

## INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

# U·M·I

University Microfilms International  
A Bell & Howell Information Company  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA  
313/761-4700 800/521-0600



**Order Number 9434234**

**Using maintenance-oriented software engineering tools to  
support software maintenance activities**

**Dishaw, Mark Thomas, D.B.A.**

**Boston University, 1994**

**Copyright ©1994 by Dishaw, Mark Thomas. All rights reserved.**

**U·M·I**  
300 N. Zeeb Rd.  
Ann Arbor, MI 48106



BOSTON UNIVERSITY  
GRADUATE SCHOOL OF MANAGEMENT

Dissertation

USING MAINTENANCE ORIENTED SOFTWARE  
ENGINEERING TOOLS TO SUPPORT SOFTWARE  
MAINTENANCE ACTIVITIES

by

MARK THOMAS DISHAW

B.S., State University of New York at Albany, 1975

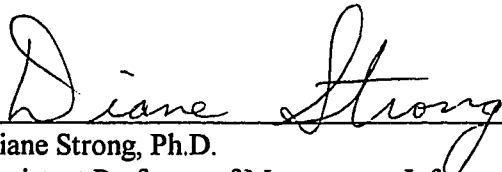
M.B.A., University of Rochester, 1981

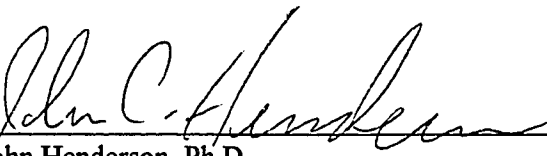
Submitted in partial fulfillment of the  
requirements for the degree of  
Doctor of Business Administration

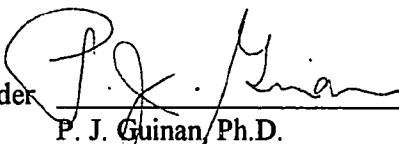
1994

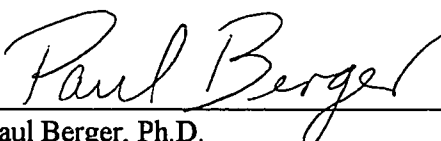
© Mark Thomas Dishaw, 1994

Approved by

First Reader   
Diane Strong, Ph.D.  
Assistant Professor of Management Information Systems

Second Reader   
John Henderson, Ph.D.  
Professor of Management Information Systems

Third Reader   
P. J. Guinan, Ph.D.  
Assistant Professor of Management Information Systems

Fourth Reader   
Paul Berger, Ph.D.  
Professor of Management Science

## Acknowledgments

This dissertation would not have been possible without the support of several individuals.

I would like to thank my advisor, Dr. Diane Strong, for devoting an enormous amount of time to guiding me through the process, and giving exactly the right advice at the right times. Thanks go also to the other members of the committee, Dr. John Henderson, Dr. P.J. Guinan, and Dr. Paul Berger for the reading of many drafts and the return of very helpful comments. I would also like to thank Dr. Jane Fedorowicz and Dr. Mike Lawson for their support during the entire doctoral program.

I would like to acknowledge the financial assistance of the Boston University Systems Research Center, Dr. P.J. Guinan, and Dr. John Henderson for the financial support that allowed this dissertation to be completed.

Finally I'd like to thank my wife, Charlene, without whose support I would have never have undertaken the doctorate. It was been a long trek, and she made it worthwhile. Her incredible patience in listening to my ideas, both good and otherwise, as well as putting up with ridiculous work schedules is appreciated in more ways than can be expressed here.



This dissertation is dedicated to the memory of

Virginia Mary McCaffrey Dishaw

my mother

and

Richard Octave Burns, Ph.D.

my godfather

# USING MAINTENANCE ORIENTED SOFTWARE ENGINEERING TOOLS TO SUPPORT SOFTWARE MAINTENANCE ACTIVITIES

(Order No.        )

MARK THOMAS DISHAW

Boston University, Graduate School of Management, 1994  
Major Professor: Diane Strong, Ph.D., Assistant Professor of MIS

## ABSTRACT

The software maintenance process is the last phase in the software life cycle. Studies have shown that, on average, 70% of MIS budgets are devoted to the maintenance process. There is considerable interest in changing software development and maintenance practices to "free" budget dollars for "development" projects and look for ways of making current practices less expensive. In a number of organizations, MIS management has turned to software engineering tools which are designed to support software maintenance as a potential solution to this problem.

In this dissertation a model based on the Technology Acceptance Model of Davis (1989) and the Task-Technology Fit model of Goodhue (1988, 1992), was developed to explain the factors which lead to the use of this class of software tool. The nature of the fit between software tool functionality and maintenance task demands is examined specifically. A model of the software maintenance process was developed based on the software debugging literature and program understanding literature. A model of software maintenance tool functionality was developed based on the Functional CASE Technology Model of Henderson & Cooperider (1990). New instruments to measure maintenance task characteristics and the functionality of software maintenance support tools were developed.

Data were collected from 36 programmers and their managers on 74 maintenance projects. This sample was drawn from the software development and maintenance organizations of three large firms. The data were analyzed using factor analysis, regression modeling and MANOVA.

The principal finding of the dissertation is that Task - Technology fit is a strong and significant predictor of software maintenance tool use. In addition, it was found that higher task complexity and higher levels of experience with software maintenance tools are associated with higher levels of tool use. Experience with the software being maintained did not result in lower levels of use. Finally, no difference can be demonstrated between debugging and enhancement projects on tool use or task activities.

These results increase our understanding of how programmers actually perform software maintenance. Further, these findings lead to recommendations for the improvement of maintenance support and maintenance software design.

# Table of Contents

<b>Chapter 1</b> .....	<b>1</b>
<b>1. Introduction</b>	
<b>2. The Nature of The Maintenance Problem</b> .....	<b>2</b>
<b>3. Prior Research on Software Maintenance</b> .....	<b>3</b>
3.1 MAINTENANCE PROCESS .....	3
3.2 SOFTWARE MAINTENANCE TOOLS .....	5
<b>4. Tool Assisted Maintenance: Part of the Solution?</b> .....	<b>6</b>
<b>5. Research Goals</b> .....	<b>7</b>
<b>6. Overview of the Dissertation</b> .....	<b>8</b>
<b>Chapter 2</b> .....	<b>9</b>
<b>1. Introduction</b>	
<b>2. Background On The Four Models</b> .....	<b>12</b>
2.1 THE TECHNOLOGY ACCEPTANCE MODEL (TAM) (VARIABLES 1-5) .....	12
2.1.1 <i>The Basis of TAM</i> .....	15
2.1.2 <i>Definition of TAM Model Variables</i> .....	16
2.1.3 <i>External Variables: Attitude Formation</i> .....	17
2.1.4 <i>Alternative Usage Model</i> .....	18
2.2 TASK/TECHNOLOGY FIT MODEL .....	20
2.3 THE MAINTENANCE TASK .....	21
2.3.1 <i>Basic Software Maintenance Activity Model</i> .....	21
2.3.2 <i>Revised Software Maintenance Activity Model</i> .....	26
2.4 MAINTENANCE SUPPORT TECHNOLOGY .....	28
2.4.1 <i>CASE Tool Function Model (FCTM)</i> .....	28
2.4.2 <i>Maintenance Support Function Model (MSFM)</i> .....	31
<b>3. Task - Technology Fit</b> .....	<b>33</b>
3.1 THE DEFINITION OF TASK - TECHNOLOGY FIT .....	33
3.1.1 <i>Dimensions of Maintenance Task and Support Technology Fit</i> .....	37
3.1.2 <i>Fit Moderating Variables</i> .....	41
<b>4. Research Questions</b> .....	<b>42</b>
4.1 TOOL USE QUESTIONS .....	42
4.2 NATURE OF TASK-TECHNOLOGY FIT QUESTIONS .....	45
4.3 NATURE OF MAINTENANCE TASK QUESTIONS .....	48
<b>5. Research Model Summary</b> .....	<b>49</b>
<b>6. Conclusion</b> .....	<b>49</b>
<b>Chapter 3</b> .....	<b>52</b>
<b>1. Introduction</b>	
1.1 RESEARCH SITES .....	52
1.2 PROJECT SELECTION .....	53
1.3 DATA COLLECTION .....	54
<b>2. Operationalization and Measurement of Model Variables</b> .....	<b>56</b>
2.1 TAM VARIABLES .....	57

2.1.1 Perceived Usefulness .....	58
2.1.2 Perceived Ease of Use.....	58
2.1.3 Attitude Towards Tool Use .....	59
2.1.4 Intention To Use Tool.....	59
2.1.5 Actual Tool Use .....	59
2.2 TASK-TECHNOLOGY FIT .....	60
2.3 PERCEIVED FIT .....	61
2.3.1 Specific (Derived) Fit.....	61
2.3.2 General Fit .....	63
2.4 PRINCIPAL FIT VARIABLES .....	64
2.4.1 Maintenance Task.....	64
2.4.2 Maintenance Technology .....	65
2.5 FIT MODERATING VARIABLES .....	66
2.5.1 Experience With Task.....	66
2.5.2 Experience With Software Maintenance Tool .....	66
2.6 CONTROL VARIABLES.....	67
2.6.1 Maintenance Task Type.....	67
2.6.2 Maintenance Complexity.....	68
2.6.3 Demographics.....	68
2.6.4 Social Norms.....	69
<b>3. Development and Validation of Instruments .....</b>	<b>69</b>
3.1 PRE-TEST OF INSTRUMENT(S) .....	70
3.2 THREATS TO VALIDITY & RELIABILITY .....	71
3.3 ITEM & SCALE DEVELOPMENT.....	73
3.3.1 Software Maintenance Activity Model .....	74
3.3.2 Maintenance Tool Functionality.....	78
3.3.3 Moderating Variables .....	80
<b>4. Data Analysis and Hypothesis Testing.....</b>	<b>85</b>
4.1 TOOL USAGE.....	85
4.2 NATURE OF FIT.....	88
4.3 NATURE OF MAINTENANCE TASK.....	90
<b>5. Conclusion.....</b>	<b>91</b>
<b>Chapter 4.....</b>	<b>92</b>
<b>1. Introduction</b>	
<b>2. Tool Use .....</b>	<b>92</b>
2.1 TASK - TECHNOLOGY FIT .....	92
2.1.1 Goodhue Method.....	93
2.1.2 Matching Method.....	95
2.2 EXPERIENCE WITH MAINTENANCE TOOLS.....	98
2.3 EXPERIENCE WITH MAINTENANCE TASK.....	99
2.3.1 General Experience .....	99
2.3.2 Experience With System .....	100
2.4 TASK COMPLEXITY.....	101
2.5 TASK TYPE.....	104
2.5.1 Fit - Task Type Interaction .....	105
2.5.2 Tool Experience - Task Type Interaction .....	107
2.5.3 Task Experience - Task Type Interaction.....	109
2.5.4 System Experience - Task Type Interaction .....	109

2.5.5 Task Complexity- Task Type Interaction.....	111
2.6 FULL MODEL (MAIN EFFECTS ONLY).....	112
2.7 FULL MODEL (TEST OF EFFECT OF NON-INDEPENDENCE OF CASES) .....	115
<b>3. Maintenance Support .....</b>	<b>116</b>
3.1 UNDERSTANDING .....	117
3.2 MODIFICATION.....	118
3.3 COORDINATION.....	118
3.4 "MINOR" SUPPORT RELATIONSHIPS.....	119
<b>4. Nature of Maintenance Task.....</b>	<b>120</b>
<b>5. Summary of Results .....</b>	<b>122</b>
<b>Chapter 5.....</b>	<b>124</b>
<b>1. Introduction</b>	
<b>2. Research Questions .....</b>	<b>124</b>
2.1 TASK-TECHNOLOGY FIT .....	124
2.1.1 Experience / Expertise .....	129
2.1.2 Task Complexity.....	131
2.1.3 Task Type.....	132
2.1.4 Software Utilization Research .....	133
2.2 MAINTENANCE ACTIVITY SUPPORT .....	135
2.3 TASK TYPE: DEBUGGING VS. ENHANCEMENT .....	137
<b>3. Limitations of the Study .....</b>	<b>138</b>
3.1 FIELD STUDY DESIGN .....	138
3.2 SINGLE MEASURE OF TOOL USE .....	140
<b>4. Contribution .....</b>	<b>141</b>
4.1 IMPLICATIONS FOR RESEARCH.....	141
4.2 IMPLICATIONS FOR PRACTICE .....	142
<b>5. Future Research .....</b>	<b>143</b>
<b>6. Conclusion.....</b>	<b>145</b>
<b>Appendix A .....</b>	<b>147</b>
<b>Appendix B .....</b>	<b>158</b>
<b>References.....</b>	<b>160</b>

# Chapter 1

## Research Problem

### 1. Introduction

While software maintenance is an extremely important activity in most MIS organizations, it is neither well understood nor adequately characterized in the literature. Our knowledge of the maintenance task is limited, as is our knowledge of how the maintenance task can be supported by software engineering tools. This research is motivated by a desire to develop a more comprehensive understanding of the software maintenance task and how software tools can be used to support the maintenance task.

The primary research goal of this dissertation is the explanation of the use of software maintenance support tools by MIS professionals. The theoretical rationale for the dissertation is based on the Theory of Reasoned Action (TRA) (Ajzen & Fishbein, 1980). The dissertation employs the Technology Acceptance Model (TAM) of Davis (1989) as an elaboration of TRA for the case of software usage. This theory is augmented by the addition of a task-technology fit model to explain the attitude formation process in TRA as suggested by Goodhue (1988b; 1992a). The dissertation develops a model of the maintenance task based on the debugging and software understanding literature. This study examines in depth the nature of the maintenance task and the software available to support maintenance.

This research results in recommendations for more effective tools for software maintenance support. To achieve high external validity and generalizability, this study

examined the use of software tools in actual use by working MIS professionals. The principal data collection method was the survey.

## **2. The Nature of The Maintenance Problem**

Software maintenance, the last phase in the software life cycle, is the process of changing existing production software. Changes are made to fix bugs, introduce enhancements, or improve the existing functions of a program or system. The maintenance process dominates the activities of many MIS organizations. On average, 70% of MIS software budgets are devoted to the maintenance process (Swanson & Beath, 1989). In mature organizations, this figure may be much higher. Consequently, in an era of down-sizing and cost containment, there is interest in changing software development and maintenance practices in order to "free" budget dollars for "development" projects and look for ways of making current practices less expensive. Some managers have also recognized the potential for using improved maintenance practices to go beyond cost containment and use the improvements in maintenance practice to improve project cycle time and improve "time to market" for new products necessary to achieve competitive advantage (Moad, 1990).

The investment decision in new software is strongly driven by the relative cost of maintenance and development. The decision to maintain or re-develop is primarily economic and can be analyzed by the MIS manager with conventional financial analysis techniques. As long as an existing system can be modified suitably and the expected value of maintenance is lower, then maintenance is preferred to development (Gode, Barua & Mukhopadhyay, 1990).

Rather than attempting to eliminate maintenance, the "problem" is a question of how maintenance can be performed in the most cost effective manner. The adoption of practices which minimize the need for maintenance and minimize the cost of the needed maintenance should be the goal of MIS management. If maintenance costs for existing systems can be kept low, new development to *replace* old systems will seldom be necessary (Gode, Barua & Mukhopadhyay, 1990).

The introduction of "improved" development practices, most notably the use of CASE technology and structured methods, is expected to result in lower maintenance costs over time<sup>1</sup>. The elimination of the need for *ex post facto* fixes of design errors, and the increase of software maintainability have been predicted, but as of yet this assertion remains to be tested empirically.

### 3. Prior Research on Software Maintenance

In academic research "the maintenance problem" has been addressed in a number of ways. While the literature has been largely prescriptive in character, there have been a small number of empirical studies. Following is a brief survey of the literature of software maintenance which is relevant to this study.

#### 3.1 Maintenance Process

The software maintenance process consists of two major steps, understanding and modification (Pennington & Grabowski, 1990; Yau & Collofello, 1985). Investigations of

---

<sup>1</sup> While improvements in future software maintainability from the use of CASE are highly touted by vendors, no empirical evidence exists to support this claim. In the absence of such evidence, academic experts appear divided on the question.



the maintenance process have employed debugging problems in laboratory settings as the principal research venue. In the debugging process, programmers engage in several types of activities. These are planning, knowledge building, and bug-related activities (Vessey, 1986). Programmers employ diagnostic problem solving strategies in the debugging process (Araki, Furukawa & Cheng, 1991; Vessey, 1985a).

Expertise in the debugging process appears to depend on the programmer's ability to "chunk" subject programs (Vessey, 1986). In addition, experts employ "breadth-first" debugging strategies while novices typically employ "depth-first" strategies (Vessey, 1985a).

The use of expert systems by novices can improve decision outcome quality in subsequent decision situations where the tools are unavailable (Fedorowicz, Oz & Berger, 1992). The novice's decision outcomes were improved even in the absence of the expert system. The use of software maintenance support tools by less experienced users may have a similar improvement in quality.

Prior experience with a program has been shown to improve (decrease) the time needed to perform a maintenance activity (Gould & Drongowski, 1974). In some cases the use of a maintenance support tool may substitute for prior experience with the software being maintained.

Programmers have difficulty understanding software which has "de-localized plans" (Letovsky & Soloway, 1986). This means that the development of a complete and correct understanding of software is difficult if the program's functions are scattered through a program, or if a program interacts with other programs in non-obvious ways. Software

maintenance support tools may assist the programmer, in part, by mitigating the problems associated with de-localized plans (Cleveland, 1989).

Although both debugging and enhancement oriented maintenance are likely to require the same knowledge and activities (Brooks, 1977; Vessey, 1986), no studies comparing the two have been done in either field or laboratory settings. Therefore, in this study, projects of both types are examined so that our conclusions will apply to all types of maintenance.

### **3.2 Software Maintenance Tools**

Software tools can abstract program function to derive a "higher" level specification from raw source code (Hausler, Pleszkoch, Linger & Hevner, 1990). This process is called "reverse engineering". This is distinct from the process of "reengineering" which involves the transformation of a program into an equivalent form which is more understandable by the maintainer (Hanna, 1990). Although it may be possible to automate the reverse engineering process, it may not be possible to totally automate the process for code which was not "generated" from higher level specifications (Gopal & Schach, 1989). This is due to the loss of information about the intended design as code is created. Research is continuing in the application of Artificial Intelligence techniques to the creation of tools which support the reverse engineering process (Waters & Tan, 1991). Some success has been achieved in the creation of tools which support program understanding (Cleveland, 1989; Wilde & Thebaut, 1989). These tools are able to produce higher level abstractions from code in the form of a variety of structure and flow charts. In addition dependency analysis is possible using these tools (Wilde & Huitt, 1991).

The predominant theme in the maintenance tools research discussed above is the support of the software understanding process. Although considerable effort is being expended in

attempts to create automated reverse engineering tools, immediate success is doubtful. The support of a manual reverse engineering process, what we also call program understanding, is likely to be the most productive approach.

#### **4. Tool Assisted Maintenance: Part of the Solution?**

One area Schneidewind (1987) identifies as a key to future productivity gains in maintenance is software tool support for the maintenance process. The use of software tools significantly contributes to perceived maintainability (Kim & Westin, 1988).

The tools which are of interest to us in this study are generally intended to support program understanding, the first step in the maintenance process. These tools assist the programmer/analyst in discovering the physical and logical designs of the program or system at hand. The discovery of impacts of a proposed change on "distant" systems is also an important part of this phase, and is assisted by the software tools in this group. The software understanding or maintenance analysis process is often undertaken under difficult circumstances. Frequently, the change request or "bug fix" must be made under tight time constraints (Letovsky & Soloway, 1986).<sup>2</sup> Hurried analysis, consisting of searches of program and JCL code, leaves many opportunities for error. Complex systems with many modules and links to other systems also invite errors of omission and incomplete understanding. These problems directly affect software quality in the form of

---

<sup>2</sup> Bug fixes, in the experience of the author, often must be accomplished in the late night or early morning hours due to production schedules which place high volume batch processing jobs in non-prime time. Maintenance work under these conditions may be stressful for the programmer due to the immediacy of the task and lack of sleep.

decreased reliability and availability and consequently may have a significant impact on user satisfaction (Letovsky & Soloway, 1986).

The use of software maintenance support tools will likely affect the quality of the maintenance process and product through the support of maintenance analysis process, a key element of which is the program understanding process (Corbi, 1989). The support of this process can logically be expected to reduce analysis errors including errors of omission and misconception. The development of software tools to support program understanding is a key to the improvement of software maintenance productivity<sup>3</sup> (Schneidewind, 1987), (Cleveland, 1989).

## 5. Research Goals

This study seeks to understand the factors which drive or determine the usage of software maintenance support tools. Since software which is not used will have no effect, positive or negative, software utilization is required in order for performance benefits to accrue to the user. Although higher levels of software use are often assumed to yield higher performance, it is not clear that this is a universal truth. In some cases it is possible that high levels of use may detract from the desired result, or may be a symptom of poorly designed software or some extraneous effect.

The principal thesis of this study is that fit between task and technology is a predictor of utilization. This study tests this, and related hypotheses. The study seeks also to examine the moderating effects of prior experience in the formation of perceived fit. The study

---

<sup>3</sup> It is unclear as to whether improvements in productivity are an immediate result of the use of these tools, or whether increased productivity is a longer term gain through the elimination of re-work.

design controls for, and estimates the effects of task type, task complexity, and social norms on the utilization of software maintenance support tools.

A secondary goal of the study, in the examination of task characteristics, is to distinguish between repair oriented and enhancement oriented maintenance in their effects upon software tool utilization. We are specifically interested in the difference between debugging and enhancement maintenance, and how these types of maintenance can be supported.

We build on the work of Vessey (1986) in the area of debugging and examine the maintenance process more generally, and through our study of the use of software tools in the support of maintenance seek to further our understanding of the maintenance process.

We will make recommendations for the enhancement of maintenance support through improvements in software tool characteristics, and software maintenance management practices. Only after we understand the use of these tools in more detail will it be possible to study the relative importance of these factors in determining the performance impacts of maintenance support tool use.

## **6. Overview of the Dissertation**

The remainder of this dissertation is divided into four chapters. In chapter 2, the research model and research questions are developed and discussed. In chapter 3, the research method is described. Finally, in chapter 4, the results of the data analysis process are discussed. Finally in chapter 5 the results of the data analysis are examined and the implications of the study are discussed.

# Chapter 2

## Research Model

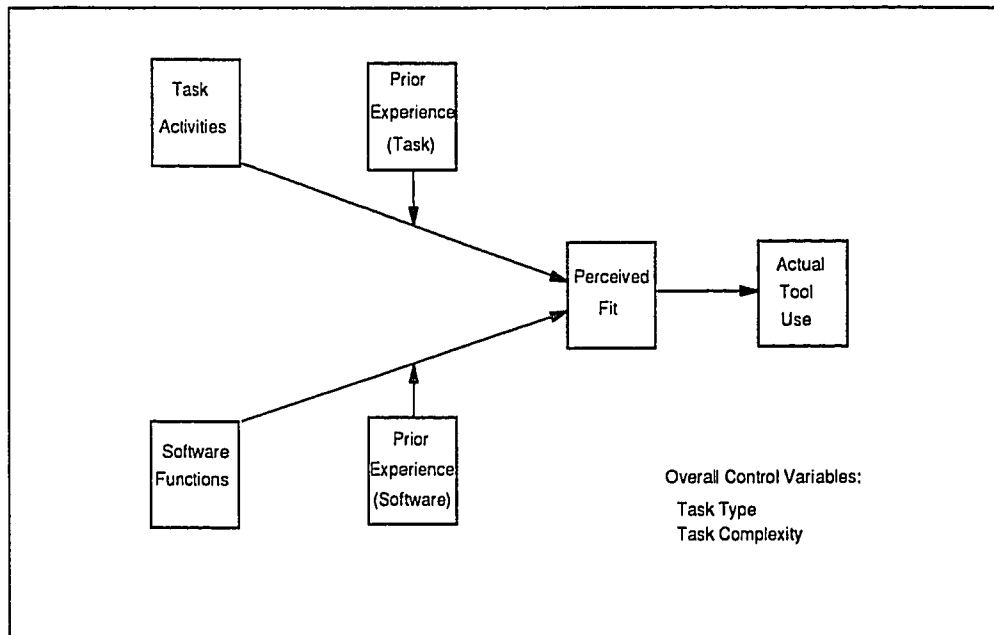
### 1. Introduction

The study focuses on those maintenance task factors which drive the use of software engineering technology in the software maintenance task. A model of maintenance software tool usage has been developed and is discussed in detail below.

This model assumes that software maintenance is generally undertaken by an individual programmer analyst, the roles of management and the user community notwithstanding. We believe that a more thorough understanding of the maintenance process at this level is needed to effect fundamental improvements in the maintenance process and its support. The assumption that maintenance is an activity of an individual programmer analyst is also necessary, and reasonable, given the current functionality available in the commercial software maintenance support products. These tools support the low level, individual oriented, maintenance task. Unfortunately, tools which support higher level functions, and tools which integrate the maintenance and software redevelopment processes are either nonexistent or experimental at this time.

The research model developed here applies to maintenance tasks which are non-trivial and involve a significant effort on the part of the programmer to implement. Maintenance efforts which fall into this category (non-trivial, significant effort) are supportable by software engineering tools. Very simple projects do not require such support and are easily accomplished without external support.

This research tests several hypotheses about maintenance tool usage, the nature of maintenance tasks, and the fit between maintenance tool functionality and the needs of these maintenance tasks. These hypotheses were developed from the research model shown in Figure 1. This research model, in turn, is a subset of the overall research framework shown in Figure 2.



*Research Model  
Figure 1*

This framework is based on two models from the general MIS literature. The Technology Acceptance Model (TAM) of Davis (1985) provides the basic framework for examining usage. However, the TAM model does not explicitly include task characteristics. The Task/Technology Fit Model of Goodhue (1988b; 1992b) provides an approach for examining task influences on software usage. Integration of the two models results in a more comprehensive framework which relates task and technology characteristics to actual software usage.

The software debugging model found in Vessey (1986), is used in the development of a general maintenance task model. The Vessey debugging model is augmented by the work of Letovsky (1987) and Letovsky and Soloway (1986) in the area of software understanding. Finally, a software maintenance tool functionality model is developed based on the problem solving literature and the work of Henderson and Cooperider (1990) in their development of the Functional Case Technology Model (FCTM).

The remainder of this chapter is divided into five major sections. Section 2 reviews the models which form the basis for this study's research model. Section 3 discusses the nature of Task-Technology fit in more detail. This is followed by Section 4 where the research questions for the study are developed. The chapter concludes with a recapitulation of the model in Section 5 and the conclusion in Section 6.

