	0.324	0.5701	0.2387
I25	0.0015	0.224	0.6364	0.2465
I26	0.0041	0.624	0.4306	0.2309
I27	0.0030	0.461	0.4982	0.2145
I29	0.0047	0.718	0.3982	0.2256
I30	0.0010	0.157	0.6921	0.2251
I31	0.0016	0.246	0.6209	0.2216

Stepwise Discriminant Analysis

Stepwise Selection: Step 11

Statistics for Entry, DF = 1, 153

Variable	Partial R**2	F	Prob > F	Tolerance
I32	0.0130	2.009	0.1584	0.2396
I33	0.0040	0.616	0.4337	0.1941
I35	0.0046	0.701	0.4037	0.1875
I36	0.0002	0.026	0.8714	0.1761
I37	0.0043	0.661	0.4175	0.2159
I38	0.0060	0.919	0.3392	0.2354
I39	0.0003	0.050	0.8230	0.2385
I40	0.0127	1.966	0.1629	0.2224
I41	0.0000	0.006	0.9397	0.2369
I42	0.0032	0.493	0.4837	0.2329
I43	0.0048	0.738	0.3915	0.2321
I44	0.0010	0.150	0.6995	0.2217
I45	0.0018	0.279	0.5982	0.2373
I46	0.0147	2.285	0.1327	0.2334
I49	0.0003	0.053	0.8186	0.2162
I50	0.0063	0.965	0.3275	0.2441
I54	0.0023	0.347	0.5570	0.2350
I55	0.0007	0.108	0.7434	0.2451
I56	0.0025	0.386	0.5353	0.2471
I57	0.0003	0.049	0.8258	0.2314
I58	0.0004	0.058	0.8106	0.2432
I59	0.0005	0.080	0.7776	0.2412
I60	0.0000	0.000	0.9875	0.2410
I61	0.0006	0.094	0.7601	0.2470

I63	0.0003	0.038	0.8450	0.2477
I64	0.0008	0.127	0.7220	0.2466
I65	0.0002	0.025	0.8755	0.2006
I66	0.0002	0.030	0.8623	0.2442
I67	0.0028	0.428	0.5140	0.1957
I68	0.0170	2.646	0.1059	0.1368

Variable I68 will be entered

The following variable(s) have been entered:

I4 I17 I28 I34 I47 I48 I51 I52 I62 I68 I69

Multivariate Statistics

Wilks' Lambda = 0.55257588 F(11, 153) = 11.262 Prob > F = 0.0001
Pillai's Trace = 0.447424 F(11, 153) = 11.262 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.44742412

Stepwise Discriminant Analysis

Stepwise Selection: Step 12

Statistics for Removal, DF = 1, 153

Variable	Partial R**2	F	Prob > F
I4	0.0137	2.127	0.1467
I17	0.1769	32.874	0.0001
I28	0.0865	14.485	0.0002
I34	0.0114	1.761	0.1865
I47	0.0549	8.890	0.0033
I48	0.0291	4.582	0.0339
I51	0.1038	17.725	0.0001
I52	0.0289	4.548	0.0346
I62	0.0322	5.094	0.0254
I68	0.0170	2.646	0.1059
I69	0.0278	4.371	0.0382

Variable I34 will be removed

The following variable(s) have been entered:

I4 I17 I28 I47 I48 I51 I52 I62 I68 I69

Multivariate Statistics

Wilks' Lambda = 0.55893414 F(10, 154) = 12.152 Prob > F = 0.0001
Pillai's Trace = 0.441066 F(10, 154) = 12.152 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.44106586

Stepwise Selection: Step 13

Statistics for Removal, DF = 1, 154

Variable	Partial R**2	F	Prob > F
I4	0.0185	2.906	0.0903
I17	0.1682	31.133	0.0001
I28	0.0832	13.969	0.0003

I47	0.0595	9.743	0.0022
I48	0.0326	5.196	0.0240
I51	0.0941	15.990	0.0001
I52	0.0300	4.755	0.0307
I62	0.0373	5.959	0.0158
I68	0.0199	3.122	0.0792
I69	0.0353	5.637	0.0188