In the following section these four models (TAM, Task-Technology Fit, FCTM, and the software debugging model) are discussed, as is the rationale for the integration of the models.



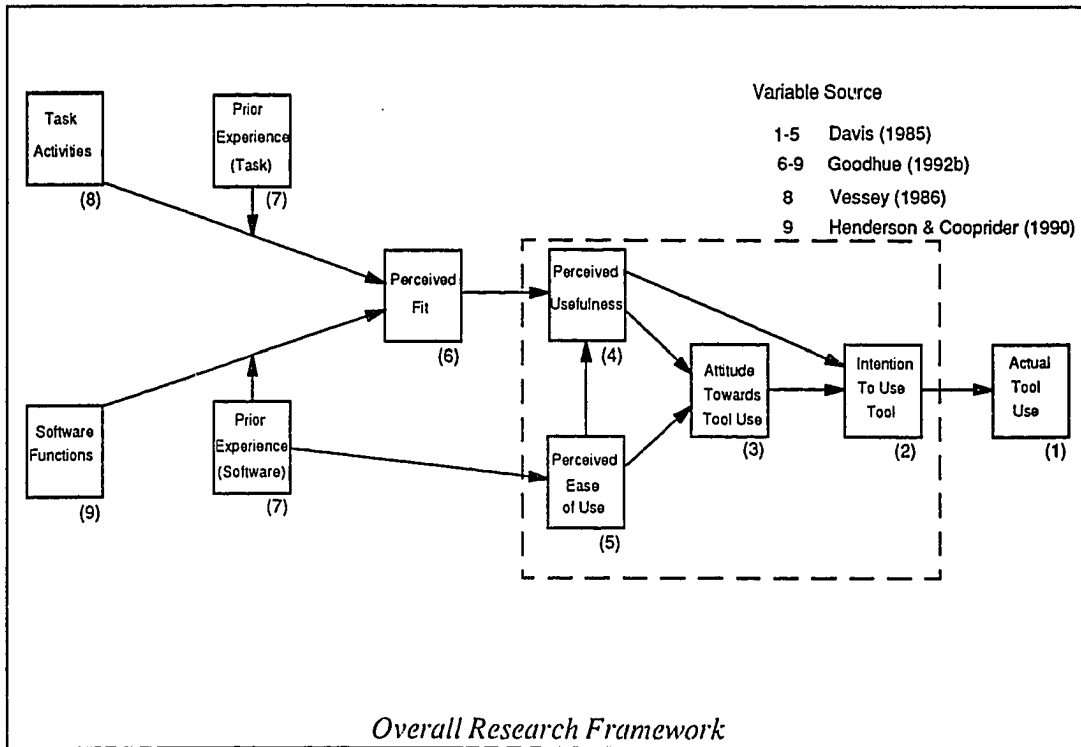


Figure 2

## 2. Background On The Four Models

### 2.1 The Technology Acceptance Model (TAM) (Variables 1-5)

The Technology Acceptance Model of Davis (1985) is a specific adaptation of the Theory of Reasoned Action (TRA) model of (Ajzen & Fishbein, 1980) to the study of computer software usage. The TRA and its successor, the Theory of Planned Behavior (TPB) (Ajzen, 1985) are well known, and have been widely employed in the study of specific behaviors (Ajzen & Fishbein, 1980).

The TAM model (Davis, 1985), shown in Figure 3, is a validated model for examining software tool usage. Davis' TAM model is an elaboration of the Theory of Reasoned

Action for the specific case of software usage. It has been tested by the original author (Davis, Bagozzi & Warshaw, 1989) in a study of word processing software use, by Mathieson (1991) in a study of spreadsheet use, and by Adams, Nelson, and Todd (1992) in a study of the use of several widely available end user software packages.<sup>1</sup> This study employs the TAM model as the basic rationale for the examination of usage behavior.

Davis examined Acceptance, which is more properly an adoption of innovation variable, at a "macro" or task set level. However, in this study the software has already been adopted or accepted by the programmer analyst. This research examines the specific, or "micro", choice to employ a tool for a particular maintenance task rather than the general, or "macro", choice or acceptance of a software tool for a class of problems. The application of the TAM model to a "micro" level of analysis is justified because the parent models, TRA/TPB have been applied to the "micro" level. For example, the TRA has been used to examine voting behavior (Fishbein, Ajzen & Hinkle, 1980), career choice (Sperber, Fishbein & Ajzen, 1980), weight loss (Sejwacz, Ajzen & Fishbein, 1980), and consumer purchasing behavior (Fishbein & Ajzen, 1980).

---

<sup>1</sup> WordPerfect, Lotus 1-2-3, and Harvard Graphics were available to the end user.

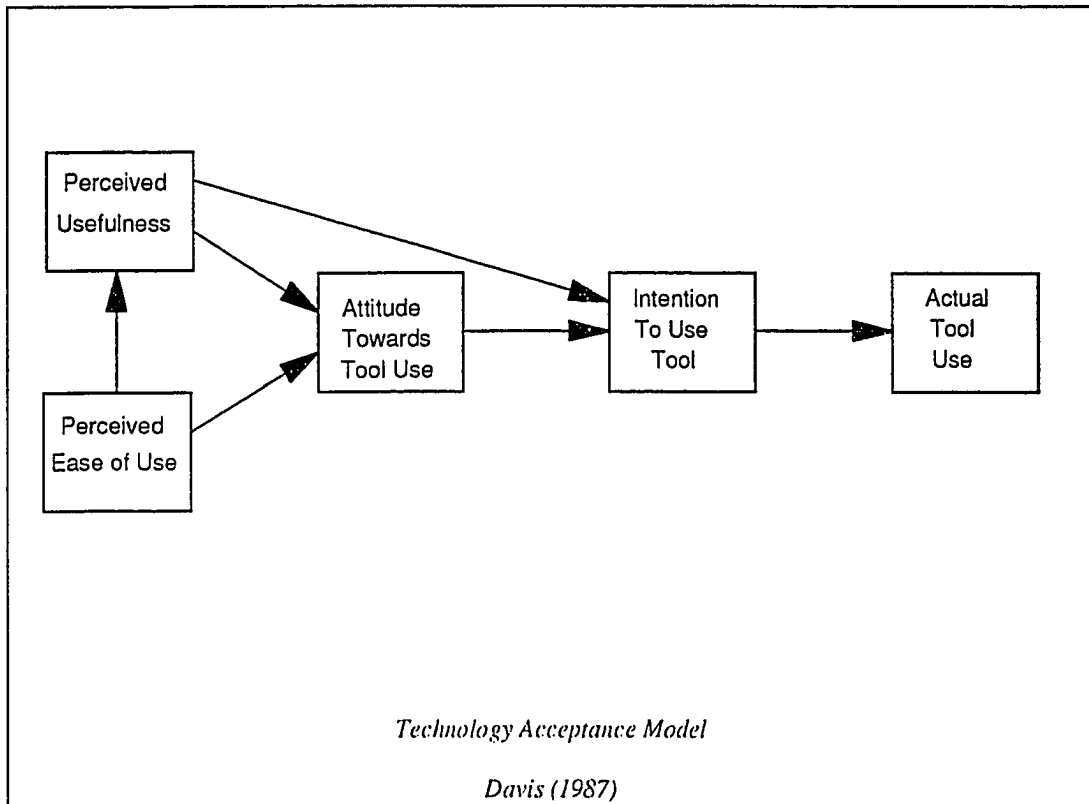


Figure 3

Davis' research, in essence, examines the external variables which determine or influence attitude towards tool use. The TAM model identifies perceived ease of use, and perceived usefulness as key independent variables. The perceived usefulness variable is most closely tied to task characteristics. It represents the subject's perception that a particular tool is helpful or useful in the accomplishment of some task. Perceived ease of use is tied most closely to software tool characteristics. This relationship is moderated by the experience of the programmer with the tool.

Davis (1985) modified the TRA to produce the TAM. Davis found that Subjective Norms, a variable in TRA/TPB did not produce significant variance. Therefore using the principal of parsimony, he eliminated this variable from the model. Although this

variable was found to be significant by Ajzen & Fishbein (1980), the lack of significance of subjective norm variables was also found by Mathieson (1991) in a comparison of TAM with the Theory of Planned Behavior model. TAM achieved superior predictive power when tested against the more general models that explicitly include social norms (Davis, Bagozzi & Warshaw, 1989; Mathieson, 1991). Thus, TAM is employed without recourse to the TRA or TPB additions.

### **2.1.1 The Basis of TAM**

In general, these theories (TRA, TPB, TAM) state that a behavior is determined by intention to perform the behavior. Intention, itself, is determined by attitude towards the behavior, perceived behavioral control over the behavior, and subjective norms concerning the behavior. Actual behavior and intention variables have been found to be highly correlated (Ajzen & Fishbein, 1980; Davis, 1985).

The original TRA includes the very important assumption that the behavior is volitional, which is to say voluntary or at the discretion of the user. While many behavior choices are indeed volitional, many, especially in business environments, are not. This problem is addressed in the Theory of Planned Behavior (Ajzen, 1985) which is shown below.

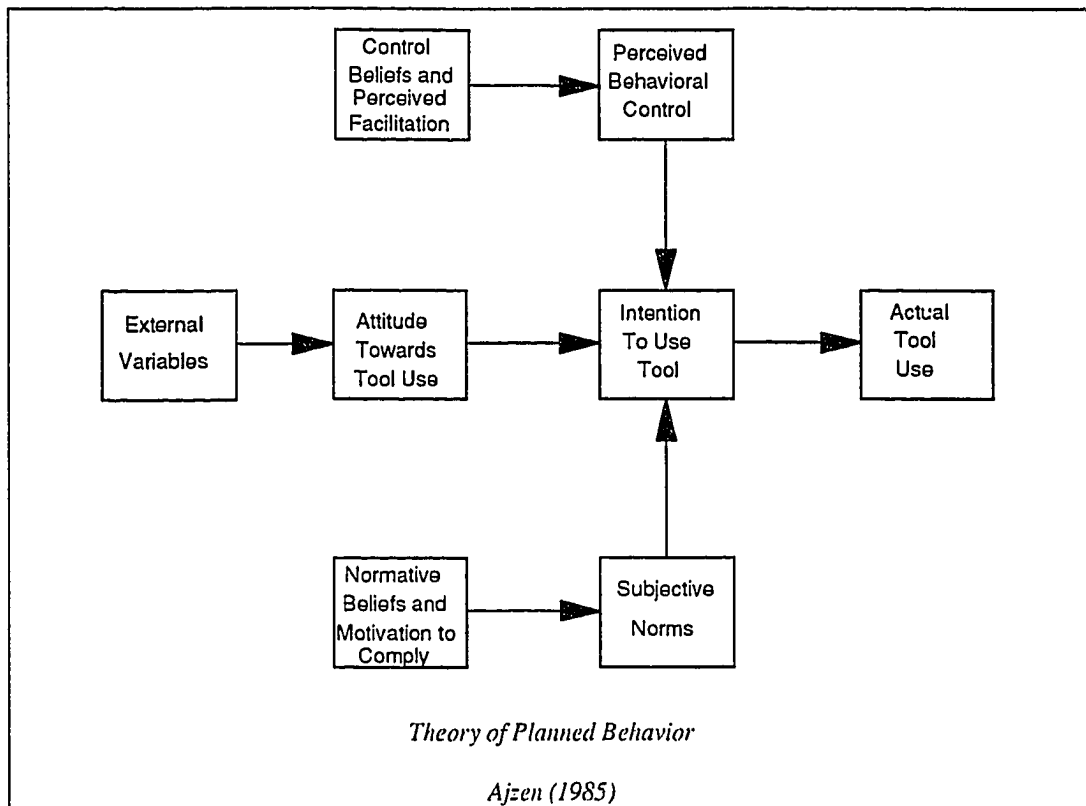


Figure 4

Intention to perform a behavior (e.g., to use a maintenance tool during a maintenance task) is in turn determined by the individual's attitude towards the behavior and by the subjective norms experienced by the user. Figure 4, illustrates the TPB as adopted for the current research context.

### 2.1.2 Definition of TAM Model Variables

#### Attitude Towards Behavior (Tool Use) (Variable 3)

A person's attitude towards the behavior is a key determinant of intention to perform the behavior in question, in this case the usage of a software tool. In general, attitude towards a behavior is linked or associated with the external variables (Ajzen & Fishbein,

1980). While in the TRA model external variables remain unspecified, TAM recognizes Perceived Usefulness and Perceived Ease of Use as the determinants of attitude towards tool use. Tasks constitute an important part of this external variable environment (Davis, 1985). External variables can be thought of as a determinant of the attitude towards use. The person using the software expects that the tool is useful in the performance of some duty or the accomplishment of some task.

### **2.1.3 External Variables: Attitude Formation**

Attitude towards the behavior in question is an intervening variable in the research model for this study. Following below is a discussion of the two factors which determine attitude towards software usage: perceived usefulness and perceived ease of use.

#### **Perceived Usefulness (Variable 4)**

The perception of usefulness means that the maintainer holds the opinion that the use of a particular software product in a situation will be beneficial. In order for the maintainer to hold such an informed opinion several conditions must have occurred. First the maintainer must have prior experience with the type of problem at hand, and therefore must have a perception of the nature of the problem, even if the problem is not yet understood sufficiently to derive a solution. Generally, the maintainer must also have experience with the software tools which are at his/her disposal. This experience gives the maintainer a basis for evaluating the capabilities of the tools and how and in what circumstances they may be useful. In the formation of initial opinions, the maintainer will not have hands-on experience, but may know of the capabilities of the software through documentation or other communication channels. It should be noted that

perceived usefulness may be negative; a software tool may be perceived to not be useful for a particular task.

### Perceived Ease of Use (Variable 5)

The second key variable in the Davis TAM model which determines or influences attitude is the perceived ease of use of the software. This variable is determined, at least in part, by prior experience in the use of the tool and in maintenance in general, as well as by the amount of training received by the user. There is a relationship reported by Davis between Ease of Use and Usefulness. Usefulness is influenced somewhat by Ease of Use.

#### **2.1.4 Alternative Usage Model**

If the Technology Acceptance Model were not available as a model validated for software usage, the Bagozzi (1982) model would be an appropriate alternative. It is closely related to the Theory of Reasoned Action, differing in that it incorporates two variables which are missing or are handled indirectly in the TRA model. These are affect toward usage, and habit. The affect variable in the Bagozzi model parallels the attitude variable in the TAM model. It does not assume rationality as does TRA and TAM, and, as such, is a more broadly defined variable. Attitude is a construct which can be measured indirectly, generally by measuring affect. Thus affect and attitude may be confounded. In addition, the literature suggests that these variables are very close. Goodhue (1992b) suggests that the TAM model does, in fact, measure affect. The habit variable in Bagozzi (1982) is largely subsumed by the prior experience variables in this research model.

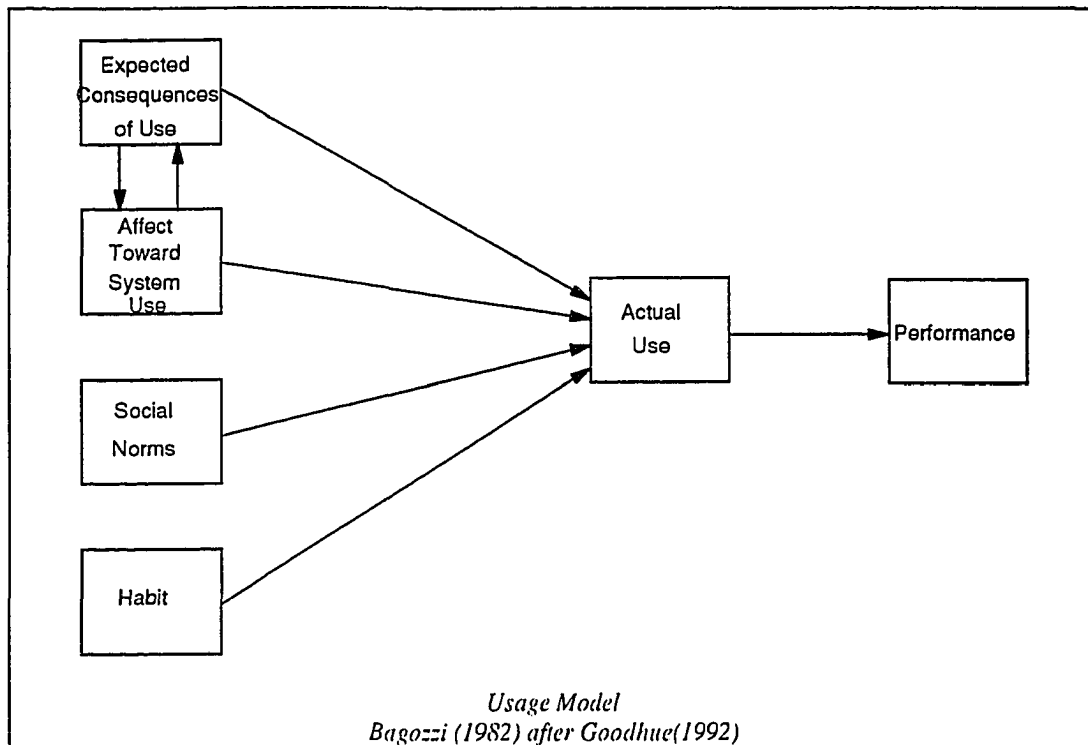


Figure 5

The Bagozzi (1982) usage model offers an alternative to the Davis (1985) TAM model. It is more general than the Ajzen (1985) TRA model which forms the basis for TAM. In addition, the Bagozzi model seems to address some of the questions raised over the use of expectancy models in user satisfaction research. Melone (1990) raises questions over the appropriateness of overly rational models such as TRA/TPB. In addition, Melone (1990) points out that the TRA and related models essentially assume that the user will have already formed attitudes toward the behavior in question. Evidence indicates that there are circumstances under which users form an attitude only when asked about their attitude (Melone, 1990).

Although these are serious questions, they do not defeat the basic TRA/TPB or TAM models which have been validated and shown to have reasonable predictive power ( $R^2$  in



excess of .6 for either model). In addition the Bagozzi (1982) model has not been used in the software usage literature. The Davis (1985) model is more appropriate for this research because it achieves acceptable prediction power, and it is specifically tailored towards software usage.

While the TRA based model, TAM, is favored over the Bagozzi model, the latter is used in the framework developed by Goodhue (1992b) in which the fit between the task and available technology is related to software usage and ultimately to performance.

## **2.2 Task/Technology Fit Model**

While incorporating technology characteristics into its general framework, the TAM model (Davis, 1985) does not explicitly include task characteristics. It indirectly accommodates task characteristic variables, however, in the Perceived Usefulness variable. The whole notion of usefulness implies that the software is used for something.

Thus, a link of the TAM model with a model specifically incorporating task characteristics is warranted. Further justification for elaborating the TAM to include an explicit reference to task is provided by the arguments of Goodhue (1992b).

Goodhue linked his Task/Technology Fit model, shown in Figure 8, with the technology usage model of Bagozzi (1982), shown in Figure 5. Specifically, Goodhue asserts a link between perceived fit between task and technology and the key independent variable in the Bagozzi model, the expected consequences of using a technology. The latter variable corresponds to Perceived Usefulness in the Davis model. Making the same link to the TAM model; perceived fit between task and technology, determines, in part, Perceived Usefulness. This link is reasonable because the underlying assumption in the TRA family of models, including TAM, is that a person engages in a behavior because he or

she has evaluated the benefits of engaging in that behavior and expect a certain result. The expected consequences construct for software usage in the Bagozzi model parallels, if not entirely contains, the Perceived Usefulness construct in the Davis model. The only significant difference between these models is in the assumptions regarding the evaluation process. The Bagozzi model allows for evaluation based, in part, on affect. The Davis model is more narrow in that it assumes an evaluation process which is rational or in the very least boundedly rational.

## **2.3 The Maintenance Task**

The software maintenance activity model of Vessey (1986), together with the program understanding literature, is used to develop a model of the maintenance process which is can be used as the task component of the task - technology fit model for maintenance.

### **2.3.1 Basic Software Maintenance Activity Model**

Generally, software maintenance is divided into three categories: adaptive, corrective, and perfective (Swanson, 1976). Later authors such as Bendifallah & Scacchi (1987) frequently add preventative maintenance as a fourth category. These categories are based on the intent or goal of the maintenance activity. Briefly, adaptive maintenance has as its goal the modification of the software to accommodate a changed business environment. Corrective maintenance involves changes that are intended to eliminate erroneous output. Perfective maintenance has as its goal the improvement of the program's technical functions. A partial re-write to improve access time is an example of perfective maintenance.

For the purposes of this research we adopt a more parsimonious division of maintenance. Maintenance is examined based on the type of activity performed during maintenance

without regard to the actual intent or goal of the process. The literature identifies two general classes of maintenance activity: debugging and enhancement (Pennington & Grabowski, 1990). Debugging involves locating and repairing a problem without altering originally intended functionality. Enhancements are modifications to existing software functions such that the originally intended functions of the software are changed. This division of maintenance activities pre-dates Swanson (1976), who essentially introduced a managerial focus to the discussion of maintenance. Debugging is equivalent to Swanson's (1976) corrective maintenance while enhancement includes both adaptive and perfective maintenance.

While there is no comprehensive model of the maintenance process, Vessey (1986) asserts that the debugging model she developed applies to the general maintenance process. Vessey's assertion is based on the belief that the process of enhancement involves the same basic process as debugging, namely the understanding of the software at hand. The difference between the two processes is essentially the assumption as to the goal of the actual change required in the code.

The activity shared by enhancement and debugging maintenance is problem solving (Vessey, 1986). The difference between debugging and enhancement arises in the actual programming, or code creation step, that occurs after the "problem" is identified and understood. However, the differences may not be very significant. Professional programmers sometimes euphemistically refer to software bugs as "Undocumented Features". If we view debugging as the modification of an "Undocumented Feature", rather than the elimination of an error, then debugging and enhancement maintenance are similar processes. In either case, the programmer must develop an understanding of the program and identify the modules to be changed in order to effect the desired change in

software behavior. Restated briefly, maintenance is a process which alters the existing behavior of software.

In Vessey (1986) three models of maintenance are described. The first is a process model, the second is a function model and third is the activity model. The activity model has been chosen for this research for several reasons. First, it is well grounded in the problem solving literature. Second, it has been used successfully in empirical studies by Vessey (1985a; 1986). Third, it represents a model which can be employed in a fit model between tool functionality and supportable activities.

Vessey (1986) divides maintenance debugging activities into three categories. These are planning activities, knowledge building activities, and "bug-related" activities. In the discussion below, the elements of Vessey's maintenance activities are summarized.

### Planning Activity

Planning activities center around the management of the task and the coordination of other activities towards the completion of the task. Vessey (1986) divides these activities into two sub-categories, goal and procedure/strategy. The first sub-category, goal, involves goal setting and goal management. Goals are identified states or objectives that the problem solver seeks to achieve. Goal in this sense may be taken to mean overall goal, sub-goals, or goal sets. An example of goal setting in maintenance would be the identification of a desired outcome of the maintenance process such as the addition of a field to a report or the elimination of a particular bug. As an activity it requires information about the nature of the problem and the resources available.

The second type of activity in this category is procedural or strategic in nature. Here the problem solver identifies and selects the activities to be performed to achieve the desired

goal or sub-goal. This process may proceed at either a high or low level of detail depending upon the goals in question.

### Knowledge Building Activity

Knowledge building activities center around the procedures involving the management of information needed to accomplish the goal at hand. Vessey (1986) divides these activities into two categories, information gathering and knowledge retrieval.

Information gathering in the Vessey (1986) description is divisible into three groups of sub-activities. First, and most dominant, of these is the actual gathering of information from any of several sources, including listings, documentation and data structures. As part of this process or activity, the programmer may mentally execute parts of the source code to obtain information regarding the software functions being performed. The second information gathering sub-activity described by Vessey (1986) is the evaluation of information. During this process the programmer weighs the significance of information or an assumption such that the outcome of the assessment may influence further activities. The third sub-activity is the identification of understanding of a process or program module or section. This is described by Vessey (1986) as "... makes a statement of understanding about the task situation." The act of stating understanding may be instantaneous and sub-conscious.

Knowledge retrieval is a relatively straightforward activity. It is the accessing of technical information regarding the language of the program or related system software. The programmer may retrieve knowledge from memory or from other external sources.

### "Bug-Related" Activity

The "Bug-Related" activities in the Vessey maintenance activity model center around three types of actions, clue generation and management, hypothesis generation and evaluation, and error management. The first two categories here are the essential elements of diagnostic problem solving (Araki, Furukawa & Cheng, 1991; Pennington & Grabowski, 1990). Vessey (1986) identifies an activity which she refers to as "bug-related" activity which we separated into two sub-activities and renamed Diagnosis and Treatment in our earlier discussion. These terms are chosen for two reasons. First, diagnostic reasoning is quite well known in the problem solving literature (Davis, 1984; Elstein, Shulman & Sprafka, 1978; Torasso & Console, 1989). Second, the parallel with more familiar medical examples is obvious and widely understood. Treatment as an activity represents the actions engaged in by the programmer which actually result in a change in functionality and is discussed below as the principal component of the Modification activity.

Diagnosis is that set of activities in which a programmer, after having gained sufficient basic information about the "case", sets out an initial differential diagnosis (a set of possible problems) and performs tests to narrow down the possible problems to a definitive cause. Diagnosis is supportable by functions which allow a programmer to trace execution of program variables, create flow charts, etc. In performing diagnosis, the programmer establishes a working hypothesis about the program or problem and sets out to test the hypothesis; seeking either to accept or reject the premise. This activity or work function is intimately tied to knowledge building; neither is meaningful or useful without the other.

The final category in this group is error management, during which the actual error in the program is "repaired".

### **2.3.2 Revised Software Maintenance Activity Model**

This study views software maintenance as an iterative two stage process. First the programmer undertakes to understand the program. This is followed by code modification or construction (Letovsky & Soloway, 1986). These two stages are performed iteratively until the problem is solved.

The programmer who is attempting to understand a program must examine the code, samples of input and corresponding output, and any other available system artifact such as program documentation. The literature identifies different strategies of how programmers attempt program comprehension (understanding) (Corbi, 1989).

In one approach to the study of program understanding, the programmer is viewed as having a choice between systematic and as-needed strategies. In the systematic strategy the programmer essentially bench tests the program to detect "causal interactions among components of the program" (Littman, Pinto, Letovsky & Soloway, 1987). In an as-needed strategy, the programmer attempts to localize the problem and thus narrow the problem focus. Unfortunately the programmer may miss key interactions. Programmers using a systematic strategy are more likely to be successful in developing a correct understanding of the program and modifying the program successfully (Littman, Pinto, Letovsky & Soloway, 1987).