No variables can be removed

Stepwise Discriminant Analysis

Stepwise Selection: Step 13

Statistics for Entry, DF = 1, 153

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0000	0.000	0.9966	0.1364
I2	0.0015	0.233	0.6298	0.1346
I3	0.0015	0.228	0.6336	0.1386
I5	0.0068	1.049	0.3073	0.1410
I6	0.0000	0.003	0.9584	0.1373
I7	0.0005	0.080	0.7772	0.1328
I8	0.0030	0.454	0.5015	0.1415
I9	0.0001	0.015	0.9023	0.1417
I10	0.0035	0.530	0.4678	0.1395
I11	0.0042	0.639	0.4253	0.1417
I12	0.0014	0.219	0.6404	0.1395
I13	0.0024	0.375	0.5411	0.1409
I14	0.0000	0.003	0.9580	0.1412
I15	0.0004	0.055	0.8144	0.1414
I16	0.0001	0.017	0.8956	0.1336
I18	0.0125	1.934	0.1664	0.1417
I19	0.0043	0.662	0.4172	0.1384
I20	0.0062	0.952	0.3309	0.1388
I22	0.0001	0.020	0.8871	0.1415
I23	0.0174	2.703	0.1022	0.1395
I24	0.0093	1.438	0.2323	0.1411
I25	0.0001	0.010	0.9206	0.1372
I26	0.0073	1.120	0.2915	0.1358
I27	0.0071	1.100	0.2959	0.1380
I29	0.0014	0.208	0.6492	0.1382
I30	0.0035	0.531	0.4673	0.1393
I31	0.0047	0.720	0.3974	0.1397
I32	0.0030	0.467	0.4953	0.1385
I33	0.0084	1.293	0.2572	0.1364
I34	0.0114	1.761	0.1865	0.1368
I35	0.0002	0.027	0.8706	0.1370
I36	0.0035	0.531	0.4672	0.1413
I37	0.0002	0.023	0.8793	0.1389
I38	0.0026	0.403	0.5265	0.1393
I39	0.0001	0.019	0.8911	0.1411
I40	0.0031	0.481	0.4890	0.1412
I41	0.0018	0.274	0.6016	0.1416
I42	0.0019	0.297	0.5868	0.1342
I43	0.0070	1.072	0.3022	0.1412
I44	0.0017	0.261	0.6103	0.1402
I45	0.0016	0.244	0.6218	0.1416
I46	0.0144	2.240	0.1366	0.1417
I49	0.0005	0.074	0.7854	0.1414
I50	0.0070	1.071	0.3023	0.1378
I54	0.0005	0.076	0.7834	0.1366

I55	0.0006	0.096	0.7568	0.1408
I56	0.0015	0.233	0.6301	0.1401
I57	0.0003	0.045	0.8318	0.1406

Stepwise Discriminant Analysis

Stepwise Selection: Step 13

Statistics for Entry, DF = 1, 153

Variable	Partial R**2	F	Prob > F	Tolerance
I58	0.0000	0.000	0.9863	0.1401
I59	0.0010	0.153	0.6963	0.1398
I60	0.0003	0.042	0.8382	0.1364
I61	0.0002	0.032	0.8587	0.1405
I63	0.0005	0.074	0.7854	0.1304
I64	0.0043	0.667	0.4155	0.1260
I65	0.0013	0.203	0.6528	0.1293
I66	0.0000	0.000	0.9943	0.1417
I67	0.0025	0.384	0.5365	0.0789

Variable I23 will be entered

The following variable(s) have been entered:

I4 I17 I23 I28 I47 I48 I51 I52 I62 I68 I69

Multivariate Statistics

Wilks' Lambda = 0.54922935 F(11, 153) = 11.416 Prob > F = 0.0001
Pillai's Trace = 0.450771 F(11, 153) = 11.416 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.45077065

Stepwise Selection: Step 14

Statistics for Removal, DF = 1, 153

Variable	Partial R**2	F	Prob > F
I4	0.0195	3.044	0.0830
I17	0.1722	31.823	0.0001
I23	0.0174	2.703	0.1022
I28	0.0874	14.657	0.0002
I47	0.0530	8.561	0.0040
I48	0.0392	6.245	0.0135
I51	0.1036	17.675	0.0001
I52	0.0288	4.531	0.0349
I62	0.0392	6.249	0.0135
I68	0.0278	4.374	0.0382
I69	0.0415	6.626	0.0110

No variables can be removed

Stepwise Discriminant Analysis

Stepwise Selection: Step 14

Statistics for Entry, DF = 1, 152

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0004	0.057	0.8124	0.1351
I2	0.0020	0.299	0.5855	0.1318
I3	0.0019	0.294	0.5882	0.1363
I5	0.0054	0.828	0.3642	0.1390
I6	0.0000	0.006	0.9359	0.1351
I7	0.0010	0.153	0.6962	0.1291
I8	0.0014	0.215	0.6439	0.1395
I9	0.0000	0.001	0.9692	0.1395
I10	0.0016	0.242	0.6233	0.1379
I11	0.0054	0.821	0.3663	0.1394
I12	0.0032	0.483	0.4883	0.1367
I13	0.0017	0.261	0.6101	0.1388
I14	0.0043	0.652	0.4205	0.1395
I15	0.0027	0.408	0.5242	0.1387
I16	0.0003	0.043	0.8363	0.1297
I18	0.0147	2.272	0.1338	0.1395
I19	0.0102	1.567	0.2126	0.1346
I20	0.0032	0.492	0.4840	0.1348
I22	0.0087	1.332	0.2503	0.1395
I24	0.0019	0.289	0.5917	0.1395
I25	0.0007	0.109	0.7417	0.1363
I26	0.0032	0.482	0.4886	0.1317
I27	0.0023	0.354	0.5527	0.1337
I29	0.0064	0.985	0.3224	0.1339
I30	0.0004	0.067	0.7956	0.1353
I31	0.0018	0.279	0.5984	0.1365
I32	0.0047	0.725	0.3959	0.1346
I33	0.0039	0.589	0.4438	0.1323
I34	0.0073	1.118	0.2921	0.1334
I35	0.0014	0.217	0.6422	0.1337
I36	0.0018	0.269	0.6045	0.1388
I37	0.0003	0.047	0.8295	0.1366
I38	0.0057	0.864	0.3540	0.1363
I39	0.0000	0.007	0.9340	0.1386
I40	0.0056	0.854	0.3568	0.1387
I41	0.0010	0.148	0.7008	0.1388
I42	0.0026	0.400	0.5281	0.1318
I43	0.0061	0.936	0.3349	0.1389
I44	0.0021	0.326	0.5690	0.1382
I45	0.0018	0.270	0.6042	0.1394
I46	0.0167	2.580	0.1103	0.1394
I49	0.0008	0.126	0.7235	0.1395
I50	0.0033	0.510	0.4764	0.1343
I54	0.0024	0.359	0.5500	0.1331
I55	0.0002	0.037	0.8472	0.1388
I56	0.0020	0.303	0.5828	0.1378
I57	0.0004	0.060	0.8074	0.1383
I58	0.0000	0.006	0.9386	0.1378

Stepwise Discriminant Analysis

Stepwise Selection: Step 14

Statistics for Entry, DF = 1, 152

Variable	Partial R**2	F	Prob > F	Tolerance

I59	0.0013	0.196	0.6584	0.1375
I60	0.0000	0.006	0.9371	0.1338
I61	0.0001	0.011	0.9175	0.1377
I63	0.0009	0.129	0.7195	0.1280
I64	0.0071	1.082	0.2999	0.1227
I65	0.0017	0.262	0.6095	0.1271
I66	0.0002	0.024	0.8762	0.1391
I67	0.0025	0.383	0.5369	0.0769

Variable I46 will be entered

The following variable(s) have been entered:

I4 I17 I23 I28 I46 I47 I48 I51 I52 I62 I68 I69

Multivariate Statistics

Wilks' Lambda = 0.54006244 F(12, 152) = 10.787 Prob > F = 0.0001
Pillai's Trace = 0.459938 F(12, 152) = 10.787 Prob > F = 0.0001

Average Squared Canonical Correlation = 0.45993756

Stepwise Selection: Step 15

Statistics for Removal, DF = 1, 152

Variable	Partial R**2	F	Prob > F
I4	0.0171	2.644	0.1060
I17	0.1784	33.013	0.0001
I23	0.0196	3.042	0.0832
I28	0.0851	14.144	0.0002
I46	0.0167	2.580	0.1103
I47	0.0652	10.607	0.0014
I48	0.0307	4.808	0.0298
I51	0.0916	15.324	0.0001
I52	0.0277	4.338	0.0390
I62	0.0349	5.502	0.0203
I68	0.0238	3.704	0.0562
I69	0.0429	6.819	0.0099