The approach of Brooks (1983) is consistent with the work of Vessey (discussed above) in the area of debugging and expertise. Brooks views program understanding as an iterative approach where the programmer gathers facts, forms hypotheses based on

available information and prior experience, tests the hypotheses, and formulates plans for further work based on the outcome of the hypothesis testing activity. This approach views program understanding as a diagnostic problem. Similar approaches to understanding are apparent in different problem domains such as electronics debugging (Davis, 1984) and medicine (Schaffner, 1985).

The approach of Letovsky (1987) is useful in integrating some of the divergent viewpoints on program understanding. He found that programmers use a mixed strategy in attempting program comprehension. Programmers use available clues and adopt top-down or bottom-up strategies as needed. Letovsky (1987) identifies, in his protocols of programmers engaging in maintenance activities, the programmer in the process of fact gathering, asking questions, making conjectures about the question, and then seeking further facts to answer the question. This is diagnostic problem solving consistent with the views of Brooks (1983). These findings indicate that programmers are adaptive, and can modify their problem solving strategies to fit the circumstances.

Although program understanding activities are at the core of the maintenance process, modification or construction activities are not entirely anti-climatic. The programmer must test any modified or new code to ensure that the desired effect is obtained.

Frequently the programmer is unsuccessful on the first try. In fact several iterations are often required to eradicate logic errors, even after the elimination of routine compiler errors. The maintenance process for any particular problem can thus be viewed as an iterative process consisting of several cycles of understanding and modification.

The Vessey Software Maintenance Activity Model (1986) serves as a framework for examining the program understanding process. The Vessey activities map onto our more basic maintenance model in the following way. Knowledge building, planning



(development and recognition), and diagnostic activities are clearly intimate parts of the understanding process. Error management is the essence of modification.

## **2.4 Maintenance Support Technology**

In this section we develop a model of maintenance software support tool functionality. This model is developed from two sources. The first source is a model of CASE support for design projects. The second source is the maintenance task literature.

### **2.4.1 CASE Tool Function Model (FCTM)**

While there is no framework available from the literature which explicitly addresses maintenance tool functionality, we employ the general model of CASE functionality (FCTM) developed by Henderson (1990), informed by the program understanding literature, as a basis for a maintenance tool functionality model. The FCTM model appears below.

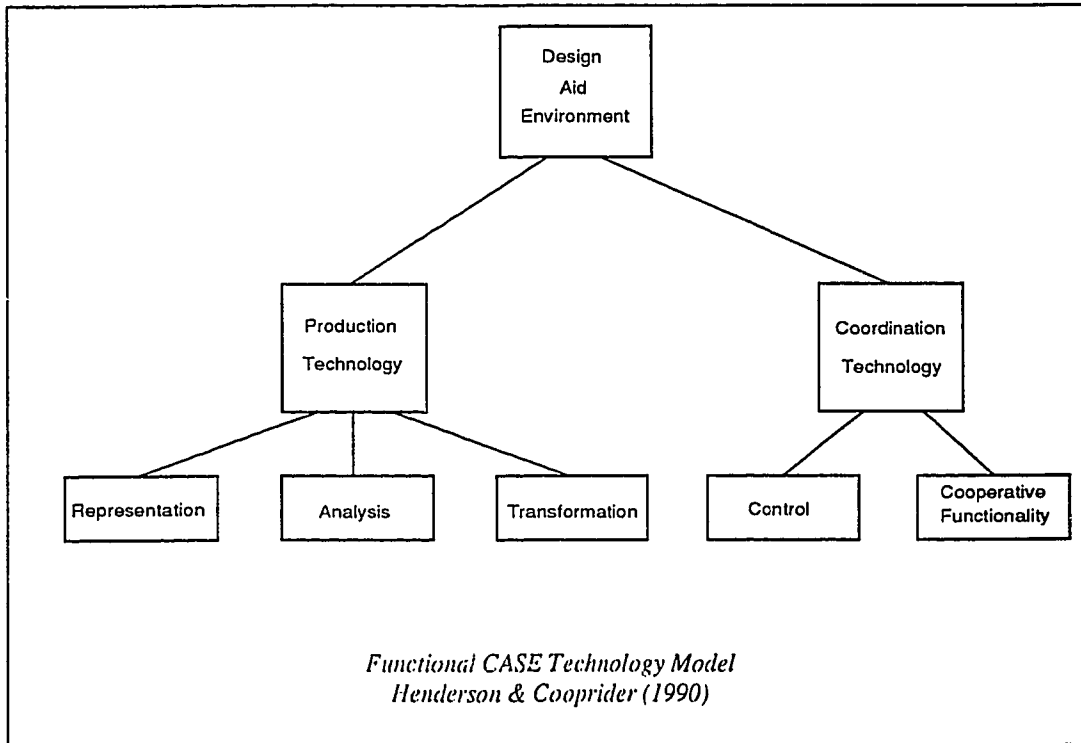


Figure 6

The Henderson (1990) model is a general model which can be applied to design situations. It has been used successfully by Robertson(1990) who studied the mechanical design process. Since software maintenance also is a design problem, we employ the FCTM in the study of software maintenance.

The model illustrated above is the simpler of two models developed in the Henderson (1990) paper. The Production dimension of the Henderson & Cooperider model (1990) includes representation, analysis, and transformation functions, which are the primary functions of software maintenance support tools. These functions are discussed briefly below.

## Representation

Representation functionality of software maintenance support tools parallels that of development oriented CASE tools. Consider the definition of representation:  
*... functionality to enable the user to define, describe or change a definition or description or an object, relationship, or process. (Henderson & Coopriider, 1990)*

This definition, although targeted towards the conventional CASE tool, clearly applies to any functionality of maintenance tools which render the structure of programs or databases in graphical form.

## Analysis

The analysis functionality which we associate with CASE tools has analogs in maintenance tools. Analysis is defined in the FCTM as follows:  
*... functionality that enables the user to explore, simulate, or evaluate alternate representations or models, relationships, or objects. (Henderson & Coopriider, 1990)*

Maintenance tools which allow the user to rationalize data names or trace execution paths to *discover* or test possible relationships that exist between objects of computation, clearly fall under the definition of analysis. Tools which trace variables and display variable contents also fall into this category.

## Transformation

The transformation dimension of conventional CASE tools does not have as clear an analog in the realm of maintenance tools.  
*... functionality that executes a significant planning or design task, thereby replacing or substituting for a human designer/planner. (Henderson & Coopriider, 1990)*

Examples of conventional CASE tool function in this area generally include code generators. In the maintenance tool environment the closest analog is re-coding or

restructuring software. On the data base "side" we find software which is capable of translating, for example, an IMS data base into an equivalent DB2 data base.

The use of re-coding tools in software maintenance projects appears to be increasing in popularity. In traditional maintenance projects, the programmer usually avoids gross or wholesale modification of software. As software ages, it becomes more difficult to modify. Although several factors contribute to this difficulty, a major cause is that a heavily modified program is difficult to understand. Today, programmers use software re-coding tools in order to make wholesale manual rewriting of code unnecessary. Re-coding modifications do not change the basic functionality of the software; it "only" improves readability.<sup>2</sup> Software *transformed* in this manner may be more easily understood and maintained.

### Coordination

The coordination dimension of CASE technology involves two dimensions, control and cooperation. Control technology enables the user to create and enforce rules and standards during the development or maintenance process. Cooperation technology, on the other hand, allows users to exchange information with other individuals (Henderson & Cooperider, 1990).

These functionalities are implemented in some software maintenance environments. Programmers are frequently required to submit programs about to be released for production for analysis to determine compliance with "shop" standards. Program or

---

<sup>2</sup> Re-coding software is sometimes called a re-engineering tool [Hanna, 1990].

project managers use project management tools to communicate tasks and deadlines to their staff of programmers and analysts.

Programmers frequently use e-mail and other coordination devices in an effort to keep their work consistent with other projects or activities which are occurring simultaneously. The most significant of these tools are library and configuration management tools.

### **2.4.2 Maintenance Support Function Model (MSFM)**

Comprehensive support of software maintenance must address the basic tasks of software maintenance. The Maintenance Support Function Model (MSFM) developed for this research consists of three dimensions, Understanding, Modification, and Coordination. These dimensions are developed from the FCTM (Henderson & Cooperider, 1990) as described below.

By deconstructing the FCTM (Henderson & Cooperider, 1990), we find that the functions of Representation and Analysis can be combined into a single function called Understanding.

The third variable in the FCTM production dimension is transformation. Tools which address this function directly support the program modification task. The second dimension of the revised maintenance support model is program modification.

The third and final dimension of the maintenance support function model is coordination. Tools in this dimension are concerned with supporting the programmer in conforming to shop programming and documentation standards as well as supporting the release and turnover process. The coordination of a maintenance project activities in the MIS organization is accomplished through project management (Planning and Tracking) tools.

The following figure summarizes the Maintenance Support Function Model:

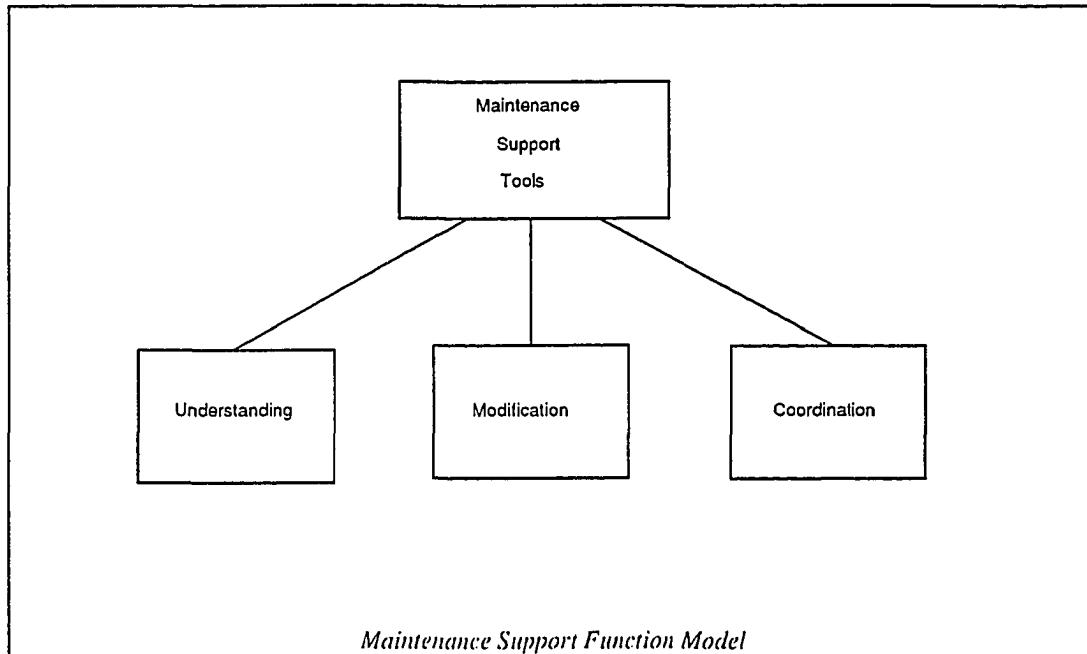


Figure 7

### 3. Task - Technology Fit

A fundamental argument of this research study is that software will be used if the functions available to the user support (fit) the activities of the user (see research model, Figure 1). A software function supports an activity if it facilitates that activity.

Alternatively, the software must serve to lower the cost to the user of the performance of some task or action. Rational, experienced users will choose those tools and methods which enable them to complete the task with the greatest net benefit. Software which does not offer sufficient advantage will not be used. The ability of software to support a task is expressed by the formal construct known as Task-Technology Fit, which is the matching of the capabilities of the technology to the demands of the task.

### 3.1 The Definition of Task - Technology Fit

Although a universal definition of task-technology fit does not exist, the literature contains several similar definitions. Consider the definition of cognitive fit (Vessey & Galletta, 1991):

*Cognitive fit is a cost-benefit characteristic that suggests that, for most effective and efficient problem solving to occur, the problem representation and any tools or aids should all support the strategies (methods or processes) required to perform that task.*

The definition of cognitive fit emphasizes the support of task performance by appropriate tools and problem representation, and is similar to the Goodhue (1992b) definition of "task-system" fit:

*"Task-system" fit is the degree to which an information system or systems environment assists individuals in performing their tasks, or the fit between task requirements and the functionality of the IS environment.*

Higher degrees of "Fit" lead to higher performance (Goodhue, 1988b), and expectations of consequences of use (Goodhue, 1992b). The latter finding is of primary interest in this study as it provides a link between fit and perceived usefulness. Essentially the Goodhue model states that the perception of the user that a software tool is appropriate for a certain task represents "Fit" between task and technology. The Goodhue model appears below in Figure 8.

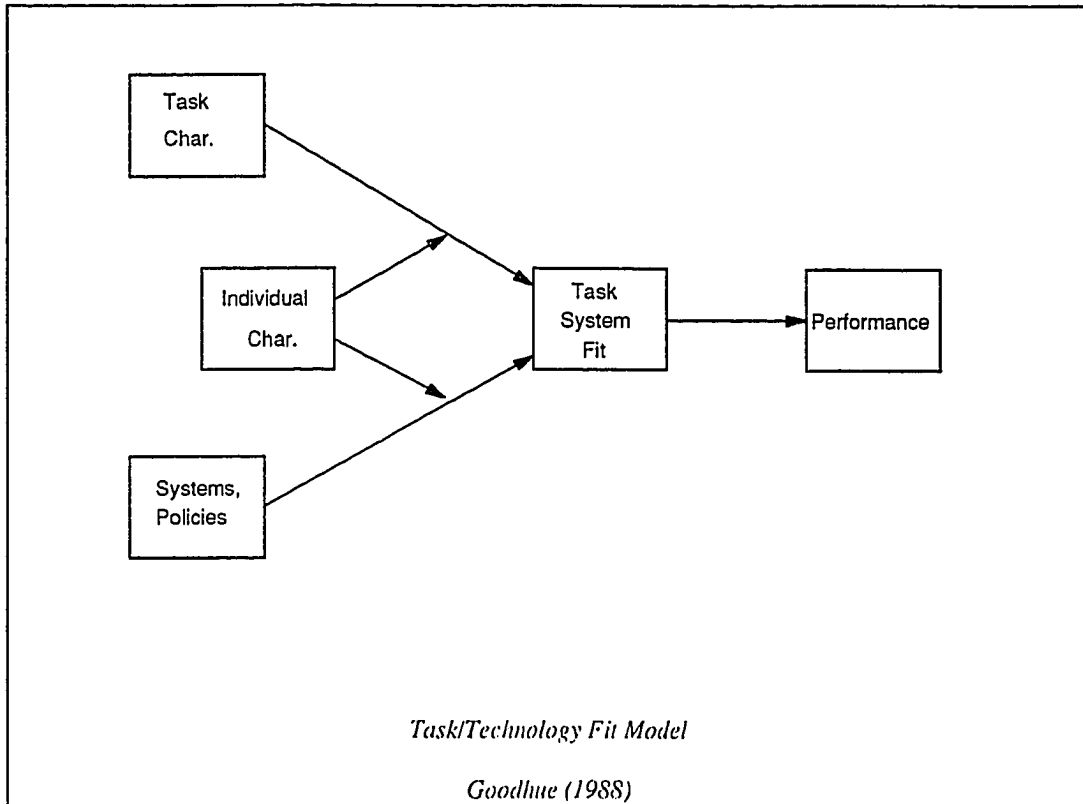


Figure 8

The Goodhue (1992b) definition is similar to the definition of Nance (1992 ) which was employed in his study of auditors use of software tools:  
*the degree to which an available information technology is useful in supporting the unique needs of a given task.*

In the studies discussed above and in Vessey (1991) the dependent variable in the models of fit is performance. This study focuses on the performance antecedent, tool usage, as the dependent variable. In cases where use is not voluntary, use need not be considered as the dependent variable; the most appropriate dependent variable in such cases is performance (Goodhue, 1994). However, we assume that the use of software maintenance tools is voluntary based on preliminary interviews with managers and programmers in the subject organizations. This assumption allows us to consider use as



the dependent variable which is closer, from the perspective of the causal chain, to the independent variable fit.

In this research we employ the following definition of task-technology fit:

*Task-Technology fit is the matching of the functional capability of available software with the activity demands of the task at hand.*

This definition corresponds with the definition of cognitive fit with respect to the “Tables-Graphs” problem described by Vessey (1991), who argued that the cognitive fit paradigm is sufficiently inclusive to allow it to be extended broadly into other problem solving venues. The same argument allows us to extend it to cover the support of software maintenance activities by maintenance oriented software engineering tools.

This definition allows us to explore the fit between the maintenance task activities of Understanding, Modification, and Coordination with the corresponding maintenance support functions of Understanding, Modification, and Coordination. The variables or task/technology characteristics of these models which “fit” are Understanding, Modification, and Coordination.

The following matrix illustrates the primary dimensions of fit between maintenance task and maintenance support technology.

**Technology  
Function**

		Understanding	Modification	Coordination
Task Activity	Understanding	X		
	Modification		X	
	Coordination			X

*Task - Technology Fit Matrix*

*Figure 9*

In the above figure an X indicates where significant fit is primarily expected to occur between a task activity and a technology function based on cognitive fit theory.<sup>3</sup> It is important to note that fit between task and technology other than those noted will, indeed must, occur. The tasks of programming or maintenance are not accomplished in isolation nor are they discrete. These tasks do not (cannot) occur in a linear fashion. The programmer will perform tasks recursively, simultaneously, or in parallel. In some cases an action may actually fall, simultaneously, into two task categories. The different types of task are thus dependent, one upon another. It is therefore logical to expect that the software function designed, or at least intended, to support an understanding task might also support modification, etc.

---

<sup>3</sup> Although Vessey & Galletta (1991) refer to the cognitive fit "paradigm", we believe that it is more appropriate to refer to cognitive fit as a theory, especially when addressing a specific research issue.

### 3.1.1 Dimensions of Maintenance Task and Support Technology Fit

In this section the three dimensions of fit are discussed. As indicated above, these dimensions are Understanding, Modification, and Coordination. Each type of task is posited to be supported by a corresponding software tool function.

#### 3.1.1.1 Supporting the Program Understanding Task

The support of the software maintenance process, especially the program understanding portion, can be viewed as a problem of supporting representation development, manipulation, and testing. At the heart of the process of understanding a program is building a representation of the problem to be solved. A problem representation is an mental model or conception of the problem (Pennington & Grabowski, 1990). The development of a program representation appears to be an essential part of the software understanding process.

The importance of program or problem representation is well known in the problem-solving literature. The entire understanding process depends upon the programmer developing representations of program elements, manipulating and integrating these elements, and testing the result for correctness. This process is iterated as necessary.

Letovsky and Soloway (1986) view program understanding as the process of recognizing plans or intentions of the code. This process is more difficult when the plans are delocalized or spread over the module, or even between modules. In this model the recognition of a plan presupposes the development of a representation of the module.

The process of reverse engineering can be viewed as a formal approach to software understanding which depends on the development of a module level understanding of program function. It is defined as "... the process of obtaining high level *representations*

from source code" (Robson, Bennett, Cornelius & Munro, 1991). This process, when accomplished without software aids, involves the inspection of design and documentation artifacts, the reading of code and the symbolic execution of code. Reverse engineering goes beyond understanding the low level function of a program's component modules to include the understanding of the higher level function and purpose of the modules. This level of understanding is necessary for many maintenance problems and cannot be accomplished using program source code alone due to the loss of design information during the program coding (construction) process (Robson, Bennett, Cornelius & Munro, 1991).

Program understanding tools support the development, manipulation, and testing of representations of the application software. This set of tools support the programmer in the development of an understanding of a program. They do so through the generation of representation of a program's functions and data structures. In addition, these tools allow the programmer to test his or her mental representation through analysis of a program by "single stepping" and displaying the contents of variables.

The understanding task, as described above is composed of three sub-tasks or activities: planning, knowledge building, and bug-related (diagnostic) activities. The key, as noted above, to understanding is the development of an appropriate representation and its manipulation. The understanding tool sub-functions, analysis and representation support these sub-activities. In particular the representation function should directly support the knowledge building activity.

While support for the understanding task support may come from understanding tools, there is also a link between the understanding task and modification technology. The bug-related sub-activity, also known as diagnosis, calls for support from tools which aid

in the testing of hypotheses regarding the software being maintained. The tool function of modification (transformation) provides such aid to the programmer in testing hypotheses by allowing for the manipulation of code or data.

The support of the understanding activity by coordination functions is important but may not be immediately obvious. The coordination function supplies control and cooperation support. The programmer in a modern MIS organization is subject to a myriad of standards and procedures. Also, the programmer does not exist in a vacuum, but must work with other programmers and analysts, if for no other reason than to stay out of each others way. Changes and maintenance releases must be coordinated so as to not interfere with the work schedules of others in the organization. Programmers must seek out information regarding the activities going on around them. In addition, information may be sought from other programmers regarding recent changes to the subject system and related systems. Also, it is customary for programmers to consult their colleagues who most recently worked on the subject system.<sup>4</sup> Therefore, the support of the knowledge building sub-activity by the cooperation function is expected.

### *3.1.1.2 Supporting the Program Modification Task*

As discussed above, program understanding is not the entire story of software maintenance support. The programmer must be able to actually change software and document the effects of that change. The change cycle includes making a change in a source module, compiling the module, and testing of the changed program. The actual

---

<sup>4</sup> It has been the experience of the author that frequently a considerable amount of information regarding applications systems is not recorded either on paper or electronically. It exists in the "organizational memory", thus necessitating consultation with other programmers in the organization.

change process is fairly simple. The software which supports it consists of an ordinary text editor and library management software. The compiler program also falls into this category.

Program modification may be performed during the understanding process. In this process the programmer arrives at a point where he or she is about to test an assumption (representation) about the software to be changed. The task being performed in this process is diagnosis. In these activities a hypothesis is tested and confirmed or rejected. A change can be made to test the behavior of a program.

The modification activity is also supported by the coordination function, especially through the control function. The maintenance release and production turnover support functions, as well as those which support of programming standards, are directly involved in supporting the maintenance process.

### *3.1.1.3 Supporting the Maintenance Activity Coordination Task*

In most MIS organizations the programmer must initiate a production release process which may include documentation updates and testing for standards adherence. The later process corresponds roughly to the coordination function in the FCTM. Project management software which facilitates coordination of maintenance activities involving dependencies with other activities is also included in this category.

While many maintenance tasks or projects are relatively short in time duration and may be undertaken by a programmer or analyst working alone, no maintenance project is undertaken in isolation from other software or programmers. Although the coordination task is a normally a small part of the maintenance task, such activities ultimately have a very significant impact on the success of the maintenance project.

Software which supports these activities includes testing and version (release) management software. In addition, software which supports programming standards aids the coordination task as well.

The support of the coordination task is also facilitated by both understanding and transformation (modification) tool functions. As the coordination activities are mutually dependent on the understanding and modification activities, so also do we find that the tools which support those activities also support the coordination task, at least indirectly. Direct support of the cooperation task, however, may be provided through the representation or analysis functions when the programmer must communicate information regarding the current subject system to others in the programming organization.

### **3.1.2 Fit Moderating Variables**

In the research model for this study, there are two sets of moderating variables. These are prior experience and task complexity. These are discussed below.

#### *3.1.2.1 Prior Experience*

The model to be employed in this study varies slightly from that of Goodhue in the treatment of prior experience. In the Goodhue (1992b) model, individual characteristics is a moderating variable. As this is a very broad factor, it is necessary to narrow the focus to more specific variables which are relevant. In this study, the relevant individual characteristic variables are prior experience with both tool and task. This narrowing of focus is suggested by the work of Davis (1989) and of Guinan (1992). These studies have shown that experience or familiarity with software has a positive correlation with

usage. Familiarity with similar tasks and the capabilities of the technology are therefore posited to moderate the Perception of Fit relationships with task and technology factors.

### 3.1.2.2 Task Complexity

Task complexity is not *explicitly* included as a moderating variable in our model. It does have a bearing on maintenance tool usage in that a problem must be sufficiently complex before a programmer will resort to tool based support.

Even if a task and its supporting technology have excellent fit, the software will not be used unless the use of the tool produces an economic benefit for the user. Restated, for any task the supporting technology will be used if, and only if, its marginal benefit exceeds its marginal cost. If tool usage will save time or money *on net*, it will be used, otherwise the potential user will resort to manual methods. For those tasks which are simple (non-complex) the use of a technology may incur marginal costs which exceed the marginal benefit. In such cases use will not occur.

## 4. Research Questions

In this section a number of research questions are derived from the research model. These questions are divided into three categories: tool use, nature of fit, and nature of the maintenance task.

### 4.1 Tool Use Questions

The research model for this study is a model of those factors which lead to the use of software tools in software maintenance tasks. The following questions examine the relationships between the dependent variable usage, and the key independent variable, fit



between task and technology. The impacts of the moderating variables, experience with task, and experience with technology are also examined.

The principal research question of this study pertains to the relationship of the fit between task and technology and the use of tools. As presented above, greater degrees of fit are expected to lead to higher tool usage.

Proposition M1) For complex maintenance tasks: Higher fit between task requirements and tool functionality is associated with higher use of tool.

Goodhue (1992b) has identified experience as an important moderating variable in the establishment of fit, the independent variable in this study. As discussed above, experience has been divided into two components. Experience with a software tool serves to inform the user as to the capabilities of the software in actual use. The greater the level of experience the more likely it will be used for an appropriate task. Experience is actually a proxy for knowledge of software capabilities. The assumption is made that knowledge is obtained via prior use in actual application.

Proposition M2) For complex maintenance tasks: Greater experience with tools is associated with higher use of tool.

Prior experience with the task is understood to mean experience with the software being maintained. This type of experience is understood to moderate the relationship between task demands and fit. The higher the amount of experience with the target software, the lower the expected usage. If a programmer is experienced with software, it is expected that there will be a lower need to invoke the maintenance software. In essence, the task of understanding the application software, which is assisted by maintenance support software tool, may have occurred over several prior efforts at maintenance.

There is another form of experience that may also come into play in this area however. This is experience with the general task type. For example, making a change in field size, or debugging logic errors. This type of experience is gained through general work experience as a programmer. It is a demographic variable.

Proposition M3) For complex maintenance tasks: Lower experience with task is associated with higher use of tool.