No variables can be removed

Stepwise Discriminant Analysis

Stepwise Selection: Step 15

Statistics for Entry, DF = 1, 151

Variable	Partial R**2	F	Prob > F	Tolerance
I1	0.0005	0.077	0.7812	0.1351
I2	0.0016	0.238	0.6265	0.1307
I3	0.0031	0.477	0.4910	0.1362
I5	0.0030	0.461	0.4981	0.1386
I6	0.0001	0.012	0.9126	0.1342
I7	0.0002	0.024	0.8765	0.1264
I8	0.0032	0.489	0.4855	0.1393
I9	0.0000	0.000	0.9948	0.1393

I10	0.0036	0.550	0.4594	0.1378
I11	0.0046	0.702	0.4035	0.1392
I12	0.0006	0.095	0.7587	0.1358
I13	0.0003	0.048	0.8265	0.1365
I14	0.0001	0.010	0.9217	0.1394
I15	0.0007	0.104	0.7476	0.1387
I16	0.0002	0.028	0.8677	0.1293
I18	0.0106	1.620	0.2050	0.1394
I19	0.0076	1.155	0.2842	0.1346
I20	0.0072	1.092	0.2976	0.1343
I22	0.0068	1.033	0.3112	0.1393
I24	0.0012	0.181	0.6714	0.1378
I25	0.0000	0.003	0.9574	0.1361
I26	0.0046	0.701	0.4039	0.1317
I27	0.0021	0.320	0.5722	0.1336
I29	0.0052	0.795	0.3740	0.1339
I30	0.0001	0.014	0.9064	0.1351
I31	0.0028	0.418	0.5191	0.1365
I32	0.0041	0.628	0.4293	0.1335
I33	0.0030	0.455	0.5008	0.1322
I34	0.0049	0.738	0.3917	0.1332
I35	0.0033	0.500	0.4805	0.1334
I36	0.0013	0.193	0.6613	0.1387
I37	0.0007	0.105	0.7469	0.1365
I38	0.0059	0.894	0.3459	0.1362
I39	0.0000	0.000	0.9936	0.1386
I40	0.0056	0.845	0.3594	0.1386
I41	0.0025	0.375	0.5412	0.1378
I42	0.0033	0.507	0.4774	0.1318
I43	0.0120	1.842	0.1768	0.1385
I44	0.0073	1.117	0.2922	0.1378
I45	0.0013	0.202	0.6540	0.1391
I49	0.0003	0.040	0.8416	0.1391
I50	0.0093	1.420	0.2353	0.1341
I54	0.0006	0.092	0.7626	0.1330
I55	0.0005	0.080	0.7776	0.1387
I56	0.0007	0.111	0.7392	0.1378
I57	0.0000	0.000	0.9963	0.1383
I58	0.0002	0.035	0.8528	0.1378
I59	0.0002	0.026	0.8710	0.1375

Stepwise Discriminant Analysis

Stepwise Selection: Step 15

Statistics for Entry, DF = 1, 151

Variable	Partial R**2	F	Prob > F	Tolerance
I60	0.0003	0.038	0.8452	0.1338
I61	0.0000	0.000	0.9940	0.1362
I63	0.0008	0.128	0.7207	0.1279
I64	0.0067	1.016	0.3150	0.1227
I65	0.0014	0.218	0.6413	0.1271
I66	0.0000	0.000	0.9881	0.1379
I67	0.0041	0.620	0.4323	0.0769

No variables can be entered

No further steps are possible

Stepwise Selection: Summary

Step	Variable Entered	Variable Removed	Number In	Partial R**2	F Statistic	Prob > F	Wilks' Lambda	Prob < Lambda
1	I17		1	0.2345	49.924	0.0001	0.76553270	0.0001
2	I51		2	0.0584	10.054	0.0018	0.72079986	0.0001
3	I62		3	0.0584	9.982	0.0019	0.67871788	0.0001
4	I28		4	0.0360	5.970	0.0156	0.65430262	0.0001
5	I34		5	0.0372	6.141	0.0143	0.62997227	0.0001
6	I52		6	0.0339	5.543	0.0198	0.60862034	0.0001
7	I4		7	0.0204	3.271	0.0724	0.59619936	0.0001
8	I47		8	0.0183	2.902	0.0905	0.58531072	0.0001
9	I48		9	0.0264	4.197	0.0422	0.56987972	0.0001
10	I69		10	0.0136	2.123	0.1472	0.56213177	0.0001
11	I68		11	0.0170	2.646	0.1059	0.55257588	0.0001
12		I34	10	0.0114	1.761	0.1865	0.55893414	0.0001
13	I23		11	0.0174	2.703	0.1022	0.54922935	0.0001
14	I46		12	0.0167	2.580	0.1103	0.54006244	0.0001