One of the key assumptions of the model is that the task must be sufficiently complex to warrant the application of technology. In the area of software maintenance, there are many simple tasks which would not call for the use of a software tool beyond that of a simple editor. For example, a change in the title of a report is so simple even for a trainee, that nothing more than the editor and the compiler are required to make a change. (Turning over the change for production in some MIS organizations is quite another matter however.)

Proposition M4) For propositions M1, M2, & M3 above, stated relationships do not hold for non-complex tasks.

As discussed in the development of the research model, maintenance has been divided into two types, debugging and enhancement. While evidence has been presented that the Vessey (1986) Debugging Model represents a potential general model for maintenance, this conjecture has not been tested empirically. This model affords us an opportunity to test this assumption.

Proposition M5) For propositions M1, M2, & M3 above, stated relationship holds regardless of maintenance task type (debugging or enhancement); i.e., there is no interaction effect between task type and fit, between task type and experience with the tool, or between task type and experience with the task.

Although propositions M1-M3 are stated simply and independently from each other, the independent variables, fit, tool experience, and task experience, act on the dependent

variable simultaneously. That is, the independent factors may have significant two-way interactions. It is expected that a highly significant interaction effect on usage will be found for the two experience factors.

Propositions M4 & M5, respectively are concerned the moderating effects of two variables, task complexity and task type. That is, these two variables are modeled as interacting with the three earlier mentioned independent variables.

## 4.2 Nature of Task-Technology Fit Questions

This section discusses research questions which are specifically related to task-technology fit. As noted above, task-technology fit is a general notion which Goodhue (1988b; 1992b) proposes as a framework for the analysis of software usage. However, in his conceptualization, fit is a variable which can be examined independently from either task or technology characteristics.

In this study Goodhue's (1988b; 1992b) approach is expanded to include specific task and technology characteristics. The goal in moving beyond the general conceptualization of fit is to obtain a perspective of how fit occurs in the maintenance environment.

In the discussion of propositions F1 - F3, a maintenance sub-activity, e.g., understanding, modification, or coordination is posited to be supported by a corresponding software function, i.e., understanding support tools, modification support tools, or coordination enabling tools.

### Understanding

The maintenance sub-activity of Understanding is the process of establishing and testing a representation of a problem. The understanding task can be viewed as being composed

primarily of planning, knowledge building and diagnostic activities as discussed earlier in §2.4. However, the basis of these activities is the seeking of information regarding the construction and function of the system. Maintenance support software is designed to supply key information regarding program and data structures, including structure charts, hierarchy diagrams, and variable traces. The supply of this type of information is fundamental in the support of the software understanding activity. Following are the propositions related to software understanding support.

Proposition F1) The maintenance activity, Understanding<sup>5</sup>, is primarily supported by tool function, Understanding<sup>6</sup>; i.e., the two-way interaction effect on usage between the understanding activity and the understanding function is higher (stronger) than any other of the two-way interaction effects between the understanding activity and any other column of Figure 9, or the understanding function and any other row of Figure 9.

## Modification

The program modification activity is the process of actually making a change in source code and transforming that code into machine executable modules. Code modification is supported directly through the use of editors and library management tools. Modification is also indirectly supported through some compiler and linker facilities. We state the following proposition:

Proposition F2) The maintenance activity, Modification, is supported by the Maintenance tool functions; i.e., the two-way interaction effect on usage between the modification activity and the modification function is higher (stronger) than any other of the two-way interaction effects between the modification activity and any other column of Figure 9, or the modification function and any other row of Figure 9.

---

<sup>5</sup> The maintenance activity, Understanding, is composed of 3 sub-activities: Planning, Knowledge Building, and Diagnosis.

<sup>6</sup> The tool function, Understanding, is composed of Representation and Analysis sub-functions.

## Coordination

While the coordination activities are relatively unimportant in terms of the amount of gross effort involved, they are essential for the success of any maintenance project. Coordination consists of two types of activities, control and cooperation. These are supported through the use of project management software and the checkout, release, and documentation software functions. The following propositions follow in the same manner as propositions F1 and F2.

Proposition F3) The maintenance activity, Coordination, is supported by tool function Coordination<sup>7</sup>: i.e., the two-way interaction effect on usage between the coordination activity and the coordination function is higher (stronger) than any other of the two-way interaction effects between the coordination activity and any other column of Figure 9, or the coordination function and any other row of Figure 9.

## Maintenance Task Type

Once again the basic research model for this study contains the assumption that the maintenance process is general. That is to say the basic process is the same regardless of whether the task is debugging or an enhancement. The following proposition tests whether debugging or modification are supported in the same way, and whether the same sub-activities characterize both processes.

Proposition F4) For propositions F1 - F3, stated relationships hold regardless of maintenance task type (debugging or enhancement), i.e., there is no significant interaction between task type and the two-way interactions between maintenance activities and support tool functions.

## Secondary Fit

While the propositions above are formulated such that there is a primary support technology for an activity, it is clearly possible that more than one function can support

---

<sup>7</sup> The maintenance activity, Coordination is made up of sub-activities control and cooperation.

an activity. Part of the difficulty is that the various activities are not completely independent of each other. This is due to the interaction of the three major activities. For example, understanding and modification may be iterative. These activities, although in concept quite separate, occur nearly simultaneously. Thus an allowance is made software functions to have secondary or indirect support roles. A number of difficulties in separating the functions will be discussed below in chapter 3.

Proposition F5) For propositions F1 - F3, an activity will (likely) be supported by a tool function other than that stated, i.e., a significant two-way interaction may exist between non-diagonal combinations of activities and tool functions in Figure 9<sup>8</sup>·<sup>9</sup>.

### 4.3 Nature of Maintenance Task Questions

In this section, two sets of propositions are discussed. First are two propositions (T1 & T2) which derive from the earlier discussion of maintenance task type and were not explicitly covered earlier by proposition M5. The remaining group of propositions address issues relating to classical software engineering issues in the area of software maintenance.

As discussed above, one of the key questions raised in this research is whether a fundamental difference exists between debugging and enhancement. The next two propositions explore this issue on a basic level. The relative frequency of each sub activity can be expected to be the same for the different types of maintenance. A difference in activity mix would suggest a difference in process.

---

<sup>8</sup> For example, a significant two-way interaction between the understanding activity and the modification function may exist.

<sup>9</sup> The reader may recall that propositions F1, F2, and F3 stated that the "dominant" interaction would lie on the diagonal in Fig. 2.8; this proposition says that the non-diagonal interactions, while less than the diagonal interactions, may be non-zero.

Proposition T1) The proportion (relative frequency) of maintenance activities is the same for either type of maintenance task.

The remaining proposition is concerned with factors commonly identified in the software engineering literature as contributing to complexity or difficulty in maintenance. As discussed above, complexity is a key element required for tool usage to occur. Tasks which are not complex will not require the use of software tools.

Proposition T2) The proportion (relative frequency) of maintenance sub-activities is the same regardless of:

- a) type of application (on-line, batch, mixed)
- b) age of system
- c) previous maintenance history (low, high)
- d) application language type (3rd, 4th).
- e) data environment (file, database).

## 5. Research Model Summary

In summary, the research model for this study is concerned with the impact of five factors or variables on the utilization of software maintenance tools. The independent variables are task-technology fit, task experience, tool experience, task type, and task complexity. The research model can be expressed formally as follows:

$$(1) \quad Y = \alpha + \sum_{i=1}^5 \beta_i X_i + \sum_{j=1}^4 \sum_{k=j+1}^5 \beta_{jk} X_j X_k + \varepsilon$$

where:

- Y = Usage
- X<sub>1</sub> = Fit Between Task Attribute and Tool Function
- X<sub>2</sub> = Task Experience
- X<sub>3</sub> = Tool Experience
- X<sub>4</sub> = Task Type
- X<sub>5</sub> = Task Complexity

The graphical version of the model expressed in equation 1 was shown in Figure 1.

## **6. Conclusion**

In this chapter a research model for the study has been described. A number of research propositions have been developed. In the next chapter the research method for the study is discussed.



# Chapter 3

## Research Method

### 1. Introduction

This research employed the field study as its principal method. The subjects for the study were working programmer analysts performing maintenance tasks. The projects which were included in the study were selected from the existing maintenance backlog. This was done in order to achieve a high degree of external validity and generalizability.

#### 1.1 Research Sites

Three organizations agreed to participate in this study. These organizations are members of the Fortune 50 and as such they are old established firms who are leaders in their respective industries. All have large MIS applications groups who expend a large proportion of their annual budgets on software maintenance. The organizations are in different industries and are in separate geographic regions of the United States. The study will include these three sites so as to increase the external validity of the study through the elimination of organizational and local cultural effects.

Organization A is a multi-divisional financial services firm located in the Northeast. This firm's MIS application environment is typical of many installations found in the Fortune 50. Its hardware environment is based on IBM 3090 mainframes running MVS COBOL/CICS applications. The software application groups' budget is dominated by software maintenance costs. The software application groups are responsible for maintaining over 100 million lines of COBOL code.

Organization A's management has taken a pro-active approach to dealing with software maintenance costs. It has installed and is actively promoting the use of software maintenance support tools. In addition, it works with software tool vendors as a beta test site for new tools, and actively seeks to adopt leading edge software maintenance tools.

Organization B is an aerospace firm located in the West Coast of the United States. It shares several characteristics with Organization A. Its hardware / software environment employs the same basic technologies, namely IBM 3090's running MVS COBOL/CICS applications. Organization B's management is also concerned with what is viewed as an increasing cost of maintaining existing software. As a result it has provided, like Organization A, software maintenance support tools. Organization B's software application groups support over one billion lines of production COBOL code.

Organization C is a large insurance company located in the Northeast. Like Organizations A and B, Organization C's systems environment is based on IBM 3090 systems running MVS COBOL/CICS applications. This company has a very large programming staff which devotes significant resources to maintaining legacy systems.

Organizations A, B, and C have many maintenance support tools in common. Although Organization B's and Organization C's management's are less aggressive in adopting new software maintenance technologies than is Organization A's, the actual software working environment for the programmer analyst is similar in all three organizations.

## **1.2 Project Selection**

The basic unit of analysis for this study is the individual maintenance project. The definition used for "project" may vary slightly from that used by the subject organization.

Project, in the context of this study, is taken to mean a change to an existing function, or group of related functions that can be accomplished by a single programmer analyst.

The research model deals specifically with projects that are individual and non-trivial. However, it was necessary to collect data on all tasks completed during the data collection period in order to simplify the data recording requirements for the respondents and eliminate respondent selection bias. By questioning the respondent regarding the complexity of the task and the degree of involvement in the project by other programmers, we will be able to identify and segregate trivial projects and group projects. The managers of the groups which agreed to participate were asked to allow all programmers in the group to participate in the study. Programmers were asked to report on all projects to be completed during the data collection period. The data collection period was negotiated with site management and varied from 4 to 8 weeks.

Application areas included in the study consist of classical business applications, including finance, accounting, inventory, and manufacturing support. MVS COBOL/CICS was the dominant applications platform.

Selection of projects based upon the criteria described above will have the effect of controlling for effects due to a) project size, b) group process, c) application type, and d) application environment. In addition, the type of maintenance support software is the same in organizations A, B and C, although the access mode differs slightly. Thus the type of support software is held constant.

### **1.3 Data Collection**

The primary data collection method for this study was the questionnaire. The data collection period varied from 4 to 8 weeks between groups and varied slightly from

programmer to programmer within groups depending on work schedules. Participation in the study was voluntary and individuals were allowed to withdraw at any time. All participants completed at least one project, and were given the opportunity to complete additional projects.

Three questionnaires were administered to programmers. The first was a demographics questionnaire. This included questions about the programmer's background and experience level. It also included questions about the programmer's perception of the maintenance support software capabilities.<sup>1</sup> This questionnaire was given once to each participant at the beginning of the data collection period.

The second survey included questions regarding the perceived fit between the maintenance task and the maintenance support software. In addition, this survey included questions regarding the programmer's intention to use the software and the constructs directly related to intention as discussed in §2.1.<sup>2</sup> This survey was self-administered at the beginning of each project.

The third survey collected from the programmer contained actual task characteristics and tool usage questions.<sup>3</sup> Survey instrument 3 was self-administered at the completion of each project. Each participant was provided with instructions for the completion of each questionnaire, as well as an appropriate supply of questionnaires and return envelopes. Completed questionnaires were forwarded by the programmer to a person within each

---

<sup>1</sup> Denoted in the research model as variable 7.

<sup>2</sup> Denoted in the research model as variables 2 through 6.

<sup>3</sup> Denoted in the research model as variables 1 and 8.

organization who agreed to collect and forward questionnaires to the researcher. Participants were given the option of returning any questionnaire directly to the researcher.

## **2. Operationalization and Measurement of Model Variables**

The research model specified in Chapter 2 is a hybrid of two existing models, the Davis TAM model and the Goodhue Task-Technology Fit model. The Davis TAM model is specifically concerned with tool use as the dependent variable while the Goodhue model deals with the fit between task and technology and the consequences of that fit (performance) as the dependent variable.

The Davis model has an established instrument with known reliability and demonstrated validity. The questions from the Davis instrument were used in survey 2 for this study. Variations from this instrument are briefly discussed below in § 2.1.

The Goodhue (1992a) Task-Technology fit model does not have an directly associated instrument. However, Goodhue (1992b) presents an instrument which potentially could be used to assess perceived fit between task demands and tool characteristics. The later paper uses a somewhat different theoretical basis than the earlier work. However, the later paper's rationale is more refined and is compatible with the earlier paper in which the notion of fit is developed. Unfortunately, the Goodhue (1992b) instrument does not, as yet, have well established validity and reliability. It is therefore premature to rely entirely on this instrument for a measurement of fit.

The Goodhue (1992b) approach attempts a general assessment of fit without deep knowledge of the task or technology. Since one of the goals of this study is to develop

our understanding of both the maintenance task and the technology which supports it, it was necessary to develop questions which operationalize fit from a specific task and tool context. It was also necessary to develop operationalizations of experience with tools, experience with task, and maintenance complexity.

However, the Goodhue (1992b) approach to fit measurement has value in this study for research method reasons. Goodhue's approach conceives of fit as a directly accessible variable or construct. This contrasts with our approach to fit which conceives of fit as a derived variable or construct through the observation of the task and technology factors which interact or "fit". The measurement of fit with different instruments will allow us to control for instrument or method bias. This alternative operationalization of fit, as suggested by Goodhue (1992b), will be referred to as general fit.

In this section, the operationalizations of the constructs of the research model are discussed. In § 2.1 the constructs of the Davis TAM model are examined. The Goodhue (1992a) fit model is discussed in § 2.2. Finally, in § 2.3 the role of maintenance task complexity is discussed.

## **2.1 TAM Variables**

The Davis TAM model has been employed without significant modification to either the underlying theoretical rationale or the instrument itself. The Davis instrument was used in its entirety, and will include the questions pertaining to the intervening TAM variables, although the intervening variables are not needed to test the research propositions. The major TAM independent variables, Perceived Usefulness and Perceived Ease of Use, are intervening variables in the research model for this study. As such, they are not directly involved in the testing of the formal research propositions stated in Chapter 2. The items

for these constructs were included for diagnostic purposes, if necessary, and to provide data for a follow-on study. The complete instrument will be used in order to avoid potential impacts on the instruments established validity and reliability.

The Davis TAM questions were adapted for the maintenance support tool set. This involved changing only the wording of the questions to include the proper local names of the tools. Therefore, it was also necessary to have slightly different questionnaires for the different organizations. The questions for each of the following sub-sections may be found in Appendix A.

### 2.1.1 Perceived Usefulness

The Perceived Usefulness is one of the key constructs in the TAM model. This construct is the intersection between the Fit and Usage portions of the research model. The formation of perceived fit is posited to lead to Perceived Usefulness. The construct is defined as follows:

*... the degree to which a person believes that using a particular system would enhance his/her job performance. (Davis, 1989)*

This construct was assessed using the 6 questions from Davis (1989).

### 2.1.2 Perceived Ease of Use

This construct is a very important part of the TAM model. The research model for this study posits that the programmer's prior experience with the software tool largely determines the perceived ease of use. This construct is essentially co-equal in the Davis conception of the external variables which drive usage. It is defined as:

*... the degree to which a person believes that using a particular system would be free of effort. (Davis, 1989)*

It was assessed using 4 questions from Davis (1989).

### **2.1.3 Attitude Towards Tool Use**

In the Theory of Planned Behavior, attitude towards a behavior is the key predictor variable for actually engaging in the subject behavior. The work of Davis in the development of the TAM model expanded Ajzen's conception of external variables to be Perceived Usefulness and Ease of Use. Attitude toward tool use remains in the model as an intervening variable. It is defined as:

*affective response to desirability of tool use for task*

Davis (1989) and Mathieson (1991) assess this variable using three questions.

### **2.1.4 Intention To Use Tool**

While this construct is not explicitly used in research questions discussed in chapter 2, it is a key intervening variable in the Davis model. As noted above, it is usually employed as proxy for actual usage. It will be included in the survey in order to provide a measure against which to compare newly developed measures of self reported usage. It is defined as:

*the user's intent to engage in tool usage for the task*

It is measured with 3 questions from Davis (1989).

### **2.1.5 Actual Tool Use**

As noted above, the Davis TAM model does not have an established measure for actual tool usage. Intention to use the tool is the customary outcome variable in studies employing the Davis TAM model. In this study we have the opportunity to survey the



user both before and after the prospective use of the software. The Intention to Use questions will be included in questionnaire 2. Self reported actual use questions were included in questionnaire 3. For the actual use of the software, we adopt the following definition :

*the level (amount) of tool use for a particular task.*

This construct will be operationalized with questions from Guinan (1992) which assess the amount of time spent using the tool. The dependent variable, Tool Use, is computed as the mean of the use levels of each maintenance tool reported used by the programmer for a particular project. The level of use is reported as a seven point scale. Programmers were able to specify up to ten maintenance tools.

## **2.2 Task-Technology Fit**

In this study Fit is modeled as the matching of two sets of variables. This approach conceives of fit as the result of the correspondence of two variables. In this research context, fit is primarily the result of the correspondence of task and technology factors, and less a function of either task or technology alone. An alternative approach to the modeling of fit as matching is the interaction approach (Venkatraman, 1989). If we viewed the presence of maintenance tools as enhancing the maintenance process, an interaction model would be the preferred conceptualization of fit. In addition, although it still would be possible (albeit not preferable) to specify an interaction relationship, the use of a matching approach to modeling fit allows us to avoid the colinearity problems inherent in interaction models (Venkatraman, 1989).

After Venkatraman (1989), the following relationship is stated:

$$Fit = f(task, technology, |task - technology|)$$

This expression states that fit is a function of the task, the support technology, and the correspondence between these variables. We posit that of these three variables, the dominant determinant of fit is the correspondence between task and technology. This assumption will be tested empirically.

Following an approach suggested by Goodhue (1992a) fit was measured in two ways. First, a new instrument was employed to measure specific dimensions of the maintenance task and the supporting technologies. The result of these measurements are used in the calculation of specific fit between task and support technology, using the above expression.

Second, using the Goodhue (1992b) approach, general fit between task and supporting technology was measured directly. We developed an instrument to measure general fit based on Goodhue's work.

In the following section, we discuss the variables which are included in the fit portion of the research model.

## **2.3 Perceived Fit**

As noted above, we measured maintenance task - technology fit using two instruments. The first approach uses a matching approach where we will measure task and technology characteristics and compute or derive fit using statistical techniques. This we denote as specific fit. The second approach measures fit in a general fashion.

### **2.3.1 Specific (Derived) Fit**

The specific fit model which is employed in the study is, technically speaking, a lack of fit model. In such a model the measurements of the variables which are to be "fitted" are

scaled as z-scores and an absolute difference is measured. This gives a measure of lack of fit. An absolute difference of zero implies a perfect fit between the variables. High absolute differences imply that fit does not exist (Venkatraman, 1989).

This approach was employed successfully by Bourgeois (1985) in his study of environmental uncertainty and volatility, and by Alexander and Randolph (1985) in their study of nursing organization structure and performance. This approach offers the advantage of allowing an assessment of both statistical significance and magnitude of fit.

Approaches to fit which use a matching model involving subjective appraisal of high and low fit do not allow for assessment of the magnitude of fit; only the existence of fit can be determined statistically. Such an approach to fit analysis, which was used by Vessey and Galetta (1991) and Joyce, Slocum, and Von Glinow (1982), essentially treats Fit as a binary variable. While this approach does clearly separate the effects of high and low fit, it does not help when the degree of fit is not perfect. Since we conceptualize fit as a continuous variable, we employ the difference scores approach to derived fit.

The assessment of Specific Fit requires that we measure dimensions of both task and technology. These variables are discussed in §2.2.2, after the discussion of general fit. There are 9 major categories of potential fit. These 9 categories are formed from a 3 X 3 matrix (see Chapter 2, fig. 9) of the three task activities and the three technology functions. Each of the individual categories will be broken into sub-categories. For example, consider the fit of the task of understanding with the technology function of understanding. Since there are three sub-activities in understanding: planning, knowledge building, and diagnosis, and two technology sub-functions: analysis, and representation, there are, therefore, 6 sub-categories of fit between the understanding activity and the understanding function.

### 2.3.2 General Fit

Goodhue (1992b) has developed a fit assessment instrument which is not context sensitive for either task or technology. As such it does not depend on having deep knowledge of the task and is thus useful in the assessment of end user software. It is intended by Goodhue to address some of the well known deficiencies in the user satisfaction instruments (Bailey & Pearson, 1983; Ives, Olson & Baroudi, 1983). Specifically, Goodhue is seeking to address the fact that the satisfaction instruments address affect as a determinant of behavior and ignore other rationally held beliefs. For example, a person may not "like" or have positive feelings about a piece of software but may still use the software as it leads to a favorable job or task outcome. Goodhue's instrument attempts to access a person's belief system regarding the possible outcomes which result from software use.

The essential contribution of the Goodhue (1992b) approach to fit is the idea that fit can be measured independently of task or technology. Goodhue's general definition of fit is as follows:

*the degree to which an information system assists an individual in performing their task, or the fit between task requirements and the functionality of the IS environment.*

This definition contains two alternative conceptualizations of fit. The latter we use in the measurement of specific fit. The former, we will use as the basis of general fit.

We created a series of questionnaire items which measure the degree to which the maintenance tool set assists the programmer in accomplishing the understanding task, the modification task, and the coordination task. This questionnaire was more specific and tailored to the task than the prototype Goodhue instrument. However, it is much less

context dependent than the specific fit instrument. The administration of the general fit questionnaire was part of survey 3 as a measure of “achieved” fit.

Goodhue’s (1992b) model of fit contains 12 dimensions of fit which are grouped into 3 major categories. The major categories are Identification of data, Access to data, and Integration and Interpretation of data. The major and minor categories will be employed as measures of fit.

As noted above, the use of this instrument will give us a measure of fit against which to compare the results from the specific task model. The general fit instrument provides an alternative view of fit and thus a second perspective.

## **2.4 Principal Fit Variables**

As noted above, the task - technology fit model posits the matching of two major sets of variables, task and technology.

### **2.4.1 Maintenance Task**

In chapter 2, three activities which constitute the maintenance task were identified. These are understanding, modification and coordination. We use Vessey's (1986) descriptions in the development of an initial set of questions covering understanding and modification. Respondents were asked to report the relative frequency of each type of action. The work of Vessey (1985b; 1986) clearly identifies specific actions which make up the major activities of understanding and modification. Vessey recorded these actions during protocol analysis sessions. While her specific intent was to understand the differences between expert and novice debugging behaviors, she also developed a description of the actual types of actions engaged in by all maintainers. The third type of activity is

coordination. Questions to assess this dimension were developed based on interviews with maintainers (see Appendix A).

These questions were part of survey 3, which was administered after each maintenance project was completed.

### **2.4.2 Maintenance Technology**

Henderson & Coopriders (1990) provides a description of the basic functions present in design support software (CASE). The two major categories of functionality are Production, and Coordination. Since software maintenance is essentially a design problem as argued in chapter 2, we employed items from the Henderson & Coopriders instrument with appropriate modifications for terminology and minor functionality differences. The selection and modification of relevant items from Henderson and Coopriders was the starting point for the MSFM instrument development process. The programmer was asked to assess the functions available in the software tool set, and the tool set's ability to accomplish each basic function.

These questions were designed to elicit *a priori* the functionality anticipated by the programmer to be necessary and available in the tool to address maintenance problems and complete the assigned projects. These questions were included in survey 1 which was administered once at the beginning of the data collection period. Since the tool characteristics will not change during the data collection period and the programmers' perceptions of the tools are expected to remain relatively unchanged over the data collection period, these questions were administered at the beginning of the study and not for each project.

## 2.5 Fit Moderating Variables

In addition to the major fit variables, task and technology, there is also a set of fit moderating variables. These include experience with task and experience with technology.

### 2.5.1 Experience With Task

Experience with the maintenance task at hand is a factor consisting of two variables. The first variable is the experience level with the subject software. The second variable is the experience level with the type of bug to be corrected or modification to be made.

Questions to assess each of these variables were asked for each project. The level of experience is again defined as:

*... the amount of prior experience with similar tasks.*

The experience level was operationalized with variables which record the number of times the subject has worked on the system, total number of hours spent in the past, and a self report of expertise. In addition, we asked these questions about similar problems in other systems rather than about the system being maintained. These experience questions were included in survey 2 which is collected at the beginning of each project. Experience with maintenance is also a basic demographic question which were assessed in survey 1.

### 2.5.2 Experience With Software Maintenance Tool

Experience with the various types of software maintenance tools was measured once at the beginning of the data collection process. The respondents were asked to identify the packages with which he/she had experience. Experience is defined simply as:

*... the amount of prior experience with software maintenance tool*

The operationalization of experience was the number of prior hours spent using the software maintenance tool. The users were asked to estimate the amount of time spent using the tool. The users were also asked to rate their level of experience in using the software maintenance tool. In a similar manner the users rated their experience level with alternative methods, including both automated and manual. These questions were included as part of survey 1.

## **2.6 Control Variables**

The discussion of the research model in Chapter 2 and the above discussion identifies a number of independent variables, and a series of intervening and moderating variables, in the area of task and technology which we believe predict the dependent variable, software tool usage. It is possible, however, that any of a number of other environmental and subject variables which are not explicitly part of the model may explain variance observed in the dependent variable. These variables are grouped into four categories and are discussed below. The questions for each of these categories are listed in Appendix A.