Stepwise Discriminant Analysis

Stepwise Selection: Summary

Step	Variable Entered	Variable Removed	Number In	Average Squared Canonical Correlation	Prob > ASCC
1	I17		1	0.23446730	0.0001
2	I51		2	0.27920014	0.0001
3	I62		3	0.32128212	0.0001
4	I28		4	0.34569738	0.0001
5	I34		5	0.37002773	0.0001
6	I52		6	0.39137966	0.0001
7	I4		7	0.40380064	0.0001
8	I47		8	0.41468928	0.0001
9	I48		9	0.43012028	0.0001
10	I69		10	0.43786823	0.0001
11	I68		11	0.44742412	0.0001
12		I34	10	0.44106586	0.0001
13	I23		11	0.45077065	0.0001
14	I46		12	0.45993756	0.0001

APPENDIX D

DISCRIMINANT ANALYSIS RESUBSTITUTION AND CROSS VALIDATION RESULTS

DISCRIMINANT ANALYIS RESUBSTITUTION AND CROSS VALIDATION RESULTS

10:57 Friday, May 21, 1999

Discriminant Analysis

165 Observations	164 DF Total
12 Variables	163 DF Within Classes
2 Classes	1 DF Between Classes

Class Level Information

DV	Frequency	Weight	Proportion	Prior Probability
0	28	28.0000	0.169697	0.169697
1	137	137.0000	0.830303	0.830303

10:57 Friday, May 21, 1999

Discriminant Analysis Pooled Covariance Matrix Information

Covariance Matrix Rank	Natural Log of the Determinant of the Covariance Matrix
12	15.1497651

10:57 Friday, May 21, 1999

Discriminant Analysis Pairwise Generalized Squared Distances Between Groups

$$D^2(i|j) = (\bar{X}_i - \bar{X}_j)' \, COV^{-1} \, (\bar{X}_i - \bar{X}_j) - 2 \ln PRIOR_j$$

Generalized Squared Distance to DV

From DV	0	1
0	3.54748	6.34294
1	9.51849	0.37193

10:57 Friday, May 21, 1999

Discriminant Analysis Linear Discriminant Function

$$Constant = -.5 \, \bar{X}_j' \, COV^{-1} \, \bar{X}_j + \ln PRIOR_j \qquad Coefficient\ Vector = COV^{-1} \, \bar{X}_j$$

	DV 0	DV 1
CONSTANT	-10.91934	-12.49978
I17	0.07424	0.90556
I51	-0.73958	-0.02867
I62	0.00978	-0.28622
I28	-0.44408	0.07920

I52	0.06928	-0.27288
I4	0.66026	0.46205
I47	-0.04142	-0.91267
I48	0.68471	1.32435
I69	0.43994	-0.16292
I68	-0.44137	0.00312
I23	1.48617	1.14976
I46	0.72207	1.05217

Discriminant Analysis Classification Summary for Calibration Data: WORK.NEW

Resubstitution Summary using Linear Discriminant Function

Generalized Squared Distance Function: Posterior Probability of Membership in each DV:

$D_j^2(X) = (X-\bar{X}_j)' COV_j^{-1} (X-\bar{X}_j) - 2 \ln PRIOR_j$ $Pr(j|X) = \exp(-.5 D_j^2(X)) / SUM_k \exp(-.5 D_k^2(X))$

Number of Observations and Percent Classified into DV:

From DV	0	1	Total
0	20	8	28
	71.43	28.57	100.00
1	8	129	137
	5.84	94.16	100.00
Total	28	137	165
Percent	16.97	83.03	100.00
Priors	0.1697	0.8303	

Error Count Estimates for DV:

	0	1	Total
Rate	0.2857	0.0584	0.0970
Priors	0.1697	0.8303	

Discriminant Analysis Classification Results for Calibration Data: WORK.NEW

Cross-validation Results using Linear Discriminant Function

Generalized Squared Distance Function:

$$D_j^2(X) = (X-\bar{X}_{(X)j})' COV_{(X)j}^{-1} (X-\bar{X}_{(X)j}) - 2 \ln PRIOR_j$$

Posterior Probability of Membership in each DV:

$$Pr(j|X) = \exp(-.5 D_j^2(X)) / SUM_k \exp(-.5 D_k^2(X))$$

Obs	From DV	Classified into DV	Posterior Probability of Membership in DV: 0	1
16	1	0 *	0.9357	0.0643
17	1	0 *	0.8472	0.1528
18	1	0 *	0.9459	0.0541
82	0	1 *	0.3276	0.6724
86	0	1 *	0.0912	0.9088
95	0	1 *	0.2900	0.7100
98	1	0 *	0.9191	0.0809
114	0	1 *	0.0191	0.9809
127	1	0 *	0.9191	0.0809
129	1	0 *	0.9191	0.0809
130	1	0 *	0.9191	0.0809
135	0	1 *	0.0812	0.9188
145	0	1 *	0.0787	0.9213
150	0	1 *	0.1188	0.8812
151	1	0 *	0.8058	0.1942
154	0	1 *	0.0108	0.9892
163	1	0 *	0.5350	0.4650

* Misclassified observation

Discriminant Analysis Classification Summary for Calibration Data: WORK.NEW

Cross-validation Summary using Linear Discriminant Function

Generalized Squared Distance Function:

$$D_j^2(X) = (X-\bar{X}_{(X)j})' \, COV_{(X)}^{-1} \, (X-\bar{X}_{(X)j}) - 2 \ln PRIOR_j$$

Posterior Probability of Membership in each DV:

$$Pr(j|X) = \exp(-.5\, D_j^2(X)) \,/\, SUM_k \exp(-.5\, D_k^2(X))$$

Number of Observations and Percent Classified into DV:

From DV	0	1	Total
0	20	8	28
	71.43	28.57	100.00
1	9	128	137
	6.57	93.43	100.00
Total	29	136	165
Percent	17.58	82.42	100.00
Priors	0.1697	0.8303	

Error Count Estimates for DV:

	0	1	Total
Rate	0.2857	0.0657	0.1030
Priors	0.1697	0.8303	

REFERENCES

Alavi, Maryam, and Wetherbe, James C. “Mixing Prototyping and Data Modeling for Information System Design,” IEEE Software 8:3 (May 1991): 86-91.

Bachman, Charles W. “Data Structure Diagrams,” Data Base 1:2 (Summer 1969): 4-10.

Baker, Richard. “The Corporate Politics of CMM Ratings,” Communications of the ACM 39:9 (September 1996): 105-106.

Basil, Victor, Frank McGarry, Rose Pajerski, Sharon Waligora, and Marvin Zelkowitz. “SEL’s Software Process-Improvement Program,” IEEE Software 12:6 (November 1995): 83-87.

Baxter, I.D. “Design Maintenance Systems,” Communications of the ACM 35:4 (April 1992): 73-89.

Bennets, Peter D. C., Wood-Harper, A. Trevor, and Mills, Stella. “The Soft System Methodology as a Framework for Software Process Improvement,” Journal of End User Computing, 10:1 (Winter 1998): 12-19.

Boehm, Barry W., and Papaccio, Phillip N. “Understanding and Controlling Software Costs,” IEEE Transactions on Software Engineering 14:10 (October 1988): 1462-1477.

Boehm, Barry W. “Anchoring the Software Process,” IEEE Software 13:4 (July 1996): 77-82.

Boehm, B.W. “A Spiral Model of Software Development and Enhancement,” IEEE Computer 21:5 (1988): 61-72.

Boehm, B.W. “Software Engineering,” IEEE Transactions of Computing C25 (1976): 1226-1241.

Brown, Mark Graham. Baldrige Award Winning Quality, Sixth Edition, How to Interpret the Malcolm Baldrige Award Criteria, Quality Resources, ASQC Quality Press (1996) vii, 113-131.

Buckley, John W., Buckley, Marlene H., and Chiang, Hung-Fu. Research Methodology & Business Decisions, New York: National Association of Accountants and The Society of Industrial Accountants of Canada, 1976.

Cook, Thomas D., and Campbell, Donald T. Quasi-Experimentation Design & Analysis Issues for Field Settings, Houghton Mifflin Company, Boston, 1979.