### **2.6.1 Maintenance Task Type**

Software maintenance can be divided into two types, debugging and enhancement (Pennington & Grabowski, 1990). While the research model does not include task type as an independent or moderating variable, we believe that the literature is incomplete regarding the differences between these types of maintenance. Vessey (1986) asserts that her debugging function and activity models should apply to enhancement maintenance as well. While this judgment appears well founded, the question has yet to be examined empirically. Therefore, this variable is included in the study as a control variable. It was



operationalized with survey questions which were included in the assessment of task characteristics in survey 3 and in the second informant survey.

### **2.6.2 Maintenance Complexity**

A key assumption of the research model discussed in chapter 2 is that the model applies only for those tasks which are complex. The question as to what constitutes a complex maintenance task is described in the software engineering literature. For example, maintenance projects which involve large systems, extensive modifications, obsolete documentation, 2nd generation languages, etc., are considered to be complex. (Gremillion, 1984; Vessey & Weber, 1983).

The principal factors or variables which make up task complexity are task ambiguity, task variability, task uncertainty, task interdependence, and task scale (Campbell, 1988). Maintenance complexity was assessed for each project using a brief set of questions developed from the task complexity literature by Guinan (1992).

### **2.6.3 Demographics**

It is customary in most research projects to collect basic demographic information regarding the participants. The usual variables include age, sex, education level, major in school, total job experience, experience in current job type, and current job tenure. Under normal circumstances age, sex, and education level are not expected to be significant predictors of software usage. However, it is possible that the education and experience factors may become significant. It was necessary to collect data on these factors in order to be able to statistically control for their effects, if any. Basic demographic data was collected in the first survey.

#### **2.6.4 Social Norms**

Social norms represent, in one sense, the demographics of the group in which the maintenance programmer analyst exists. In our discussion in Chapter 2 of subjective norms in the TRA model, we explored the non-significance of subjective norms in the TAM model, and consequently their absence. Note that the subjective norms factor includes social norms as the maintainer perceives them. There are several potential reasons why subjective (social) norms were not significant in the development of the TAM model and instrument. One of the candidate explanations is that the norms were (are) relatively constant within a group of respondents. This was also expected to be true in this study of software maintainers. However, as our research design called for several groups of programmers in different organizations, it was prudent to collect subjective norm data. The instrumentality chosen for this factor was that of Mathieson (1991). This data was collected along with basic demographics in survey 1.

### **3. Development and Validation of Instruments**

In this section, we briefly discuss the contents of the three programmer questionnaires, the contents of the two second informant questionnaires and the procedures for their development and validation.

This study employed 5 separate questionnaires. The questions for these instruments were derived in large part from published works. A questionnaire containing selected items from survey 1 and survey 3 was administered to the second informant (the project manager) for each project. In addition, a second informant demographics and experience questionnaire is administered to the programmer's manager at the same time the

programmer is completing his or her own demographics questionnaire (survey 1). The placement of constructs in questionnaires is illustrated in figure 3.1.

The TAM Questionnaire (Davis, 1985) was used with minor modifications as noted earlier. This instrument has established reliability and demonstrated validity, and needed minimal work to prepare it for use in this study such as the changing of the names of software packages to reflect local terminology.

The area of maintenance task activities did not have an established instrument. Vessey's (1986) protocol analysis paper served as the starting point for the development of the items for this instrument for the understanding and modification activities. References in the problem solving, debugging, and program understanding literature's were also consulted for descriptions of activities in the debugging process. An extensive list of debugging activities was constructed, and from this list several items were constructed for each activity. The preliminary list of activities and items was examined by a group of 4 experienced maintainers and managers to identify both missing activities and irrelevant activities and items.

The development of the items for the assessment of maintenance tool functionality proceeded along similar lines. In this case the FCTM questionnaire (Henderson & Cooperider, 1990) served as a starting point for this activity.

### **3.1 Pre-Test of Instrument(s)**

A final pre-test of the full set of instruments was arranged with a workgroup from organization B which had members who are experienced in the use of the same tools to be used in the main study. The programmers and analysts who participated in the pre-test did not participate in the main data collection activity. The pre-test data was discarded.

Participants were asked to respond to the items based on their current maintenance activity. In the pre-test version of the instruments, scales for rating understandability, and meaningfulness were included for each item. In addition, free form comments were solicited from respondents. The manager of the group was also interviewed to identify problems with individual items and to identify potential operational problems with the instrument distribution and collection scheme.

The goals of the pre-test were to identify items which were vague or ambiguous and to conduct an operational test of the instruments. As a result of the pre-test, several items were re-written. In addition, the instructions for the administration of the instruments, and the cover pages for the instruments were re-written.

### **3.2 Threats to Validity & Reliability**

The validation of the instruments as discussed above does not address, at least completely, threats to validity due to method bias. In one sense method bias cannot be eliminated as it is inherent in any measurement, however it can be managed through a number of techniques. In the measurement of a single construct, method bias is effectively dealt with through the demonstration of convergent and discriminant validity. Employing multiple measures and multiple methods for the single construct essentially cancels out method bias effects.

Method bias becomes problematic even with measures validated as above when several related constructs are measured using the same method at the same time. In this study this type of method bias is addressed through the design of the data collection protocol. In figure 3.1 below the principal constructs (variables) are listed in the first column. The data sources for the study are: three programmer surveys, project documents, and second

informants surveys. An 'X' in a cell of this table indicates that a variable will be measured using that method. Note that surveys 1-3 are done at different times.

The primary data collection methods are the survey, and examination of project documents. The second informant and interview techniques are added to obtain a second measurement of the indicated variables in order to control for method bias. Note that specific fit is a derived (computed) measure based on task and tool characteristics. These variables are measured at different times using multiple methods. In addition, the second operationalization of fit, general fit, allows us to further control for method bias in the measurement of fit. It is not computed, but is measured in both survey 2 and survey 3.

Four of the variables in the above table are measured only in survey 2. These are the TAM variables, Perceived Usefulness, Perceived Ease of Use, Attitude towards tool use, and Intention to use the tool. The TAM model has been extensively tested and the standard items which measure these constructs show high discriminant and convergent validity. Since the properties of this instrument is well known, and these variables do not appear directly in the research questions, the survey method is sufficient. Method bias will not be an issue since the two variables most susceptible, attitude towards tool use, and intention to use the tool are not of interest in this study. They appear in the model only for completeness. The variables perceived usefulness and perceived ease of use are of interest and may be examined further than the formal research questions immediately imply. However, these two constructs represent intervening variables in the model and are measured at different times and with different methods than those variables immediately succeeding or preceding in the posited causal chain.

Method Variable	Survey 1	Survey 2	Survey 3	2nd. Informant Survey #1	2nd. Informant Survey #2
Tool Use			X		
Use Intention		X			
Specific Fit	derived		derived		
General Fit		X	X		
Task Experience			X	X	
Tool Experience	X			X	
Task Characteristics			X		X
Tool Characteristics	X				
Task Complexity			X		X
Social Norms	X				
Demographics	X				X

*Research Method - Variable Matrix  
Figure 1*

### 3.3 Item & Scale Development

The instrument development was concerned with issues of instrument reliability and validity. The use of published instruments with previously demonstrated reliability and validity was employed for several constructs. The TAM and FCTM items fall into this category. The constructs of experience with task, general maintenance experience, and task complexity, and general fit have measures which have been used in heretofore unpublished studies and have shown satisfactory reliability (Goodhue, 1992b; Guinan, 1992).

The constructs of maintenance task however did not have validated measures. The process of item development was discussed above. The use of multiple independent experts in maintenance to refine the items was sufficient to demonstrate face validity. In addition feed-back from the pre-test process was used to refine the wording of some items.

The assessment of instrument reliability was performed using Cronbach alpha coefficients (Cronbach, 1951). A summary of these results appears in Appendix A together with the final items for each construct in the model. All but two construct measures achieved minimally acceptable Alpha scores of .65 or greater ( DeVellis, 1991). Many measures achieved Alphas of .75 or greater. In the sections below, Cronbach alpha coefficients are displayed as the diagonal element of correlation tables which show Pearson Correlation coefficients for closely related constructs and variables. In addition these tables contain basic statistical descriptives.

Assessment of discriminant and convergent validity was accomplished primarily through the use of factor analysis<sup>4</sup> of construct items. For the most part, items separated cleanly into factors. It was necessary in a limited number of cases to delete one or more items from the original scale to achieve a cleaner separation of items into factors. In the sections which follow, the circumstances for each construct are briefly discussed. The results depicted show the final result after the deletion of any items, which are the constructs used on the analysis described in Chapter 4

### **3.3.1 Software Maintenance Activity Model**

The Software Maintenance Activity Model consists of a total of six types of activities, divided into three classes. The first class is Understanding, consisting of Bug-Related, Knowledge Building, and Planning Activities. The second class is Transformation activity. Finally, there is Coordination, consisting of Control and Cooperation Activities. Table 1 shows the correlations of these constructs and their descriptive statistics. All items were

---

<sup>4</sup> Factor analysis was performed using the Varimax rotation.

measured using a seven point scale. Responses for the items were averaged to produce a construct score.

Variable Descriptives & Correlations†	# of items	Mean	S. D.	1	2	3	4	5	6
1. Bug Related Activity	4	4.34	1.88	(.81)					
2. Knowledge Building Activity	5	3.31	1.59	.37**	(.78)				
3. Planning Activity	6	3.32	1.59	.38**	.44**	(.68)			
4. Transformation Activity	5	3.45	1.82	.40**	.42**	.34**	(.68)		
5. Control Activity	2	5.11	1.84	.07	-.09	-.16	.07	(.71)	
6. Cooperation Activity	4	2.60	1.96	.38**	.35**	.37**	.56**	.14	(.78)
†n=74, Cronbach's $\alpha$ on diagonal				* - Signif. LE .05	** - Signif. LE .01		(2-tailed)		

Table 1

3.3.1.1 Understanding Activities

The understanding class of activities consists of three types of activities: “bug-related”, planning, and knowledge building. Table 2 shows the factor analysis of these items.

Item	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
Bug Related-1	.86				
Bug Related-5	.78				
Bug Related-2	.78				
Bug Related-4	.54		.51		
Planning-4		.84			
Planning-3		.69			
Planning-5		.68			
Planning-6		.47			
Knowledge Building-10			.88		
Knowledge Building-8			.68		
Knowledge Building-4			.61	.55	
Knowledge Building-7				.79	
Knowledge Building-1				.78	
Planning-1					.85
Planning-2					.83
Eigenvalue	4.21	2.35	1.54	1.23	1.10
% Variance	28.1	15.7	10.3	8.2	7.3
factor loadings < .45 suppressed					

Table 2

The original list of items contains 6 items for Bug-related activity, 6 items for planning, and 16 for Knowledge Building. We used an extended list of items because no items or



items descriptions had been published for these constructs. The final set of items was based on inter-correlation analysis and factor analysis.

This analysis clearly indicates that the items for the three constructs separate cleanly from each other. However, both Knowledge Building and Planning separate into two distinct factors each. This result is not entirely surprising because the original description of these activities identified several sub-categories of activities (Vessey, 1986). We conclude that these items show adequate discriminant and convergent validity. Also note that Table 1 shows these constructs to have Cronbach  $\alpha$  ranging from .68 to .81. This indicates a reliable measure for these constructs.

#### *3.3.1.2 Transformation Activities*

Transformation activity is the second major category of maintenance activity. Factor analysis, as shown in Table 3, indicates that the items load on a single factor. All original items have been retained in the final measure. An inspection of Table 1 indicates an acceptable Cronbach  $\alpha$  of .68. We conclude that the measure is reliable and shows convergent validity. However, an inspection of Table 1 indicate that this construct shows strong and significant correlation with other constructs. This measure may have a problem in the area of discriminant validity. However, this is not conclusive. It was expected that the transformation activity and the understanding activities would be correlated due the fact that these processes appear to be dependent on each other.

Item	Factor 1
Transformation Activity - 5	.79
Transformation Activity - 1	.73
Transformation Activity - 3	.70
Transformation Activity - 4	.59
Transformation Activity - 2	.51
Eigenvalue	2.24
% Variance	44.8

Table 3

### 3.3.1.3 Coordination Activities

Coordination consists of two types of activity, cooperation and control. The initial list of items for control was reduced to two items from five. All original cooperation items were retained. The resulting factor analysis shown in Table 4, shows two cleanly separated factors. In addition, the Cronbach  $\alpha$  for Control was .71, and .78 for Cooperation.

Item	Factor 1	Factor 2
Cooperation Activity - 2	.84	
Cooperation Activity - 4	.81	
Cooperation Activity - 3	.73	
Cooperation Activity - 1	.69	
Control Activity -2		.87
Control Activity -3		.87
Eigenvalue	2.47	1.51
% Variance	41.2	25.2
factor loadings < .5 suppressed		

Table 4

We conclude that the measures for Coordination are reliable, and show acceptable evidence of convergent and discriminant validity.

Overall the items developed for the Software Maintenance Activity Model appear to be reliable. In general, there is adequate evidence of discriminant and convergent validity.

### 3.3.2 Maintenance Tool Functionality

The FCTM model provided the starting point for the development of the items for the measurement of Maintenance Tool Functionality. The descriptions of the sub-functions provided by Henderson and Coopriider (1990) were used to develop several items for sub-functions. The understanding function consists of the analysis and representation sub-functions. The Coordination Function consists of the Control and Cooperation sub-function. The transformation function is not divided into sub-functions. The final items can be found in Appendix A. Table 5 below displays the inter construct correlations and descriptive statistics. Cronbach  $\alpha$  for the scales ranged from a low of .69 to .92.

Variables Descriptives & Correlations†	# of items	Mean	S.D.	1	2	3	4	5
1. Analysis Sub-Function	2	2.39	1.95	(.69)				
2. Representation Sub-Function	3	1.93	1.72	.23	(.88)			
3. Transformation Sub-Function	3	1.84	1.77	.15	.35*	(.80)		
4. Control Sub-Function	4	2.43	2.00	.50**	.40*	.24	(.92)	
5. Cooperation Function	3	2.65	2.11	.64**	.33	.12	.47**	(.91)
†n=36, Cronbach's $\alpha$ on diagonal		* - Signif. LE .05		** - Signif. LE .01		(2-tailed)		

Table 5

#### 3.3.2.1 Understanding Functions

The Understanding Function consists of two sub-functional categories, Analysis and Representation. Table 6 displays a factor analysis of these items. Of all original items, one in the analysis sub-function category was deleted. The factor analysis indicates the factors separate cleanly. Cronbach  $\alpha$  for Analysis was .69. It was .88 for representation.

Item	Factor 1	Factor 2
Representation Sub-Function -2	.92	
Representation Sub-Function -1	.90	
Representation Sub-Function -3	.86	
Analysis Sub-Function -3		.88
Analysis Sub-Function -1		.88
Eigenvalue	2.62	1.43
% Variance	52.5	28.6
factor loadings < .86 suppressed		

Table 6

We conclude that the item scales show acceptable reliability. There is evidence to suggest that the items have acceptable discriminant and convergent reliability.

### 3.3.2.2 Transformation Function

As in the case of the factor analysis for the Transformation Activities, the factor analysis in Table 7 indicates that the Transformation Function items do not separate into factors.

This is the expected result. All original items were retained.

Item	Factor 1
Transformation Function -3	.91
Transformation Function -1	.85
Transformation Function -2	.77
Eigenvalue	2.14
% Variance	71.5

Table 7

The Cronbach  $\alpha$  for Transformation is .80. We conclude that the scale shows acceptable reliability. Since the items form a coherent factor, we may conclude that the items have convergent validity. An inspection of Table 5 indicates a general lack of correlation between the Transformation function and other functions with the sole exception of Representation. We conclude that this scale has acceptable discriminant validity.

### 3.3.2.3 Coordination Function

The coordination function items were factor analyzed as shown in Table 8. The sub-functions Control and Cooperation clearly separate into two distinct factors. All original items were retained.

Item	Factor 1	Factor 2
Control Sub-Function -2	.91	
Control Sub-Function -3	.90	
Control Sub-Function -1	.84	
Control Sub-Function -4	.73	
Cooperation Sub-Function -2		.93
Cooperation Sub-Function -3		.85
Cooperation Sub-Function -1		.85
Eigenvalue	4.61	1.22
% Variance	65.9	17.5
factor loadings < .7 suppressed		

Table 8

The Cronbach  $\alpha$  coefficients are .91 and .92. We conclude that these scales are highly reliable. In addition, since the factors separate cleanly, we conclude that the scales have acceptable discriminant and convergent validity.

### 3.3.3 Moderating Variables

The research model contains three categories or classes of moderating variables. These are Task Complexity, Task Experience, and Experience with Maintenance Tools. The approach for the analysis of the items and scales for each of these classes is described below.

#### 3.3.3.1 Task Complexity

The analysis of Task Complexity was approached in the same manner as the analysis of Maintenance Activities and Software functions described above. Originally, a multiple informant approach was taken to the measurement of these constructs. However, the

second informant measures, obtained from the team leader or manager of the programmer, proved to have poor reliability and failed to separate into factors. We concluded that these measures lacked discriminant or convergent validity. A likely explanation is that the programmer's manager lacks intimate knowledge of the actual task faced by the programmer. Table 9 shows the construct correlation coefficients and descriptive statistics for the Task Complexity items obtained from the programmers. The items, which were modified to reflect maintenance activities, were obtained from an unpublished study (Guinan, 1992).

Variables Descriptives & Correlations†	# of Items	Mean	S.D.	1	2	3	4	5
1. Task Ambiguity	4	1.93	1.38	(.81)				
2. Task Interdependence	3	5.28	1.45	-.40**	(.72)			
3. Task Scale	3	2.86	1.28	.46**	-.32**	(.72)		
4. Task Uncertainty‡	2	3.42	1.43	.39**	-.11	.55**	(.48)	
5. Task Variability	2	4.71	1.91	.26*	-.02	.27*	.49**	(.41)
†n=74, Cronbach's $\alpha$ on diagonal ‡Deleted (Discriminant Validity)				* - Signif. LE .05	** - Signif. LE .01	(2-tailed)		

Table 9

It is noted that Task Uncertainty and Task Variability show low Cronbach  $\alpha$ . We conclude that these measures are unreliable. In addition, an inspection of an item level correlation matrix showed that the Task Uncertainty items correlated heavily with many other items in different constructs. Factor analysis also confirmed that these items did not separate into a unique factor. Table 10 shows a clear separation of the remaining factors with the Task Uncertainty items deleted. All other original items are retained.

Item	Factor 1	Factor 2	Factor 3	Factor 4
Ambiguity -3	.81			
Ambiguity -1	.80			
Ambiguity -2	.73			
Ambiguity -4	.70			
Interdependence -1		.86		
Interdependence -2		.80		
Interdependence -3		.66		
Scale -3			.86	
Scale -2			.75	
Scale -1			.66	
Variability -2				.87
Variability -1				.65
Eigenvalue	4.32	1.54	1.34	1.24
% Variance	36.0	12.8	11.2	10.3
factor loadings < .5 suppressed				

Table 10

We conclude that these items show evidence of discriminant and convergent validity.

With the exception of Task Variability, the remaining scales are reliable.

### 3.3.3.2 Tool Experience

Our approach to the analysis of the tool experience constructs varies slightly from that employed above. For this construct and that of task experience, we posited that a single factor existed a priori. This differs from the earlier approaches which, with limited exceptions, assumed an underlying factor structure based on prior experience or published data. In this analysis we are performing exploratory factor analysis to identify factors within the general construct of Tool Use Experience.

Table 11 displays the factor analysis of these items.

	Tool Experience
Tool use frequency	.95
Tool experience level	.93
Tool use total hours	.49
Cronbach $\alpha$	.72
Eigenvalue	2.01
% Variance	67.0

Table 11

This analysis produced one factor. Table 12 displays the correlation coefficients and descriptive statistics for these variables.

Variables Descriptives & Correlations	# of cases	Mean	S.D.	1	2	3
1. Tool use total hours	24	2658.8	3401.5	1.00		
2. Tool experience level	34	4.51	1.04	.19	1.00	
3. Tool use frequency	36	4.47	1.35	-.03	.81**	1.00

\* - Signif. LE .05    \*\* - Signif. LE .01 (2-tailed)

Table 12

All variables in this group were measured with seven items scale questions, with the exception of total hours which is expressed in hours. This variable was computed as the total hours of use across all tools. The programmer reported frequency and level of use variables were computed as the mean levels for all tools used.

### 3.3.3.3 Programmer Experience

Programmer experience was approached in much the same manner as tool experience. However it is recognized that programmer experience may be divided into general maintenance and programming experience and experience with the system being maintained. Table 13 displays the correlation coefficients and descriptive statistics for general maintenance experience. In this case all variables are measured using seven item scales, and were obtained from the manager’s assessment of the programmers experience.



Variables Descriptives & Correlations	# Cases	Mean	S.D.	1	2	3	4
1. Exp. leading Maint.	33	3.30	2.16	1.00			
2. Formal Edu. In Dev/Maint	29	4.66	1.49	.57**	1.00		
3. Develop. Exp.	33	4.91	1.49	.65**	.50**	1.00	
4. Maintenance Exp.	33	5.48	1.20	.40*	.41*	.72**	1.00
* - Signif. LE .05 ** - Signif. LE .01 (2-tailed)							

Table 13

Table 14 displays the correlation coefficients and descriptive statistics for experience with the system being maintained.

Variables Descriptives & Correlations	# Cases	Mean	S.D	1	2	3
1. Exp. with System	70	4.21	1.97	1.00		
2. Relative Exp. with System	70	4.29	1.86	.87**	1.00	
3. Num. of Times Maintained	71	4.03	2.22	.38**	.44**	1.00
* - Signif. LE .05 ** - Signif. LE .01 (2-tailed)						

Table 14

Table 15 displays the factor analysis for experience with the specific system being maintained and for the general experience with software maintenance. All variables are measured using seven item scales. Note that the factors separate cleanly and the item scales display high Cronbach  $\alpha$ .

	General Exp.	System Exp.
Exp. leading Maint.	.84	
Formal Edu. In Dev/Maint	.80	
Maintenance Exp.	.77	
Develop. Exp.	.75	
Relative Exp. with System		.95
Exp. with System		.91
Num. of Times Maintained		.63
Cronbach $\alpha$	.74	.79
Eigenvalue	2.6	2.2
% Variance	37.2	31.4
factor loadings < .6 suppressed		

Table 15

We conclude that these items display evidence of convergent and discriminant validity. We also note that the scales are very reliable.

## 4. Data Analysis and Hypothesis Testing

In this section we discuss the statistical methods which will be used to test research propositions presented in Chapter 2, §3. Actual results are presented in Chapter 4. In general, two methods are used throughout the analysis: regression and correlation. Some propositions are examined using MANOVA.

The data analysis was conducted using SPSS. Actual SPSS run output, edited to remove extraneous information and messages, is incorporated. Note that final solutions only are displayed. The SPSS output is edited to use meaningful data names.

Regression models are analyzed using both forced entry and a stepwise methodology. Stepwise regression is employed to detect significant variables in the model which are not significant in an ordinary “forced entry” regression due to multi-collinearity between independent variables.

Unless otherwise indicated the method used in stepwise analysis is backward elimination. This method is preferred as it begins with all hypothesized relationships in the model. This method also tends to leave more variables in the final model, yielding somewhat richer explanations of the variance observed in the dependent variable. This approach also is preferred, since the goal of the regression analysis in this study is explanation rather than prediction.

### 4.1 Tool Usage

Propositions M1 through M5 are concerned with the explanation of maintenance tool usage. The key variables in these propositions, Fit, Task Experience, Tool Experience,

and Task Complexity are represented by multiple questions in the research instrumentation.

Proposition M1) Higher fit between task requirements and tool functionality is associated with higher use of tool.

Proposition M2) Greater experience with tools is associated with higher use of tool.

Proposition M3) Lower experience with task is associated with higher use of tool.

Proposition M4) Higher task complexity is associated with higher use of tool.

Proposition M5) For propositions M1, M2, M3 & M4 above, stated relationship holds regardless of maintenance task type (debugging or enhancement); i.e., there is no interaction effect between task type and fit, between task type and experience with the tool, or between task type and experience with the task.

Multiple regression analysis will then approach the explanation of usage based on several variables in the model working together rather than in isolation. The 5 propositions above will be tested first using a multiple regression model with all of the independent variables included in the propositions, the moderating variables task type and complexity, and all two-way interactions among the primary factors. The regression model will also be tested using stepwise regression to detect and minimize problems with multi-collinearity. Following is the model which will serve as the basis for our analysis.<sup>5</sup>

$$(1) \text{ToolUse} = \alpha + \beta_1 \text{LackOfFit} + \beta_2 \text{TaskExp} + \beta_3 \text{ToolExp} + \beta_4 \text{TaskExp} * \text{ToolExp} \\ + \beta_5 \text{LackOfFit} * \text{TaskExp} + \beta_6 \text{LackOfFit} * \text{ToolExp} + \varepsilon$$

In addition to the more complex models, of which equation 1 is an example, several simpler models will be examined. For example, with equation 1 as the full model, we will use a full/reduced model analysis in an effort to judge the efficiency of the simple (reduced) model:

---

<sup>5</sup> Lack of fit ( | task - tech | ) is used as a proxy for Fit; it is mathematically equivalent, and is preferred for convenience in the analysis process.

$$(2) \text{ToolUse} = \alpha + \beta_1 \text{LackOfFit} + \beta_2 \text{TaskExp} + \beta_3 \text{ToolExp} + \varepsilon$$

This is the simplest equation implied by the Goodhue Task-Technology Fit model. Other, more complicated models which include the variables in the TAM, perceived usefulness and perceived ease of use, will also be examined.

The linear-additive functional form is the most common in studies which employ the theory of reasoned action and its derivatives.<sup>6</sup> Other functional forms have been proposed (Melone, 1990). However, in this study the question of functional form is less important than the nature and identity of the variables. In the prior use of the TAM model (Adams, Nelson & Todd, 1992; Davis, 1989; Davis, Bagozzi & Warshaw, 1989; Mathieson, 1991) the model employed has been additive or unspecified. The functional form of the relationship among the variables in the Goodhue (1992b) model is unspecified. However, many examples of additive fit models exist in the organizational theory literature (Alexander & Randolph, 1985).

One of the key macro questions, proposition M5, contained in this study concerns the difference between debugging and enhancement maintenance activities. We conceptualize maintenance type, debugging or enhancement, as a binary (indicator) variable. Task type will be included in the regression model as a independent term and as an interaction term with the other independent variables. By including this indicator variable in the regression model in this manner, we are able to examine whether task type makes a difference for tool use. The following model illustrates the treatment of this variable.