Crosby, Philip B. Quality is Free. The Art of Making Quality Certain, McGraw-Hill Book Company, 1979.

Dansky, Kathryn H., and Brannon, Diane. “Discriminant Analysis: A Techniques of Adding Value to Patient Satisfaction Surveys.” Hospital & Health Services Administration 41:4 (Winter 1996): 503-513.

Davis, Alan M., Bershoff, Edward H., and Comer, Edward R. "A Strategy for Comparing Alternative Software Development Life Cycle Models," IEEE Transactions on Software Engineering 14:10 (October 1988): 1453-1461.

DeLone, William H., and McLean, Ephraim R. “Information Systems Success: The Quest for the Dependent Variable,” Information Systems Research 3:1 (March 1992): 60-95.

Ebert, Christof. “Technical Controlling in Software Development,” International Journal of Project Management 17:1, (February 1999): 17-29.

Hammer, M., and Champy, J. Re-engineering the Corporation, New York: Harper Business, 1993.

Hayes, Michael H. “ISO9000: The New Strategic Consideration,” Business Horizons 37:3 (May-June 1994): 52-60.

Herbsleb, James, Zubrow, David, Goldenson, Dennis, Hayes, Will and Paulk, Mark. “Software Quality and The Capability Maturity Model,” Communications of the ACM, 40:6 (June 1997): 30-40.

Hoffnagle, G. F., and Beregi, W.E. "Automating the Software Development Process," IBM Systems Journal 24:2 (June 1985): 102-120.

Hollenbach, Craig, Young, Ralph, Pflugrad, Al, and Smith, Doug. “Combining Quality and Software Improvement,” Communications of the ACM, 40:6, (June 1997): 41-45.

Huberty, Carl J. Applied Discrminant Analysis, John Wiley & Sons, Inc., New York, NY, 1994.

Humphrey, Watts S. “Using a Defined and Measured Personal Software Process,” IEEE Software 13:3 (May 1996): 77-88.

Ives, Blake, Hamilton, Scott, and Davis, Gordon B. “A Framework for Research in Computer-Based Management Information Systems,” Management Science 26:9 (September 1980): 910-934.

Ives, Blake, and Olson, Margrethe, “User Involvement and MIS Success: A Review of Research,” Management Science, 30:5 (May 1984): 586-603.

Janson, Marius A., and Smith, L. Douglas. "Prototyping for Systems Development: A Critical Appraisal", MIS Quarterly 9:4 (December 1985): 305-316.

Johnson, N. and Wichern, D. Applied Multivariate Statistical Analysis, Prentice-Hall, Englewood Cliffs, N.J. 1988.

Jones, Capers. Software Quality Analysis and Guidelines for Success, International Thomson Computer Press, 1997.

Jones, Capers. "The Economics of Software Process Improvement," Computer 29:1 (January 1996): 95-97.

Jones, Capers. "Assessing the Software Engineering Institute: Gaps in the SEI Programs," Software Productivity Research, Burlington, MA (1994): 1-6.

Kemerer, Chris F., and Porter, Benjamin S. “Improving the Reliability of Function Point Measurement: An Empirical Study,” IEEE Transactions on Software Engineering 18:11 (November 1992): 1011-1024.

Khoshgoftarr, Taghi M., Allen, Edward, B. Kalaichelvan, Kalai S., and Goel, Nishith. “Early quality prediction: A Case Study in Telecommunications,” IEEE Software 13:1 (January 1996): 65-71.

Krasner, Herb, Terrel, Jim, Linehan, Adam, Arnold, Paul and Ett, William H. “Lessons Learned from a Software Process Modeling System,” Communications of the ACM 35:9 (September 1992): 91-100.

Laser, Stephen A. "The Race Without a Finish Line: Americas Quest for Total Quality: Lessons from Malcolm Baldrige Award Winners," Personnel Psychology 47:1 (Spring 1994): 211-214.

Manley, J. H. “Computer Aided Software Engineering (CASE) Foundation for Software Engineering,” Proceedings COMPCON, (September 1984): 84-91.

Mason, Richard O. “Measuring Information Output: A Communication Systems Approach,” Information & Management, 1:5 (October 1978): 219-234.