---

<sup>6</sup> This set of theories include the Theory of Reasoned Action, the Theory of Planned Behavior, and the Technology Acceptance Model.

$$\begin{aligned}
 (3) \text{ToolUse} = & \alpha + \beta_1 \text{LackOfFit} + \beta_2 \text{TaskExp} + \beta_3 \text{ToolExp} + \beta_4 \text{TaskType} \\
 & + \beta_5 \text{TaskExp} * \text{ToolExp} + \beta_6 \text{LackOfFit} * \text{TaskExp} \\
 & + \beta_7 \text{LackOfFit} * \text{ToolExp} + \beta_8 \text{TaskType} * \text{LackOfFit} \\
 & + \beta_9 \text{TaskType} * \text{TaskExp} + \beta_{10} \text{TaskType} * \text{ToolExp} + \varepsilon
 \end{aligned}$$

A reduced model will also be tested without interaction terms in the same way as equation 1 was examined with equation 2 to determine the efficiency of the reduced model.

The analysis of the effect of maintenance task type is accomplished through a test of coincidence (Kleinbaum, Kupper & Muller, 1988). As in any model which employs indicator/binary variables, the actual model implies the possibility of two different regression lines. The test of the model inquires whether the two lines are statistically different. If this is shown, then the conclusion can be made that the indicator variable has a significant effect. It is expected that task type will not be shown to have a significant effect on usage.

It is expected that task complexity will have a significant effect on usage through both the main effect and interaction effects.

## 4.2 Nature of Fit

Propositions F1 - F4, from chapter 2, examine the nature of fit between task demands and tool functionalities. The principal modes of analysis of these associations will be the correlation of factor scores and regression.

Proposition F1) The maintenance activity, Understanding<sup>7</sup>, is primarily supported by tool function, Understanding<sup>8</sup>; i.e., the two-way interaction effect on usage between the understanding activity and the understanding function is higher (stronger) than any other two-way interaction effects between the understanding activity and any other column of Figure 9 (Chapter 2), or the understanding function and any other row of Figure 9 (Chapter 2).

Proposition F2) The maintenance activity, Modification, is supported by tool the Maintenance tool functions; i.e., the two-way interaction effect on usage between the modification activity and the modification function is higher (stronger) than any other two-way interaction effects between the modification activity and any other column of Figure 9 (Chapter 2), or the modification function and any other row of Figure 9 (Chapter 2).

Proposition F3) The maintenance activity, Coordination, is supported by tool function Coordination<sup>9</sup>; i.e., the two-way interaction effect on usage between the coordination activity and the coordination function is higher (stronger) than any other two-way interaction effects between the coordination activity and any other column of Figure 9 (Chapter 2), or the coordination function and any other row of Figure 9 (Chapter 2).

In these questions, the phrase "... is primarily supported by" implies that particular tool function is more useful or is used more frequently than the other functions. A significant correlation between an activity and the use of a particular function would be a strong indicator that the function supports the activity in question and, if more so than the other functions, it can be said to provide **primary** support. However, note that we are not attempting to demonstrate causal relationships in this analysis.

Proposition F4 below and proposition F5 examine the issue of the difference between debugging and simple enhancement maintenance. Regression models with indicator variables are used as described earlier. A significant effect of maintenance type would indicate that the processes debugging and modification are different. Additional

---

<sup>7</sup> The maintenance activity, Understanding, is composed of 3 sub-activities: Planning, Knowledge Building, and Diagnosis.

<sup>8</sup> The tool function, Understanding, is composed of Representation and Analysis sub-functions.

<sup>9</sup> The maintenance activity, Coordination is made up of sub-activities control and cooperation.

information as to how the debugging and modification processes differ would be available in such an analysis.

Proposition F4) For propositions F1 - F3, stated relationships hold regardless of maintenance task type (debugging or enhancement).

The next proposition is another companion for propositions F1-F3. It states that although a primary relationship may exist between function and activity, secondary relationships may exist as well. For example, the planning activity may be primarily supported by the analysis function. However, representation may also play a significant role. The analysis will be the correlation of variable scores. This proposition implies that secondary support relationships may also exist.

Proposition F5) For propositions F1 - F3, an activity will (likely) be supported by a tool function other than that stated (ex. other function may have significant regression coefficient but of lower magnitude).

### 4.3 Nature of Maintenance Task

The final set of propositions, T1-T2, are concerned with the differences between debugging and enhancement maintenance.

Proposition T1) The proportion (relative frequency) of maintenance sub-activities is the same for either type of maintenance task.

Proposition T2) The proportion (relative frequency) of maintenance sub-activities is the same regardless of software complexity factors:

- a) type of application (on-line, batch, mixed)
- b) age of system
- c) previous maintenance history (low, high)
- d) application language type (3rd, 4th).
- e) data environment (file, database).

We will test for differences in the set of variables between these two populations using MANOVA.

## **5. Conclusion**

This chapter has discussed the research method to be employed in this study. The basic operationalizations of the variables, data collection, instrument development, and data analysis procedures were discussed.



# **Chapter 4**

## **Data Analysis ~ Results**

### **1. Introduction**

This chapter presents the principal results of the study. The major focus of this chapter, and of the entire dissertation, is the first set of propositions. These five propositions investigate the association between tool usage and the explanatory variables, Task-Technology Fit, and Experience with the Software Tools. It is organized around the analysis of the three major groups of research propositions: Tool Use (Propositions M1 - M5), Maintenance Support (Propositions F1 - F4), and The Nature of Maintenance Tasks (Propositions T1 - T2). In this chapter we continue with the convention used in earlier chapters of presenting each research proposition informally, describing the expected result, rather than the as a formal null hypothesis.

### **2. Tool Use**

This section examines the propositions concerned with the use of maintenance tools. We explore below the impacts of five sets of independent variables, Task-Technology fit, Experience with Software Maintenance Tools, Experience with the Software Maintenance Task, Task Complexity, and Task Type on Tool Use, the dependent variable.

#### **2.1 Task - Technology Fit**

One of the principal theses of this study is that Task-Technology fit explains software tool utilization. Earlier in Chapter 3, we identified two methods for operationalizing Task-

Technology Fit. The first, which we label the Goodhue Method, is described by Goodhue (1992b). This method directly measures fit through 12 variables which group into three factors. These factors are Data Accessibility, Data Identification, and Data Interpretation. Scores for each factor were computed as the mean of the associated items.

The second operationalization of fit is a matching of the task requirements with the technology environment. The actual computation is accomplished as follows as suggested by Miller (1992):

$$\text{Fit} = -1 * |\text{task attribute} - \text{technology attribute}|$$

This computation is done for each task - technology pair. In this model there are a total of 26 fit pairs.

The following proposition has been advanced in chapters 2 and 3:

Proposition M1) Higher fit between task requirements and tool functionality is associated with higher use of tool.

It is tested in the two sections which follow. Note that the dependent variable is tool use.

### 2.1.1 Goodhue Method

Following below in Figure 1, we find the regression results for the proposition M1.

(Forced Entry)					
Multiple R		.50944			
R Square		.25953			
Adjusted R Square		.22123			
Standard Error		1.18531			
Analysis of Variance					
	DF	Sum of Squares		Mean Square	
Regression	3	28.56082		9.52027	
Residual	58	81.48792		1.40496	
F =	6.77617	Signif F =	.0005		
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Accessibility	-.640489	.254390	-.624912	-2.518	.0146
Identification	-.094736	.193069	-.122955	-.491	.6255
Interpretation	.523224	.165715	.525278	3.157	.0025
(Constant)	6.447138	.671643		9.599	.0000

Figure 1

The second regression for Proposition M1 follows in Figure 2. It shows the results for a stepwise regression on the same variables in Figure 1.

(Stepwise Entry)					
Multiple R		.50641			
R Square		.25645			
Adjusted R Square		.23125			
Standard Error		1.17766			
Analysis of Variance					
	DF	Sum of Squares		Mean Square	
Regression	2	28.22255		14.11128	
Residual	59	81.82619		1.38688	
F =	10.17480	Signif F =	.0002		
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Accessibility	-.735834	.163128	-.717938	-4.511	.0000
Interpretation	.501284	.158539	.503251	3.162	.0025
(Constant)	6.632321	.551998		12.015	.0000

Figure 2

Note that the two regression models used in testing this proposition, of course, agree as to the independent variables which are significant: Accessibility and Interpretation. Whether

Identification is irrelevant or redundant, in either case it cannot be said to add any explanatory ability to the regression model. As the regressions are significant, we may also conclude that tool use and fit are associated. However, the degree of the association, and its direction are inconclusive. Note that the signs of the beta coefficients for the two variables are in opposition. A positive sign implies that higher tool use is associated with a higher degree of the fit attribute. A negative sign implies the opposite. Thus higher accessibility to the tools is associated with lower tool use. Higher Interpretation power is associated with higher tool use. This phenomena is discussed further in Chapter 5. The Adjusted  $R^2$  for the model is .23.

### 2.1.2 Matching Method

In Figure 3, the regression of the computed fit against tool use is shown. Note that for this figure, and subsequent figures showing fit, a fit variable code is employed. A cross reference table of variable names and meaningful descriptions appears in Appendix B. Briefly stated, the fit pair name describes the location of the pair in the task technology matrix. For example, F11A is a fit in the upper left corner of the matrix shown in Table 2, below. In this case it corresponds to the Fit between Bug Related Activity and the Representation Function.

Note that Figure 3 shows the results of the forced entry of all fit pairs into the regression.

(forced entry)					
Multiple R	.82753				
R Square	.68480				
Adjusted R Square	.40011				
Standard Error	1.04031				
Analysis of Variance					
	DF	Sum of Squares	Mean Square		
Regression	28	72.89104	2.60325		
Residual	31	33.54955	1.08224		
F =	2.40542	Signif F = .0095			
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
F11A	.417036	.416324	.232381	1.002	.3242
F11B	-.782712	.497998	-.430535	-1.572	.1262
F11C	.369281	.514415	.210043	.718	.4782
F11D	.469997	.589323	.243316	.798	.4312
F11E	.055896	.325126	.032491	.172	.8646
F11F	.547694	.481585	.289228	1.137	.2641
F12A	-.398469	.322141	-.250283	-1.237	.2254
F12B	-.542768	.431405	-.285853	-1.258	.2177
F12C	.267380	.291090	.170480	.919	.3654
F13A	-.074460	.430250	-.046793	-.173	.8637
F13B	-.302437	.459751	-.151531	-.658	.5155
F13C	.273142	.436020	.143358	.626	.5356
F13D	.133203	.452626	.063770	.294	.7705
F13E	.157170	.426290	.084460	.369	.7149
F13F	-.743344	.464209	-.352132	-1.601	.1195
F21A	-.523841	.292366	-.299977	-1.792	.0829
F21B	.723363	.416895	.352585	1.735	.0927
F22	-.062025	.288725	-.034780	-.215	.8313
F23A	.084178	.409323	.046271	.206	.8384
F23B	-.072288	.455001	-.036527	-.159	.8748
F31A	-.539078	.325408	-.303994	-1.657	.1077
F31B	-.255544	.533426	-.143414	-.479	.6353
F31C	-.654944	.323622	-.370704	-2.024	.0517
F31D	.003622	.412346	.001809	.009	.9930
F32A	-.321480	.383167	-.176914	-.839	.4079
F32B	.729309	.339595	.477950	2.148	.0397
F33A	1.217355	.351614	.562679	3.462	.0016
F33B	.152592	.488557	.083116	.312	.7569
(Constant)	5.655046	.604687		9.352	.0000

Figure 3

Note that while the regression model<sup>1</sup> above is significant at the .01 level, and the adjusted  $R^2$  of the model is .40, only two of the independent variables are significant at the .05 level. This is highly suggestive of multi-collinearity between the independent variables.

<sup>1</sup> The regression models in this chapter employ, unless otherwise noted, pairwise deletion of missing data.

This problem is addressed using stepwise regression.<sup>2</sup> The following figure contains this regression.

(backward stepwise)					
Multiple R	.78084				
R Square	.60971				
Adjusted R Square	.51006				
Standard Error	.94015				
Analysis of Variance					
	DF	Sum of Squares	Mean Square		
Regression	12	64.89792	5.40816		
Residual	47	41.54267	.88389		
F =	6.11861	Signif F = .0000			
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
F11B	-.843660	.244888	-.464059	-3.445	.0012
F11C	.728015	.301487	.414086	2.415	.0197
F11D	.538165	.262918	.278607	2.047	.0463
F11F	.684341	.305436	.361388	2.241	.0298
F12A	-.572481	.236604	-.359582	-2.420	.0195
F13F	-.507578	.275833	-.240447	-1.840	.0721
F21A	-.521455	.228116	-.298610	-2.286	.0268
F21B	.684534	.251308	.333659	2.724	.0090
F31A	-.545564	.219768	-.307652	-2.482	.0167
F31C	-.890326	.224059	-.503932	-3.974	.0002
F32B	.565553	.231930	.370633	2.438	.0186
F33A	1.259762	.241339	.582280	5.220	.0000
(Constant)	5.978308	.417956		14.304	.0000

Figure 4

This result clearly shows that the regression is very significant and has a quite respectable adjusted  $R^2$  of .51. In addition, all of the variables in the model are significant at least at the .05 level and at best at the .0000 level, with a single exception that is significant at the .072 level. We conclude that the proposition M1 is supported. We reject the null hypothesis that there is no relationship between fit and tool use. Fit between task and

<sup>2</sup>Stepwise regressions in this chapter were accomplished, unless otherwise noted, using the backward elimination method. P(OUT) was set at .15; P(IN) was set at .10. Output was edited to show only the variables in the equation.

technology is associated with tool use. However, the signs of some of the fit variables are negative. This implies that these types of fit lead to lower tool use. Positive signs imply greater tool use with greater degrees fit. It is clear that some types of fit do produce higher use of tools.

## 2.2 Experience with Maintenance Tools

Experience with software tools used in the maintenance process has been proposed as being associated with higher levels of tool usage.

Proposition M2) Greater experience with tools is associated with higher use of tool.

Experience with tools was measured with three variables, experience level, prior level of use, and prior hours of use. Each of these variables is a composite; each is computed as the mean of the values reported for individual tools. Finally these variables are combined in the factor Tool Experience by taking the sum of the normalized scores for each variable. The first regression, Figure 5, is the forced entry regression model. The dependent variable is, as before, tool use.

(forced entry)					
Multiple R		.43544			
R Square		.18961			
Adjusted R Square		.17160			
Standard Error		1.27399			
Analysis of Variance					
	DF	Sum of Squares		Mean Square	
Regression	1	17.08864		17.08864	
Residual	45	73.03784		1.62306	
F =	10.52864		Signif F =	.0022	
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Tool Exper.	.270951	.083504	.435439	3.245	.0022
(Constant)	5.632797	.187643		30.019	.0000

Figure 5

This regression is highly significant and has an adjusted  $R^2$  of .17. The null hypothesis that there is no relationship between prior experience with tools and tool use is rejected.

Proposition M2 is supported; higher experience with tools is associated with higher tool use.

## **2.3 Experience with Maintenance Task**

Experience with the maintenance task actually consists of two kinds of experience, general maintenance experience, and experience with the system being maintained.

The principal thesis of this section is that lower amounts of task experience are associated with higher tool use.

Proposition M3a) Lower experience with the general maintenance task is associated with higher use of tool.

Proposition M3b) Lower experience with system being maintained is associated with higher use of tool.

### **2.3.1 General Experience**

General maintenance experience was reported by the programmer's manager.



(forced entry)					
Multiple R		.13113			
R Square		.01719			
Adjusted R Square		-.00208			
Standard Error		1.38217			
Analysis of Variance					
	DF	Sum of Squares		Mean Square	
Regression	1	1.70454		1.70454	
Residual	51	97.43051		1.91040	
F =	.89224		Signif F =	.3493	
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Task Exper.	-.037136	.039315	-.131126	-.945	.3493
(Constant)	6.294891	.770726		8.167	.0000

Figure 6

An inspection of Figure 6 reveals a regression model which is not significant. On this basis we cannot reject the null hypothesis that there is no relationship between general experience and tool use. Proposition M3a is not supported.

### 2.3.2 Experience With System

Experience with the system being maintained was thought to have a negative effect on the level of tool use. The following regression tests this hypothesis. While the previous model tested several aspects of experience with maintenance, including specific system experience, this model uses a measure of experience only with the system being maintained. This data was collected from the manager. Figure 7, below shows the forced entry of the variables in the model.

(forced entry)					
Multiple R		.32678			
R Square		.10679			
Adjusted R Square		.09025			
Standard Error		1.28302			
Analysis of Variance					
	DF	Sum of Squares		Mean Square	
Regression	1	10.62749		10.62749	
Residual	54	88.89202		1.64615	
F =	6.45597		Signif F =	.0140	
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
System Exper.	.087297	.034357	.326784	2.541	.0140
(Constant)	4.654877	.443683		10.491	.0000

Figure 7

Note that the Adjusted  $R^2$  value for this model is quite low from a practical viewpoint, 0.090. We reject the null hypothesis that there is no relationship between experience with the system being maintained and tool use. Higher experience with the system is associated with higher tool use. However, since the sign of the beta coefficient for the variable is positive, contrary to that expected, the proposition M3b is not supported by this result.

## 2.4 Task Complexity

Task complexity is included in the overall model as a moderating variable. As such it is treated in statistical analysis in a manner similar to independent variables. The following two regressions, shown in Figure 8 and Figure 9, test the proposition:

Proposition M4) Higher task complexity is associated with higher use of tool.

(Forced entry)					
Multiple R		.54828			
R Square		.30061			
Adjusted R Square		.23817			
Standard Error		1.17235			
Analysis of Variance					
	DF	Sum of Squares		Mean Square	
Regression	5	33.08199		6.61640	
Residual	56	76.96676		1.37441	
F =	4.81400		Signif F =	.0010	
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Ambiguity	-.183093	.132258	-.188041	-1.384	.1717
Interdependence	.239099	.116176	.257278	2.058	.0442
Scale	.366396	.152560	.348096	2.402	.0197
Variability	-.149580	.090913	-.212981	-1.645	.1055
Uncertainty	-.323896	.141514	-.344531	-2.289	.0259
(Constant)	5.461206	.850798		6.419	.0000

Figure 8

Note that this regression may involve colinearity between independent variables. The regression itself is significant at the .001 level; the two independent variables not significant the .05 level may be collinear. We turn to the stepwise analysis to clarify the situation.

(backwards elimination)					
Multiple R		.52600			
R Square		.27668			
Adjusted R Square		.22592			
Standard Error		1.18174			
Analysis of Variance					
	DF	Sum of Squares	Mean Square		
Regression	4	30.44799	7.61200		
Residual	57	79.60075	1.39650		
F =	5.45075	Signif F =	.0009		
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Interdependence	.291185	.110795	.313324	2.628	.0110
Scale	.317749	.149647	.301879	2.123	.0381
Variability	-.163677	.091065	-.233055	-1.797	.0776
Uncertainty	-.353169	.141045	-.375670	-2.504	.0152
(Constant)	5.138023	.824693		6.230	.0000

Figure 9

This regression model shows an Adjusted  $R^2$  of .23, and it is significant at the .001 level. Clearly there was colinearity between Task Ambiguity and Variability. The stepwise algorithm acknowledged this by deleting Ambiguity, thereby increasing the significance of Variability. We note that three of the variables which remain in the model are significant at the .05 level or better. We reject the null hypothesis that there is no relationship between task complexity factors and tool use. There is mixed support for the research proposition M4. The signs of the beta coefficients for the highly significant variables are mixed: two negative, two positive. Task Interdependence and Scale are associated with higher tool use, whereas Task Variability and Task Uncertainty are associated with lower tool use. Any positive sign would support the proposition that an increase in that variable enhances tool use.

Note that this model includes the task uncertainty variable. Our discussion in chapter 3 indicated that this variable was to be discarded as its measure failed to demonstrate

evidence of discriminant validity. The following regression, however, shows the result of excluding this variable.

(backward elimination)					
Multiple R		.48496			
R Square		.23519			
Adjusted R Square		.18152			
Standard Error		1.21516			
Analysis of Variance					
	DF	Sum of Squares		Mean Square	
Regression	4	25.88206		6.47052	
Residual	57	84.16668		1.47661	
F =	4.38201		Signif F =	.0037	
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Ambiguity	-.228325	.135548	-.234496	-1.684	.0976
Interdependence	.212318	.119806	.228461	1.772	.0817
Scale	.212386	.141919	.201779	1.497	.1400
Variability	-.233218	.086286	-.332071	-2.703	.0090
(Constant)	5.417853	.881645		6.145	.0000

Figure 10

Figure 10 shows the model without task uncertainty. While the model is still quite significant with an Adjusted R<sup>2</sup> of .18, note that task variability becomes significant in this model while it is not in the other models. This indicates that significant colinearity exists between the independent variables in this set of models. We do however, continue to reject the null hypothesis. Proposition M4 has mixed support. Task complexity, at least in the task interdependence aspect, is associated with higher degrees of tool use.

## 2.5 Task Type

Research proposition M5 concerns the effect of task type.

Proposition M5) For propositions M1, M2, M3 & M4 above, stated relationship holds regardless of maintenance task type (debugging or enhancement); i.e., there is no interaction effect between task type and fit, between task type and experience with the task, or between task type and experience with the tool.

There are two types of maintenance tasks, debugging and enhancement. Each project was identified by the manager as being in one of these two categories. Task type is coded as an indicator variable. In order to test this proposition, it is necessary to re-visit the analysis of the individual propositions. Testing the significance of an indicator variable requires that the indicator variable, in this case Task Type, be introduced into the model as an independent variable, and as an interaction term with other independent variables. In the models which follow, this has been done. In order to simplify the analysis, the interaction terms were limited to those independent variables previously identified as significant. This decision was made on the assumption that components identified as significant would be more likely to interact in the more complex models.

### **2.5.1 Fit - Task Type Interaction**

Figure 11 shows the regression model for fit and the effect of task type on Tool Usage. It is a forced entry model. Note that in this and the following figure, we use the fit variable codes. Please see Appendix B for the variable and identifier cross-reference table.

(forced entry)					
Multiple R		.79734			
R Square		.63575			
Adjusted R Square		.39639			
Standard Error		1.04353			
Analysis of Variance					
	DF	Sum of Squares	Mean Square		
Regression	23	66.52293	2.89230		
Residual	35	38.11359	1.08896		
F =	2.65602	Signif F =	.0045		
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
F11B	-.502453	.308435	-.276377	-1.629	.1123
F11C	.340248	.323575	.193529	1.052	.3002
F11D	.932754	.284040	.482885	3.284	.0023
F11F	.111381	.283726	.058818	.393	.6970
F12A	.036655	.312824	.023023	.117	.9074
F12B	-.267857	.290937	-.141069	-.921	.3635
F31A	-.377017	.291954	-.212606	-1.291	.2050
F31C	-1.156529	.270556	-.654605	-4.275	.0001
F32B	.240357	.306671	.157517	.784	.4385
F33A	1.199505	.276019	.554428	4.346	.0001
Task Type	.707892	3.325034	.212336	.213	.8326
F11B*TTtype	-.227283	1.112624	-.080944	-.204	.8393
F11C*TTtype	-.186057	1.654977	-.084106	-.112	.9111
F11D*TTtype	-.151985	1.807589	-.050388	-.084	.9335
F11F*TTtype	.419991	1.195961	.116201	.351	.7276
F12A*TTtype	-.653522	1.366917	-.358317	-.478	.6356
F13F*TTtype	-.107542	1.942473	-.028606	-.055	.9562
F21A*TTtype	-.204474	1.872197	-.078646	-.109	.9137
F21B*TTtype	.570477	1.197101	.189540	.477	.6366
F31A*TTtype	-.322594	1.734202	-.094394	-.186	.8535
F31C*TTtype	.603236	.954523	.266425	.632	.5315
F32B*TTtype	.992818	2.306867	.307994	.430	.6696
F33A*TTtype	-.063197	1.336186	-.018708	-.047	.9625
(Constant)	5.867574	.518255		11.322	.0000

Figure 11

This regression is significant at the .0045 level. However, none of the interaction terms are significant. We test the model further below using a stepwise method to detect and clarify multi-collinearity.

(backwards elimination)					
Multiple R		.76095			
R Square		.57905			
Adjusted R Square		.52127			
Standard Error		.92934			
Analysis of Variance					
	DF	Sum of Squares		Mean Square	
Regression	7	60.58966		8.65567	
Residual	51	44.04685		.86366	
F =	10.02203	Signif F =	.0000		
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
F11B	-.616815	.190601	-.339282	-3.236	.0021
F11D	1.024450	.196813	.530355	5.205	.0000
F31A	-.271067	.166374	-.152859	-1.629	.1094
F31C	-.931638	.177277	-.527315	-5.255	.0000
F33A	1.112616	.214709	.514267	5.182	.0000
F12A*TTtype	-.899913	.260318	-.493409	-3.457	.0011
F32B*TTtype	1.012233	.458846	.314017	2.206	.0319
(Constant)	5.573535	.324625		17.169	.0000

Figure 12

We note that this regression model is highly significant and displays an Adjusted  $R^2$  of .521. However, while the model is significant overall, the task type variable did not remain in the model. Two of the interaction terms which remain are significant at the .01 level. Note that Task Type does not appear as a main effect, but does have an impact through two 2-way interactions. We reject the null hypothesis that there is no relationship between task type and tool use. This does not support the research proposition M5.

### 2.5.2 Tool Experience - Task Type Interaction

We test the effect of Task Type with Tool experience variables in the same manner. Figure 13 shows the forced entry of the variables into the model. Figure 14 shows the corresponding stepwise analysis.



```
(forced entry)
Multiple R          .42137
R Square           .17755
Adjusted R Square  .11737
Standard Error     1.32534

Analysis of Variance
                   DF      Sum of Squares    Mean Square
Regression         3         15.54698         5.18233
Residual          41         72.01710         1.75651

F =          2.95035      Signif F = .0438

----- Variables in the Equation -----
Variable          B          SE B      Beta      T      Sig T
Tool Exper.       .248256   .096259   .397683   2.579   .0136
Ttype * Tool Ex. -.026682   .599563  -.012382  -.045   .9647
Task Type         .238099   .959797   .068275   .248   .8053
(Constant)        5.611170  .229805                24.417  .0000
```

Figure 13

This model is significant, but the task type variable and the interaction term are not significant.

```
(stepwise)
Multiple R          .41784
R Square           .17459
Adjusted R Square  .15539
Standard Error     1.29647

Analysis of Variance
                   DF      Sum of Squares    Mean Square
Regression         1         15.28780         15.28780
Residual          43         72.27629         1.68084

F =          9.09531      Signif F = .0043

----- Variables in the Equation -----
Variable          B          SE B      Beta      T      Sig T
Tool Exper.       .260839   .086490   .417839   3.016   .0043
(Constant)        5.654531  .194468                29.077  .0000
```

Figure 14

The models presented in Figure 13 and Figure 14 agree. Neither task type nor the interaction term are significant. We cannot reject the null hypothesis that there is no

relationship between task type and tool use. Task type is not associated with tool use. This supports the proposition M5 that here is no significant effect from task type.

### 2.5.3 Task Experience - Task Type Interaction

The following regression model, illustrated by Figure 15, shows the treatment of task type and task experience.

(forced entry)					
Multiple R	.27588				
R Square	.07611				
Adjusted R Square	.01586				
Standard Error	1.37198				
Analysis of Variance					
	DF	Sum of Squares	Mean Square		
Regression	3	7.13313	2.37771		
Residual	46	86.58748	1.88234		
F =	1.26317	Signif F = .2981			
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Task Type	-3.519989	3.253631	-.987761	-1.082	.2850
Task Exper	-.030335	.041786	-.107730	-.726	.4715
Ttype* Task Ex.	.248236	.191322	1.176172	1.297	.2009
(Constant)	6.118857	.833166		7.344	.0000

Figure 15

This figures shows a regression which is not significant. We cannot reject the null hypothesis that there is no effect of task type. Task type, in this model does not make a difference in tool use. This result supports the research proposition.

### 2.5.4 System Experience - Task Type Interaction

This analysis was repeated using system experience rather than general experience. The result is similar. The first regression model (Figure 16) shows the forced entry of all variables into the model.

(Forced entry)

Multiple R	.34949			
R Square	.12214			
Adjusted R Square	.07051			
Standard Error	1.30750			

Analysis of Variance				
	DF	Sum of Squares	Mean Square	
Regression	3	12.13121	4.04374	
Residual	51	87.18699	1.70955	

F = 2.36538      Signif F = .0818

----- Variables in the Equation -----

Variable	B	SE B	Beta	T	Sig T
Task Type	1.528242	2.033387	.438634	.752	.4558
System Exper.	.092904	.038833	.345995	2.392	.0205
Ttype * Sys. Ex.	-.090442	.133030	-.405441	-.680	.4997
(Constant)	4.575167	.472674		9.679	.0000

Figure 16

The second regression shows the stepwise regression result (Figure 17).

(backward elimination)

Multiple R	.33414			
R Square	.11165			
Adjusted R Square	.09489			
Standard Error	1.29024			

Analysis of Variance				
	DF	Sum of Squares	Mean Square	
Regression	1	11.08856	11.08856	
Residual	53	88.22965	1.66471	

F = 6.66095      Signif F = .0127

----- Variables in the Equation -----

Variable	B	SE B	Beta	T	Sig T
System Exper.	.089720	.034763	.334136	2.581	.0127
(Constant)	4.640773	.446738		10.388	.0000

Figure 17

Clearly there is no interaction effect between task type and experience with the system.

The proposition is supported.

### 2.5.5 Task Complexity- Task Type Interaction

Figure 18, and Figure 19 illustrate the examination of the effect of task type with task complexity. As done in the previous analysis, the earlier figure shows the forced entry model and the later, the stepwise model.

(forced entry)					
Multiple R	.58953				
R Square	.34755				
Adjusted R Square	.19484				
Standard Error	1.20522				
Analysis of Variance					
	DF	Sum of Squares	Mean Square		
Regression	11	36.36597	3.30600		
Residual	47	68.27055	1.45256		
F =	2.27597	Signif F = .0253			
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Ambig*TType	-.294487	.452004	-.188534	-.652	.5179
Ambiguity	-.172391	.162980	-.177050	-1.058	.2956
Interdependence	.329747	.183221	.354818	1.800	.0783
Interd*TType	-.118339	.274226	-.185962	-.432	.6681
Task Type	1.199672	1.831727	.359848	.655	.5157
Scale*TType	-.765532	.537623	-.740185	-1.424	.1611
Scale	.449844	.192886	.427377	2.332	.0240
Uncertainty	-.352417	.157776	-.374870	-2.234	.0303
Uncert*TType	.758059	.634587	.744245	1.195	.2382
Variab*TType	.063220	.245850	.088297	.257	.7982
Variability	-.144524	.111760	-.205783	-1.293	.2023
(Constant)	4.710925	1.397803		3.370	.0015

Figure 18

This model is statistically significant at the .025 level. Two variables, Scale and Task Uncertainty are significant at the .05 level. Task Interdependence is significant at the .078 level. We proceed to an inspection of the stepwise model.

(Backward elimination)				
Multiple R		.52600		
R Square		.27668		
Adjusted R Square		.22310		
Standard Error		1.18389		
Analysis of Variance				
	DF	Sum of Squares	Mean Square	
Regression	4	28.95055	7.23764	
Residual	54	75.68596	1.40159	
F =	5.16387	Signif F =	.0013	
----- Variables in the Equation -----				
Variable	B	SE B	Beta	T Sig T
Interdependence	.291185	.113831	.313324	2.558 .0134
Scale	.317749	.153747	.301879	2.067 .0436
Uncertainty	-.353169	.144910	-.375670	-2.437 .0181
Variability	-.163677	.093560	-.233055	-1.749 .0859
(Constant)	5.138023	.846713		6.068 .0000

Figure 19

This model is significant at the .001 level. Note that the task-type variable and the interaction terms were eliminated and do not appear in the final model. We therefore cannot reject the null hypothesis that there is no effect of task type on tool use. The research proposition, M5 is supported by this result.

## 2.6 Full Model (Main Effects only)

In the earlier portions of this section, we examined the effects of the independent variables on tool use as “stand-alone” or simple models, although many variables in the same class or group are examined simultaneously using multiple regression. In this sub-section, we examine the effects of several groups of variables simultaneously using multiple regression. The independent variables selected as “candidates” for this model were those identified in earlier models as significant at least at the .05 level. Please see appendix B for the fit variable cross-reference table.

(Forced entry)					
Multiple R		.81523			
R Square		.66461			
Adjusted R Square		.50232			
Standard Error		.94755			
Analysis of Variance					
	DF	Sum of Squares	Mean Square		
Regression	15	55.15420	3.67695		
Residual	31	27.83338	.89785		
F =	4.09528	Signif F =	.0004		
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Sys. Exper.	-.011791	.068121	-.044483	-.173	.8637
Tool Exper	.210852	.179603	.367487	1.174	.2493
Variability	-.064863	.100573	-.092357	-.645	.5237
F11B	-.667382	.302146	-.367096	-2.209	.0347
F11C	.528853	.363756	.300805	1.454	.1560
F11D	.387289	.328807	.200499	1.178	.2478
F11F	.570832	.399637	.301446	1.428	.1632
F12A	-.006334	.383819	.003979	.017	.9869
F13F	-.573755	.365323	-.271796	-1.571	.1264
F21A	-.396580	.269977	-.227101	-1.469	.1519
F21B	.429624	.312640	.209409	1.374	.1792
F31A	-.335499	.285580	-.189193	-1.175	.2490
F31C	-.695726	.279439	-.393787	-2.490	.0184
F32B	.279866	.321006	.183409	.872	.3900
F33A	1.169676	.310956	.540641	3.762	.0007
(Constant)	6.681795	1.103012		6.058	.0000

Figure 20

Figure 20 shows the full model, with forced entry of all variables. We note that while the overall model is highly significant, there are several variables which are not significant at the .05 level. The following figure shows the same model analyzed using stepwise regression.

(Backward Elimination)					
Multiple R		.79938			
R Square		.63902			
Adjusted R Square		.52556			
Standard Error		.92516			
Analysis of Variance					
	DF	Sum of Squares		Mean Square	
Regression	11	53.03041		4.82095	
Residual	35	29.95717		.85592	
F =	5.63248		Signif F =	.0000	
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Tool Exper.	.216755	.067005	.377776	3.235	.0027
F11B	-.608834	.286322	-.334892	-2.126	.0406
F11C	.668503	.230751	.380236	2.897	.0065
F11F	.803947	.297042	.424550	2.707	.0104
F13F	-.756132	.274360	-.358190	-2.756	.0092
F21A	-.450158	.241733	-.257782	-1.862	.0710
F21B	.557538	.254876	.271758	2.187	.0355
F31A	-.349993	.205969	-.197366	-1.699	.0981
F31C	-.800131	.246537	-.452881	-3.245	.0026
F32B	.302971	.189524	.198551	1.599	.1189
F33A	1.296777	.266880	.599389	4.859	.0000
(Constant)	6.160637	.463829		13.282	.0000

Figure 21

We note that this model is very highly significant at the .0000 level. In addition, the Adjusted  $R^2$  for the model is .525. In addition, all but 3 of the variables remaining in the model are significant at the .05 level. We are able to reject the null hypothesis that these variables have no effect on Tool Use, the dependent variable. Note that this result is consistent with earlier results where single groups of variables were examined separately. The propositions regarding fit, tool experience, task experience, and task complexity, are not contradicted by this model. It should be noted that the resulting model shown in Figure 21 is more parsimonious than the earlier models taken as a group. This may be explained by multi-collinearity effects between the independent variables in the model.

## 2.7 Full Model (Test of Effect of Non-Independence of Cases)

The design of this study allows programmers to complete more than one project. Strictly speaking, this implies that the cases (individual projects) may not be independent.

Independence of cases is a key assumption in regression model analysis. We cannot ignore the possibility that this violation has a significant effect on the model. In order to test this “proposition” we selected one case per programmer, and tested the model developed above and shown in Figure 21. The result follows in Figure 22.

(backward elimination)					
Multiple R	.99739				
R Square	.99480				
Adjusted R Square	.97918				
Standard Error	.14407				
Analysis of Variance					
	DF	Sum of Squares	Mean Square		
Regression	12	15.86830	1.32236		
Residual	4	.08302	.02075		
F =	63.71299	Signif F =	.0006		
----- Variables in the Equation -----					
Variable	B	SE B	Beta	T	Sig T
Sys. Exper.	.175121	.012821	.804528	13.659	.0002
Tool Exper.	.313349	.028501	.647584	10.994	.0004
F11B	-.994760	.084912	-.662001	-11.715	.0003
F11C	.508210	.106315	.336421	4.780	.0088
F11D	1.860799	.105662	1.147214	17.611	.0001
F11F	-.689587	.077476	-.498080	-8.901	.0009
F12A	-.333023	.095471	-.284695	-3.488	.0252
F13F	1.276024	.095167	.713678	13.408	.0002
F21B	-1.466373	.106874	-.871624	-13.721	.0002
F31A	-1.508706	.095510	-1.347285	-15.796	.0001
F31C	.629591	.074166	.502332	8.489	.0011
F32B	1.150388	.089502	1.180698	12.853	.0002
(Constant)	3.529547	.215255		16.397	.0001

Figure 22

This result shows a regression model which is very highly significant. We note that the results shown in Figure 22 and Figure 21 differ only slightly. In the later model, we see that system experience was significant whereas it did not enter in the earlier model. This result suggests that system experience may be collinear with another variable, possibly tool



experience. We conclude that the potential violation of the independence of cases assumption has negligible effect.

### 3. Maintenance Support

In this section we examine the relationships between Maintenance Activities required by a maintenance task and the Technology Functions available to the programmer to support the task at hand. The principal indicator of support is the correlation of the Maintenance Activity with a Support Function. Two tables show the Pearson Correlation Coefficients for these relationships. Table 1 shows the correlation coefficients for the highest level of association.

	Understanding Function	Modification Function	Coordination Function
Understanding Activity	.2334 <i>P</i> = .047	.5458 <i>P</i> = .000	.4927 <i>P</i> = .000
Modification Activity	.3596 <i>P</i> = .002	.0576 <i>P</i> = .629	.1300 <i>P</i> = .273
Coordination Activity	.4404 <i>P</i> = .000	.5355 <i>P</i> = .000	.6873 <i>P</i> = .000

Table 1

Individual scores for the variables in Table 1 were computed as a mean of all items for that category. Thus, the Understanding Activity score is the mean of all items in the Bug-Related, Planning, and Knowledge Building Activities.

	Analysis	Representation	Transformation	Control	Cooperation
Bug-Related	.2280 P= .052	.2115 P= .075	.3982 P= .000	.2295 P= .051	.3382 P= .003
Planning	.0074 P= .951	.0771 P= .519	.3419 P= .003	.1734 P= .142	.3834 P= .001
Knowledge Building	.0121 P= .919	.0964 P= .420	.4169 P= .000	.3889 P= .001	.3927 P= .001
System Modification	.2687 P= .022	.2922 P= .013	.0576 P= .629	.1207 P= .309	.1104 P= .352
Control Compliance	.2896 P= .013	.0061 P= .960	.0741 P= .531	.1760 P= .136	.0901 P= .448
Cooperative Activity	.1781 P= .132	.3778 P= .001	.5612 P= .000	.6084 P= .000	.5856 P= .000

Table 2

Table 2 shows the correlations at the next level of analysis; Table 2 is a disaggregation of the results shown in Table 1. Individual scores are a mean of the items for a category.

Following are discussions of the four major propositions concerning the relationships between the Maintenance Activities and Support Functions.

### 3.1 Understanding

The following proposition has been proposed regarding Understanding support relationship:

**Proposition F1)** The maintenance activity, Understanding is primarily supported by tool function. Understanding, i.e., the two-way interaction effect on usage between the understanding activity and the understanding function is higher (stronger) than any other two-way interaction effects between the understanding activity and any other column of Figure 9 (Chapter 2), or the understanding function and any other row of Figure 9 (Chapter 2).

This proposition states that we expect that the highest correlation between the Understanding Activity should be the Understanding Function. Inspection of Table 1 shows that this is not the case, although a strong and very significant correlation between the Understanding Activity and Function does exist. Further examination of Table 2 shows a particularly strong correlation between Knowledge Building activity and the

Cooperation Function. In addition, strong and significant associations exist between all of the Understanding sub-activities and the Transformation Function. The proposition, F1 is not supported in the first level of analysis in the sense that the strongest association is not as proposed. We do, however, find a strong and significant association does exist between Understanding Activities and the Understanding Support Functions.

### 3.2 Modification

The following proposition is advanced concerning the relationship between transformation Activity and the Transformation Support Function.

Proposition F2) The maintenance activity, Modification, is supported by tool the Maintenance tool functions: i.e., the two-way interaction effect on usage between the modification activity and the modification function is higher (stronger) than any other two-way interaction effects between the modification activity and any other column of Figure 9 (Chapter 2), or the modification function and any other row of Figure 9 (Chapter 2).

Inspection of Table 1 indicates that the association between this Activity and Function is weak and not statistically significant. The proposition is not supported. However we do note that there are significant associations between the Modification activity and both Understanding functions, Analysis and Representation.

### 3.3 Coordination

The following proposition is advanced concerning the relationship between the Coordination Activities and the Coordination Support Function.

Proposition F3) The maintenance activity, Coordination, is supported by tool function Coordination, i.e., the two-way interaction effect on usage between the coordination activity and the coordination function is higher (stronger) than any other two-way interaction effects between the coordination activity and any other column of Figure 9 (Chapter 2), or the coordination function and any other row of Figure 9 (Chapter 2).

Inspection of Table 1 reveals a strong and statistically significant relationship between these functions and activities. Other correlations are significant especially those with the Understanding Functions, but none quite as strongly as the primary relationship. Table 2 shows the relationships in more detail. The proposition F3 is strongly supported.

### 3.4 “Minor” Support Relationships

The following proposition concerns support relationships which are significant and “off the diagonal”.

Proposition F4) For propositions F1 - F3, an activity may be supported by a tool function other than that stated (ex. other function may have significant regression coefficient but of lower magnitude).

Cognitive Fit Theory, as discussed in Chapter 2, predicts that the primary support for a particular class of activities will come from the corresponding functionality. However, the major thesis of this section is that while the primary support relationships are “on the diagonal” in the support, there may also be other significant relationships. As noted in Chapter 2, activities may be supported by other than the primary, or expected, functionalities. This effect is possible since most of the activities engaged in by programmers during the maintenance process are dependent upon other activities. Thus the support of an activity by its primary, or expected, support tool also supports any activity which depends upon it. For example, the knowledge building task may require that a programmer communicate or cooperate with another programmer. A technology functionality which supports the communication process, such as e-mail, also supports, indirectly, the knowledge building activity.

Inspection of Tables 1 and 2 indicate that significant “off diagonals” do exist. Especially noteworthy is the correlation of the Understanding Activity with the Coordination

function, particularly Knowledge Building with Cooperation. Also, notable associations exist between all three Understanding activities and the Transformation function. This result is expected because transformations or manipulations of representations are required during the understanding process (Vessey, 1986). There is also a significant association between the Cooperation sub-activity and the Representation sub-function. This association can be understood in a similar manner. It appears that the Representation sub-function supports the programmer in cooperating with or supplying information to other programmers. Finally we note the significant associations between the Modification Activity and both Understanding sub-functions. Once again the result should be understood in the context of the mutual dependence of the Understanding and Modification activities and their mutual support by the corresponding tool function Understanding and Transformation.

The proposition F4 is clearly supported.

#### **4. Nature of Maintenance Task**

One of the theses of this study is that the Maintenance Task is the same for both debugging and enhancement maintenance. We specifically propose:

Proposition T1) The proportion (relative frequency) of maintenance sub-activities is the same for either type of maintenance task.

The analysis of this proposition is accomplished using MANOVA. The various maintenance activities are the dependent variables in the model while task type is the independent variable. Figure 23 presents the analysis results.

EFFECT .. Task Type						
Multivariate Tests of Significance (S = 1, M = 2, N = 30 1/2)						
Test Name	Value	Exact F	Hypoth. DF	Error DF	Sig. of F	
Pillais	.10069	1.17567	6.00	63.00	.331	
Hotellings	.11197	1.17567	6.00	63.00	.331	
Wilks	.89931	1.17567	6.00	63.00	.331	
Roys	.10069					
Note.. F statistics are exact.						
Univariate F-tests with (1,68) D. F.						
Variable	Hypoth. SS	Error SS	Hypoth. MS	Error MS	F	Sig. of F
Bug Related	3.83627	68.38413	3.83627	1.00565	3.81472	.055
Planning	.13158	69.75089	.13158	1.02575	.12828	.721
Knowledge Build.	.10882	70.76380	.10882	1.04064	.10457	.747
Transformation	.70000	63.77417	.70000	.93786	.74638	.391
Control	.63452	67.76311	.63452	.99652	.63673	.428
Cooperation	.15050	71.66781	.15050	1.05394	.14280	.707

Figure 23

The interpretation of this result indicates that there is no difference between the two types of projects. The multivariate analysis shows that there is no significant overall difference, while none of the Univariate tests show a significant difference at the .05 level. Stated formally, the null hypothesis that there is no difference between debugging and enhancement maintenance cannot be rejected. The proposition is supported.

The final research proposition is that there will be no differences between enhancement and debugging maintenance based on a number of complexity factors.

Proposition T2) The proportion (relative frequency) of maintenance sub-activities is the same regardless of software complexity factors:

- a) type of application (on-line, batch, mixed)
- b) age of system
- c) previous maintenance history (low, high)
- d) application language type (3rd, 4th).
- e) changes to data structure.
- f) stand alone application
- g) observance of formal procedures

The following MANOVA analysis addresses this proposition.

EFFECT ..Task Type						
Multivariate Tests of Significance (S = 1, M = 3, N = 6)						
Test Name	Value	Exact F	Hypoth. DF	Error DF	Sig. of F	
Pillais	.53106	1.98182	8.00	14.00	.126	
Hotellings	1.13247	1.98182	8.00	14.00	.126	
Wilks	.46894	1.98182	8.00	14.00	.126	
Roys	.53106					
Note.. F statistics are exact.						
Univariate F-tests with (1,21) D. F.						
Variable	Hypoth. SS	Error SS	Hypoth. MS	Error MS	F	Sig. of F
Times Maint.	220.00104	9809.73810	220.00104	467.13039	.47096	.500
Stand Alone System	.10352	3.80952	.10352	.18141	.57065	.458
System Type	.20290	16.66667	.20290	.79365	.25565	.618
Formal Procedure	.03727	2.57143	.03727	.12245	.30435	.587
New programs	.81159	4.66667	.81159	.22222	3.65217	.070
Files Changed	.14907	4.28571	.14907	.20408	.73043	.402
System Age	15.83172	69.68519	15.83172	3.31834	4.77097	.040
Prog. Language	.10352	11.80952	.10352	.56236	.18408	.672

Figure 24

This result indicates that taken as a group, these factors do not distinguish between enhancement and debugging maintenance. An inspection of the Univariate tests indicates that only system age is a significant difference between the two project types. The proposition is supported.

## 5. Summary of Results

The following table summarizes the results of the data analysis, organized by research proposition.

Research Proposition	Topic	Result
M1	<i>Higher fit is associated with higher tool use</i>	Supported*
M2	<i>Higher experience with tools is associated with higher tool use</i>	Supported
M3a	<i>Lower experience with maintenance is associated with higher tool use.</i>	Not Supported
M3b	<i>Lower experience with the system being maintained is associated with higher tool use.</i>	Not Supported
M4	<i>Higher task complexity is associated with higher tool use.</i>	Supported*
M5	<i>For propositions M1 -M4, relationships hold regardless of task type.</i>	Supported for propositions M2 & M4
F1	<i>Understanding activities are primarily supported by understanding technology</i>	Supported in sub-activity / sub-function analysis*
F2	<i>Transformation activities are primarily supported by transformation technology</i>	Not Supported
F3	<i>Coordination activities are primarily supported by Coordination technology</i>	Supported
F4	<i>For propositions F1 - F3, other "off - diagonal" support is significant.</i>	Supported
T1	<i>Maintenance activities are the same for either type of maintenance task.</i>	Supported
T2	<i>Maintenance activities are the same regardless of software complexity factors.</i>	Supported
* Each of these propositions involves several "dimensions" for the independent variables. For each of these propositions, several but not all of the (correlated) dimensions were highly significant in the appropriate direction.		

Table 3

The results summarized above in Table 3 are discussed in detail in Chapter 5.



# **Chapter 5**

## **Discussion and Conclusion**

### **1. Introduction**

In this chapter we discuss the results of the tests of the research propositions presented in Chapter 4. This discussion follows below in section 2. The remainder of this chapter is divided into two major sections: section 3 which deals with the limitations of the study, and section 4 which describes the research contribution and the implications for managers.

### **2. Research Questions**

In this section, we discuss the major findings of the study. The discussion will follow the general order of research propositions as developed in Chapter 2, and tested in Chapter 3.

#### **2.1 Task-Technology Fit**

One of the three major theses of this study is that the fit between a maintenance task and the available maintenance support software is associated with maintenance support software tool use. We have clearly shown that task-technology fit is associated with higher levels of software tool use for several types of fit. However, our results indicate that certain types of fit are associated with lower levels of tool use. This is contrary to our original expectation that fit, at least for the major types of fit, would be associated with higher amounts of tool use.

A positive effect of fit on software maintenance tool use was found for the types of fit listed in Table 1. We find that there was support for our expectation that fit would be associated with higher amounts of tool use for two of the three major fit (on diagonal

elements) dimensions. Only the fit between transformation activities and functions is not associated with software use. In addition, we see that two of the minor, or “off-diagonal” fits are associated with higher software maintenance tool use.

Name	Description
F11c	Fit of KB with Analysis
F11d	Fit of KB with Representation
F11f	Fit of Planning with Representation
F21b	Fit of Modification with Representation
F32b	Fit of Control with Transformation
F33a	Fit of Cooperation Activity and Function

*Table 1*

These results were expected, based on the model developed in Chapter 2. Cognitive fit theory (Vessey, 1991), predicts the on-diagonal results which we observe above. Recall that this theory states, in effect, that the best or primary support for an activity will come from those tool functions which are designed specifically for that activity. The fit of the Understanding activities with the Understanding Functions (F11c, F11d, F11f) meets this expectation. Likewise, the fit of Cooperation Activity and Function (F33a) also meets this expectation.

The fit of Modification with Representation (F21b) and the fit of Control with Transformation (F32b) correspond to our expectation that certain tasks would be supported indirectly by tool functions. Recalling our earlier discussions in Chapter 2 and Chapter 4, the tasks of maintenance are dependent upon one another. Our findings indicate that Control activity support from the Transformation Function leads to higher tool use. Our findings also indicate that the fit between the Modification activity and the Representation function is associated with higher tool use. Once again this result is understandable based on the close association between the Understanding and Modification Activities.

A negative effect of fit on software maintenance tool use was found for the types of fit listed in Table 2.

Name	Description
F11b	Fit of Bug Related with Representation
F12a	Fit of KB with Transformation
F21a	Fit of Modification with Analysis
F31a	Fit of Control with Analysis
F31c	Fit of Cooperation with Analysis

Table 2

This set of results is, in at least one aspect, more interesting than the former. The general notion of Task-Technology Fit, and Cognitive Fit theory in particular, predicts higher tool use when fit is present. While we expected to find the results summarized in Table 1, our finding that fit is associated with lower tool use in certain cases is unexpected. Of particular interest is the fit of Bug-Related activity with the Representation Function. This activity is one of the core maintenance activities in the Vessey de-bugging model (1986). Essentially this activity describes a series of activities, including hypothesis generation and evaluation, which are diagnostic in character, and are arguably at the heart of the maintenance process.

This result has a number of potential explanations. While it is possible, of course, that this result is spurious, or is an artifact of the research method, an interesting possible explanation may be found in the original research framework. In the research framework described in Chapter 2, fit is related to utilization primarily through a causal chain which includes perceived usefulness. As we have stated earlier, fit and perceived usefulness are closely linked.

In the Davis Technology Acceptance Model (1989) Perceived Ease of Use was found to be strongly associated with tool use. In the case outlined above, it is entirely possible that

ease of use was low for the software maintenance tool functions needed to perform diagnostic (bug related) activities. Stated plainly, even though fit, and presumably perceived usefulness, exist, it may be too difficult or time consuming to use tools for this set of activities. Task-Technology Fit may be necessary, but insufficient to bring about tool use. We do not have Perceived Ease of Use data for any of the tool functions. This data would enable us to test the relationship between fit and Perceived Ease of Use. It is also possible that the programmer's experience with the tools had previously been unfavorable for this set task and technology combinations. The programmers may find the process more error prone, and therefore rely on traditional manual methods. These speculations remain unproved and are subject to further investigation.

We should also note that the other fit variables, including those we would predict from Cognitive Fit theory, involving the bug-related activities did not enter the regression. This "non-result" does not tell us that a relationship does not exist, rather it tells us that we cannot say a relationship does exist. We do not know if these other fit variables, which might have shed light on the situation had they entered the regression, are redundant with the fit variables which did enter or if they are irrelevant. In any event, this result is interesting and leads us to propose additional research on the interaction of fit and ease of use. It is our initial expectation, in the cases where fit leads to lower use, that the perceived ease of use, for the technology in question, will be found to be low.