Mitchell, Terence R. “An Evaluation of the Validity of Correlational Research Conducted in Organizations,” Academy of Management Review 10:2 (1985): 192-205.

Myers, Ware. “Improve Your Personal Software Process,” IEEE Software 12:3 (May 1995): 104-105.

Niederman, Fred, Branchequ, James C., and Wetherbe, James C., Information Systems Management Issues, MIS Quarterly (December 1991): 475-500.

Nolan, Norton & Co. <http://www.us.kpmg.com>, <http:usserve.us.kpmg.com/ssc/et> (1996).

Oskarsson, Osten, and Glass, Robert L. An ISO 9000 Approach to Building Quality Software, Prentice Hall PTR, Upper Saddle River, New Jersey, 1996.

Paulk, Mark C. ”How ISO 9001 Compares with the CMM,” IEEE Software 12:1 (January 1995): 74-83.

Paulk, Mark C., Curtis, Bill, Chrissis, Mary Beth, and Weber, Charles V. Capability Maturity Model for Software, Version 1.1, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania. February, 1993.

Pressman, Roger. “Software Process Impediment,” IEEE Software 13:5 (September 1996): 16-17.

Pressman, R.S. and Associates. <http://www.rspa.com> (1996).

Pryor, Alan N. and McGuire, Eugene G. “Tailoring of the Software Engineering Institute’s CMM (Capability Maturity Model),” PICMET, Portland Oregon, July 1997.

Porter, Michael E. The Competitive Advantage of Nations, The Free Press, A Division of Macmillan, Inc. New York, (1990)

Ross, Niall. “Using Metrics in Quality Management,” IEEE Software (July 1990): 80-81,85.

Royce, W.W. “Managing the Development of Large Software Systems: Concepts and Techniques,” Proceedings of IEEE WESTCON Los Angeles, CA, (1970): 1-9.

Saiedian, Hossein and McClanahan, Lauram. “Frameworks for Quality Software Process: SEI Capability Maturity Model versus ISO 9000,” Software Quality Journal, 5, (1996): 1-23.

Shannon, Claude E. and Weaver, Warren. The Mathematical Theory of Communication, University of Illinois Press, Urbana, IL, 1949.

Sharma, Subhash. Applied Multivariate Techniques, John Wiley & Sons, Inc (1996): 237.

Sharp, Walter. Phone interview on 8 November 1996 (972)830-7365, <http://www.gartner.com>.

Shetty, Y. K. "The Quest for Quality Excellence: Lessons from the Malcolm Baldrige Quality Award," Sam Advanced Management Journal 58:2 (Spring 1993): 34-40.

Simon, Herbert A. "Whether Software Engineering Needs To Be Artificially Intelligent," IEEE Transactions on Software Engineering 12:7 (July 1986): 726-732.

Singleton, John P., McLean, Ephraim R., and Altman, Edward N. "Measuring Information Systems Performance: Experience with the Management by Results System at Security Pacific Bank," MIS Quarterly (June 1988): 325-337.

Smith, Adam. An Inquiry into the Nature and Causes of the Wealth of Nations, 1776. New York: The Modern Library, 1937.

Sommerville, Ian. "Software Process Models," ACM Computing Surveys 28:1 (March 1996): 269-271.

Stevens, W. P., Myers, G. J., and Constantine, L.L. "Structured Design." IBM Systems Journal 13:2 (May 1974): 115-139.

Strehlo, Kevin. "Raytheon Lays Down Practices for Developers to Follow," InfoWorld 18:20 (June 24, 1996): 85.

Strehlo, Kevin. "Psst…wanna save some big bucks on development?" InfoWorld 18:25 (June 17, 1996): 96.

Tenner, Arthur R. and Detoro, Irving J. Total Quality Management. Three Steps to Continuous Improvement, Addison-Wesley Publishing Company, Inc., 1992.

Turban, Efraim, McLean, Ephraim and Wetherbe, James. Information Technology For Management, New York: John Wiley & Sons, Inc., 1996.

Wojtkowski, Wita and Wojtkowski, W. Gregory. "A Look at the Status of CASE Technology," in Research Issues In Information Systems: An Agenda for the 1990's Jenkins, et al., editors, Wm. C. Brown, Dubuque, IA, 1990, 173-192.

Zmud, Robert W. "An Empirical Investigation of the Dimensionality of the Concept of Information," Decision Sciences 9:2 (April 1978): 187-195.