It is especially interesting that three of the fit variables which had negative associations (see Table 2) involve the analysis function. This particular function may have proved difficult to use or it may be ineffective. Ease of Use data for each major tool function could be collected in a future well focused laboratory experiment which we discuss below. It is important for an software maintenance tool to provide an effective analysis function in

order to support this activity. This appears to be a significant deficiency in the tool environment for many maintenance programmers. In earlier software maintenance tools, the representation function appears to be dominant. Unfortunately, most of the tools available today for automated process analysis appear to be weak relative to the representation function. Programmers may eschew the current analysis functionality in favor of a manual approach which is partially supported by the representation function.

In the development of the model in Chapter 2 and in the tests of the hypothesis in Chapter 4, we have allowed for the existence of the so called “off-diagonal” fit relationships. The interpretation of the off-diagonal fit relationships is, as noted earlier, straightforward. The core elements of the maintenance process are understanding and transformation. These processes are the fundamental activities needed to accomplish a maintenance task. There is a close relationship between these processes as they are mutually dependent (Vessey, 1986). The third major type of activity, coordination, is not unique to maintenance, but is common to any task or activity which takes place in an organizational context. The maintenance programmer may have sole responsibility to accomplish a task, but the task must be completed within the context and strictures imposed by the organization.

We see that there are several instances where an off-diagonal, or minor fit, is significant. As illustrated in Table 1, there are two instances of minor fit which have a positive impact on tool use. These are the fit of Modification with Representation, and fit of Control with Transformation. This result is expected and is consistent with the explanation above.

The minor fit relationships which involve a negative association with maintenance software tool use (see Table 2), are somewhat more interesting. We discussed above the possibility that the minor fit variable involving Analysis may be the result of an artifact or the relative immaturity of the Analysis function. However, the remaining fit variable of

Knowledge Building with Transformation is more interesting. The argument here, however, parallels that used to explain the major fit variable which has a negative association with tool use. The perceived ease of use for the transformation functions may be sufficiently low to preclude tool use, even in the presence of high fit. This speculation, as before, remains an empirical question which can only be addressed through further research.

### **2.1.1 Experience / Expertise**

This study examined the association of experience with software maintenance tools and experience with software maintenance task on software maintenance tool use. The original task technology fit model of Goodhue (1988b) employs individual characteristics (of the programmer) as a moderating variable. Experience has been found to be a significant factor in predicting software use (Fuerst & Cheney, 1982). It is, however, not entirely clear that experience is the most appropriate variable. Certainly the notion of expertise is a richer and more encompassing construct. Experts employ different strategies in debugging programs (Vessey, 1986). This finding is germane with regard to the design and operation of software maintenance tools.

While the use of experience as a partial substitute for expertise is a limiting factor, it does present the advantage of simple measurement and it enables us to work with continuous or interval scales rather than the dichotomous expert / novice variable as in Vessey (1986).

In this study, three types of experience were associated with higher levels of tool use. The positive association of prior experience with the software maintenance tools with current utilization was expected and is hardly surprising. Somewhat more interesting is the association of prior experience with software being maintained with higher tool use.

It was our *a priori* expectation that higher levels of experience with the software being maintained would lead to lower rather than higher amounts of tool use. It appears that the effect of encountering an unknown or lesser known system is entirely the opposite of that expected. Our assumption that the programmer would employ a tool in an attempt to initially understand a system is erroneous. It appears that the programmer, when faced with the unknown, will revert to the “traditional” methods of code reading and taking notes. Given the capabilities, as measured by low means, of the analysis, representation, and transformation functions available to the programmers (see Chapter 3), it is possible that the tools available to the programmers are, on average, not sufficiently mature or sophisticated for the programmer to comfortably and efficiently “engage” new or unknown software systems for maintenance. The results further suggest that the programmer, once sufficiently familiar with the target software, understands it enough to be able to apply software maintenance tools effectively. In other words, it is necessary for the programmer to have experience with both the software maintenance tools and the target software in order for tool use to be effective.

While this conclusion is rational based on the outcomes described in Chapter 4, it is not entirely clear that this conclusion generalizes to all maintenance software nor to all organizations. The sample of projects and programming environments in this study is heterogeneous. A number of groups had access to software maintenance tools which are limited in functional capabilities. Two groups of programmers had access to more modern, and more capable tools. We believe that our conclusions regarding experience with tools may not apply to both types of environments. We were unable to control for the effect of different tool sets. The question of the effect of prior experience on use

remains open, and could be more conclusively addressed in a laboratory experiment where task and tool access are controlled.

### 2.1.2 Task Complexity

It was our *a priori* expectation that higher task complexity would be associated with higher levels of software maintenance tool utilization. We found that task complexity was indeed a significant predictor of tool utilization. We employed five factors in our operationalization of task complexity (Campbell, 1988). These factors included task interdependence, task variability, task ambiguity, task variability, and task scale. The findings indicated a mixed result. Task ambiguity was not significant, while the remaining four factors were significant. Higher task interdependence and task scale were associated with higher amounts of software maintenance tool use. The remaining task complexity variables, task uncertainty, and task variability are associated with lower levels of tool use.

This result is consistent with our finding that lower experience with the software being maintained is associated with lower amounts of software maintenance tool use. Task uncertainty and variability appear to lead the programmer to “revert” to traditional approaches using code reading and note taking. The programmer may be rely on traditional methods until the task is sufficiently well understood. Alternatively, the software maintenance tools may not have functionality which benefits the programmer under these conditions, or the ease of use of these functions may be low. Consequently, it is possible that these results do not generalize to all software maintenance tool environments.



### 2.1.3 Task Type

Software maintenance can be divided into two categories, debugging and enhancement (Pennington & Grabowski, 1990). Other authors have divided maintenance into as many as four categories (Bendifallah & Scacchi, 1987; Lientz, Swanson & Tompkins, 1978; Swanson, 1976). Our use of the two categories of debugging and enhancement is based on the suggestion that these two types of maintenance may involve different underlying processes. This is a conjecture from the literature, and it remains an empirical question. The more usual division of maintenance into three or four categories of adaptive, corrective, perfective, or preventative is based on the purpose of the maintenance project, not on the underlying process. This broader classification system is relevant with respect to the management of maintenance, but not with respect to the conduct of maintenance by a programmer nor to the support of the process with software engineering technology.

This study examined whether the type of maintenance project, debugging or enhancement, determines the level of software maintenance tool use. We found that while task type has no direct effect on tool use, it has an indirect effect through the interaction of task type with two fit variables. The interaction of task type and the fit of Bug Related activity with the Transformation function has a negative effect on the level of tool use. The interaction of task type with the fit of Control activity with the Transformation function has a positive effect on tool use. As these fit variables are known already to have a significant effect on software maintenance tool use, we conclude that the role of task type is to attenuate the effect of the fit variables.

The result suggests that task type may have a rather subtle effect on tool use. In the absence of a significant main effect and in light of the fact that only two of twelve interaction terms are significant, we are unable to conclude that from the practical

perspective, different tasks produce different levels of tool use. The fit variables in question are difficult to interpret, especially in light of the contradictory signs of the coefficients. It appears that this result is best understood in the context of our earlier discussion regarding the effect of perceived ease of use.

#### **2.1.4 Software Utilization Research**

The study of software utilization is often linked to the study of system success (DeLone & McLean, 1992); that is, utilization is used as a proxy for success. This assumption is made frequently in the software development literature. In the study of software development, utilization is ultimately a necessary, but insufficient condition for success (Nance, 1992; Trice & Treacy, 1988). In other words, a successful outcome or result presupposes utilization. Therefore, the study of utilization is essential in order to develop an understanding of the factors which lead to a successful outcome. A number of other factors must frequently act in order to produce a successful outcome. Trice and Treacy (1988) point out that a task which uses an information system must be designed efficiently, otherwise, software use may actually lead to a sub-optimal outcome. When utilization is not voluntary, e.g. software use is the result of a mandate, the study of use is, in and of itself, rather uninteresting. When use is voluntary, the fact that use occurs is significant (DeLone & McLean, 1992; Trice & Treacy, 1988).

The issue of voluntariness of use is germane in all studies of utilization. However, voluntariness is most often an issue in the study of the more classical data processing oriented systems. In processes which are highly rationalized (standardized) the fact of use of a prescribed software tool is a forgone conclusion. The measurement of software utilization in such a circumstance is largely irrelevant.

This study concerns itself with the use of software maintenance tools. The process of software maintenance is certainly not standardized, nor are the procedures for the use of software in the software maintenance process. The use of these software products is largely at the discretion of the individual user. This use of software tools resembles the use of DSS in many organizational settings. Contrast this with the use of a typical MIS or data processing system in which use is involuntary and prescribed by the organization.

While some organizations have begun to develop (or purchase) so called “software maintenance methodologies”<sup>1</sup>, the maintenance process continues to vary widely both between and within software development and maintenance organizations. Indeed, the fact that the process varies widely is what makes it interesting. We can observe programmers working as they prefer to work, without excessive organizational or procedural strictures. It is certainly the goal of a good portion of the current research in software engineering to standardize, and render programming as a rational, repeatable process. However, once this occurs, the study of the use of certain tools and techniques in field settings will become much more difficult as the effects of use and voluntariness become confounded.

The notion that task-technology fit is associated with tool use has been proposed by other researchers, notably Trice and Treacy (1988), Goodhue (1988b), and Nance (1992). The results of this study clearly support the validity of this approach to the study of software

---

<sup>1</sup> While software development methodologies are well known in the literature and have been in place in many organizations for over a decade, only recently have software maintenance methodologies become available. To date these methodologies have failed to achieve widespread, much less universal, acceptance.

utilization. The results indicate that fit may be broken into several specific components and that a set of fit variables may be combined with other variables to explain tool use.

This study presents a unique approach to task-technology fit in that we proceeded from task and technology models which are unique or specific to the task-technology combination. In order to extend this approach to other research venues, it would be necessary to develop or obtain from the literature specific models of both task activities and tool functionalities which are reasonably specific to that circumstance. The alternative to this approach is to use a general model of task-technology fit such as Goodhue (1988b). This approach has the advantage of being largely “context free”. Unfortunately, a general model may not have the explanatory power of a more specific model. This assertion is supported by our tests of the general fit model. While producing a significant explanation of tool utilization in our task environment, the general model in our case lacked the explanatory power of the specific model.

## **2.2 Maintenance Activity Support**

In this study, we posited that the support for a maintenance activity, understanding, transformation, or coordination, would come primarily from the corresponding tool function. We also posited that support for an activity could come from other tools. Our analytical approach to these questions involved the simple correlation table between the various task activities and tool function. This analysis is a companion to the regression based analysis which was employed in our examination of the impact of task-technology fit.

Our results parallel, as expected, the results in our examination of task-technology fit. We found, somewhat contrary to our expectations, that in the case of the understanding

activity, support was stronger from the modification and coordination functions than from the understanding function itself. The transformation activity is likewise strongly associated with the understanding support functions. This finding indicates understanding activities and transformation are strongly related. The fact that understanding activity is not as strongly supported by understanding functionality is interesting. Upon inspection of the correlation matrix for the sub-activities and sub-functions, the situation becomes clearer. The planning and knowledge building sub-activities are not highly associated with analysis and representation, whereas the bug-related sub-activity is associated with these functions. Once again we believe that this indicates that the software maintenance tools which were available to the programmers in the study are immature or the tool functionalities in question have relatively low perceived ease of use. The software maintenance tools at all of the participating organizations included several tools which are mainframe based and have been available for a number of years. Anecdotal evidence suggests that the more modern workstation (GUI) based tools address some of these issues.

The most surprising finding in our examination of maintenance activity support is that the transformation activity was not associated with the transformation function. This result is consistent with the finding that the task-technology fit between the transformation activity and function were not associated with software maintenance tool utilization. Two possible explanations exist. First, it is possible that the transformation activity is not well supported by existing transformation technology. This is the likely cause of the result. The mean level for the transformation functionality in the sample is only 1.83 (see Table 5 in Chapter 3). On a seven point scale, this is quite low. Furthermore, the state of development of maintenance tool functionality is well behind that of development oriented software (conventional CASE tools). All of the programmers, with the exception of 2

groups at one site, had access to mainframe oriented software maintenance tools which we believe are weak in this functional area, while being especially strong in the area of representation. Second, it is possible, but much less likely, that the transformation function construct may be inadequately defined or operationalized for the maintenance software tool context.

### 2.3 Task Type: Debugging vs. Enhancement

One of the objectives of this study was to examine maintenance projects which differ on the basis of task type to determine if these projects differ in their underlying process. While a definitive answer to this question would require an experiment with control over task type and the ability to record the process, we examined this question using the data already collected in our study of the effect of task-technology fit.

In this study we obtained data regarding the activities which the programmer engaged in during the course of each project. This data, together with information regarding the type of maintenance project, enabled us to examine whether there was a difference in the type and frequency of activities between debugging and enhancement projects.

We have found that there was no difference between debugging and enhancement maintenance across the six maintenance sub-activities, using either multivariate or univariate tests. This finding is consistent with our *a priori* expectations.

The existing literature on the underlying processes of maintenance has long emphasized the debugging process. The process of enhancement maintenance has not been explicitly investigated. However, the common underlying basis for debugging and enhancement maintenance is program understanding (Pennington & Grabowski, 1990). The key difference between the two types of maintenance would seem to be that debugging

maintenance may involve more testing of the output of a program to determine how well the program is performing relative to expectations. This would tend to suggest that debugging may be either more a more difficult or more sophisticated process. Although this study was not conducted under laboratory circumstances, it provides empirical evidence for what was heretofore only a well founded conjecture from the literature.

The finding that the processes are not different essentially means that the design of software maintenance tools need not take into account different underlying processes. The results, taken in the context of the existing literature, would indicate that software designed to support the debugging process would also serve the enhancement process.

### **3. Limitations of the Study**

This study is subject to limitations with regard to the generalizability and interpretation of the results. These limitations arise from the design of the study and in the data collection methods employed. Diligence was employed in the design of the study and in the selection and development of measures. However, it was impossible to eliminate all sources of bias. The sources of the limitations to the study are discussed below.

#### **3.1 Field Study Design**

This study was designed and implemented as a field study. As such, this study is subject to those limitations inherent in all field studies, the lack of control over environmental variables such as the makeup of the maintenance tool sets in the participating organizations, programming practices, and the coordination and control procedures in each organization with regard to the initiation and management of maintenance projects.

To compensate for the inherent limitations of field studies, this study employed a model of the maintenance process which was developed in a laboratory experiment in which the investigator had positive control over the task to be performed, the experience levels of the participants, as well as some control over procedures to be followed in the task (Vessey, 1986). While the use of a field study methodology generally would imply a lower degree of internal validity, the use of models developed in a setting with high internal validity clearly enables us to maintain some degree of internal validity, especially with regard to maintenance tasks. Thus, this field study builds on previous work with high internal validity, and extends it to a more generalizable sample.

This study collected data from 36 programmers on 74 projects in three organizations. The participation of the three organizations was clearly not random. They were actively sought out through personal contact. The sample consisted of projects which we were able to obtain from these organizations. Once an organization agreed to participate, we solicited "volunteers" from managers and programmers within the MIS organization with the assistance of our site contact.

All participants in the study were volunteers. However, they were approached by their managers in an uncontrolled setting. Although assurances were given both by the investigator and the manager, we cannot be certain that participants believed that their involvement was voluntary in all cases. We did note that there were programmers in each organization who did decline to participate with no apparent reprisals from their managers.

A serious concern in the design of the data collection protocol was the management of project selection bias on either the part of the managers or programmers involved. Our approach to this problem was to instruct the participants to supply data on all projects to be started and completed within the negotiated time frame. There were a number of



programmers who declined to continue to participate after one or two projects were furnished. However, as we collected data from several groups across three companies, we do not believe that a systematic bias exists in the sample.

### **3.2 Single Measure of Tool Use**

A source of bias which imposes limitations on our ability to generalize and interpret our results arises from the use of a single data source in the measurement of the dependent variable. Recall from our discussion of the design of the study in Chapter 3 that programmers were asked as part of the Project Completion questionnaire to report on a seven item scale the level of tool usage for each of the tools used during the course of the project. The fact that the level of tool use is assessed by a single source introduces the possibility of method and response bias. Self reports of behavior are, in general, problematic. Sproull and Kiesler (1986) report that e-mail users may systematically underestimate their e-mail usage. Self-report bias has also been found in system developers self evaluations (Lee, Goldstien & Guinan, 1989). Their findings indicate that the source of self-report bias may have been the respondents' desire to inflate their own importance or to provide a socially desirable response.

In our study this problem is mitigated somewhat because individual programmers use several tools and provided a separate response for each tool. The study was designed to provide confidentiality to the respondents, thus minimizing the incentives to provide a socially desirable response. In addition, the programmers provided usage data in response to a seven point scale. This has the effect of minimizing use estimation problems when a respondent is asked to provide an estimate of a specific number of instances of use.

Finally, the relatively large sample of programmers from several organizations leads us to believe that a systematic response bias does not exist.

However, method bias arising from the single source remains an issue. The original design of the study included a separate assessment of tool use through the automated monitoring of tool use. The study was originally designed to control for method bias through the use of a second and objective data source: IDCAMS database records automatically kept at one of the original participant organizations. However, as the result of a merger action, this organization withdrew from the study before data could be collected and replacement organizations were unable to supply this data. Additional measures for the collection of usage data were considered and discarded. Usage reports by managers were judged to be unreliable. Additional reporting mechanisms from the programmers, such as activity logs, were too intrusive or impractical.

## **4. Contribution**

### **4.1 Implications for Research**

This study makes several important contributions to the literature of software maintenance and software tool utilization. We developed a research model based on previously developed models. It extended the model of software debugging (Vessey, 1986) to include enhancement maintenance. In addition, this model, which was developed from protocol analysis conducted in a controlled laboratory setting, has been operationalized as a series of questionnaire items and has been tested in a field study. Our findings strongly suggest that the software debugging model (Vessey, 1986) is a suitable general model for software maintenance.

Our results indicate that the fit between a task and the technology available to support that task is a strongly associated with tool use. While task-technology fit is not identical to perceived usefulness, (Nance, 1992), the two constructs are closely related. The development of task-technology fit as a construct which explains utilization is important to the process of understanding the links between the nature and antecedents of task outcome when software is involved in the performance of the task. The study clearly demonstrates the viability of the specific fit approach to the measurement of fit.

The third contribution is in the design of software maintenance tools. The literature of software maintenance tool development is largely concerned with automated understanding. The research in this area proceeds from models of maintenance which are not based in the psychology of programming literature or the human cognition literature. It is becoming apparent that completely automated design recovery (reverse engineering) from legacy software may be impossible (Bennett, 1993). This study gives some insight into the direction in which research on the development of maintenance support tools may fruitfully proceed. Software tool builders should begin to focus their efforts on building tools which assist the programmer in performing program comprehension, rather than automating the process.

## **4.2 Implications for Practice**

The contribution of this study to the area of MIS practice is in the area of software maintenance support. While this study did not concern itself with performance outcomes, our findings indicate that programmers will make use of software maintenance tools when they are available. In addition, our findings indicate that tools which support the program understanding process are particularly important and useful. Furthermore the findings suggest that the more sophisticated workstation or GUI oriented software maintenance

tools would support the software maintenance process in a more comprehensive way than the mainframe based software maintenance tools.

Finally, our findings indicate that management must realize that even though programmers have individual project responsibilities, their work is constrained and to a large degree defined by the organizational context. The support of the programmers work in the area of coordination is essential, especially in the area of control (compliance with standards and practices). The mismatch between levels of task demands and tool support for this area is especially wide. The development and adoption of the more modern maintenance tools is expected to address this problem. MIS managers should be aware of the availability of modern software maintenance tools and adopt this technology whenever possible. We believe that it is essential for MIS managers to provide modern and flexible tools to support all phases of the programming process, especially maintenance.

## **5. Future Research**

This study leaves a number of avenues open for future research. The basic task-technology fit framework of Goodhue (1988a) employs as an ultimate dependent variable, performance or outcome. In this study we employed utilization, an intervening variable in the earlier model (Goodhue, 1988a), as the dependent variable. The first and most obvious extension to this study is to examine the relationship between software maintenance tool utilization and performance in the maintenance context. The study design may be extended to include the collection of several performance measures from the programming manager and the user-customer.

The model or research framework developed in Chapter 2 is based, in part, on the Technology Acceptance Model of Davis (1985). The TAM model has been evaluated by

several authors without directly extending the model. Hartwick and Barki (1994) recently have extended the Theory of Reasoned Action (Ajzen & Fishbein, 1980), the parent of the TAM, in an examination of the relationship between user participation in the development process and the subsequent use of the system. This paper extended the TRA model in a manner similar to that used in the development of the model for this study. However, in their paper, Hartwick and Barki were able to test their model in its entirety using EQS, a structural equation modeling program, because their sample size was much larger than ours. However, the collection of additional data will subsequently allow us to examine our model in more detail using the structural equation modeling technique.

This study was conducted as a field study. As a consequence it is subject to limitations, as described above, which are typically associated with field studies. Especially notable are limited control over task, demographic characteristics, and available software maintenance tools. This raises the usual issue of internal validity. In addition to these limitations, it is difficult in a study design such as this, to obtain multiple measures of behaviors. In a laboratory experiment these concerns can be managed closely. It is our intention to examine the task-technology fit for the understanding and transformation dimensions of the task and technology matrix using a laboratory setting. The preliminary design of this extension calls for the programmer to be given a standard task to be completed using the Micro Focus COBOL Workbench. This particular software product contains an advanced software engineering tool known as an animator. This is an interactive tool which allows the user to extract and test program structures. This particular tool was used by some of the participants in this study. The use of the same tools will allow us to compare the results obtained in the field study more directly with the results we hope to obtain in the laboratory. The major objective in the laboratory experiment will be to confirm and

extend our understanding of the results obtained in the field for these two dimensions of task-technology fit.

Our results raise the question of the effect of Perceived Ease of Use, and its interaction with Task-Technology Fit, on software maintenance tool use. In order to determine this effect, it will be necessary to conduct a laboratory experiment where we collect data regarding Perceived Ease of Use at the tool function level, rather than at the tool set level, as was done in this study. This will enable use to examine this question more completely and lend explanatory power for some of the more “unexpected” results noted above.

One of the questions regarding software maintenance tool utilization which is unanswered by this study is the effect of learning (continued experience with software maintenance tools) on tool utilization. The design of this study was cross-sectional in nature. In order to address the effects of learning, a longitudinal design is necessary. We expect to address this issue through a continuing relationship with one of our research sites which has migrated from an mainframe based tools environment to a workstation (GUI) based environment since data was collected at that site.

## **6. Conclusion**

Software maintenance is a critical area in MIS. This area of software engineering practice occupies, by most reports, the vast majority of the software budgets of organizations. Yet historically this area receives relatively little attention from MIS managers except to complain of the difficulty and expense of the process. Indeed, this “neglect” extends to research into the maintenance process. A recent survey of the major MIS, Software

Engineering, and Computer Science journals found that of nearly 1000 articles, less than 3% concerned software maintenance in a significant way (Schneberger, 1993)<sup>2</sup>. It is fair to say that the area of software maintenance has been neglected by researchers and practitioners alike, considering the resources consumed by maintenance. The attitude of most MIS managers, and indeed many MIS academics has been one of “benign neglect”. Essentially, the solution to the so-called “maintenance problem” has been, for most, to look for more effective or efficient development processes. Just as once it was thought that nuclear power would eventually be “too cheap to meter”, the objective of many in the software development tool and management areas has been to make software “too cheap to maintain”.

In this study we have sought to further our understanding of how maintenance is done and how it can be supported now, in the “real” world. The eventuality of disposable software is far from certain. It is our view that the solution to the “maintenance problem” is to learn how necessary maintenance can be supported effectively and efficiently. We view the software portfolio of an organization as a valuable resource. Keeping this resource available and effective should be our objective as long as the economics of repair and renovations favor it over replacement.

---

<sup>2</sup> Nearly half of the articles involving software maintenance during the survey period appeared in a single issue of *IEEE Transaction on Software Engineering* in March 1987 (Schneberger, 1993).

# **Appendix A**

## **Constructs and Final Items**



## Maintenance Tool Function Items

(To what extent do the maintenance software tools available to you supply the following functions?)

### **Analysis Function** ( $\alpha=0.69$ )

Check for consistency between different system representations or "models".

Search the system or part of the system for inconsistencies or redundancies in data definitions or data or process models.

### **Control Compliance Function** ( $\alpha=0.92$ )

Track budget and cost information for the project.

Track schedule and/or budget information for the project.

Maintain project management status and information.

Maintain a record of who is responsible for each part of the project.

### **Project Cooperation Function** ( $\alpha=0.91$ )

Send system design information or messages to other individuals.

Exchange information relating to the project with other individuals.

Share project data or information with other individuals.

### **Representation Function** ( $\alpha=0.88$ )

Construct representations of entities, relationships, or processes in a diagram or model.

Represent the objects, relationships, or processes of the system or part of the system in terms of models (data flow diagrams, entity-relationship diagrams, flowcharts, etc.)

Model a system in terms of process, flow, or data models.

### **Transformation Function** ( $\alpha=0.80$ )

Convert a physical specification of a system or part of a system into a logical one. (e.g., create a flow chart from code)

Automatically produce documentation as a by-product of systems maintenance activities.

Produce a high level specification (e.g., diagram) from a lower level, or more detailed representation

## Maintenance Activity Model

### Bug Related Activity ( $\alpha=0.81$ )

I frequently compared actual results of a program run to the expected results for that run.

I revised my assumptions regarding the program after making test runs of the program.

I tested the program and made additional changes in the code based on the result of my tests.

### Control Compliance Activity ( $\alpha=0.71$ )

I made an effort to insure that the changes I made in this project would not interfere with other work being done at the same time by others.

I had to observe company standards during this project.

This organization has standards for variable names or source code formatting.

I followed a standard procedure in turning this project over for production

### Cooperation Activity ( $\alpha=0.78$ )

I had to keep another programmer or analyst informed of my work so as to keep my work consistent with another project.

This project was part of a program or set of projects which were related.

I communicated with other programmers or analysts so that my work would not negatively impact their work.

### Transformation Activity ( $\alpha=0.68$ )

I made several changes in the code.

I added new functions to this system.

I improved this system or program.

I modified the system's functions.

**Knowledge Building Activity** ( $\alpha=0.81$ )

- I obtained information about the system from comments in the body of the programs
- I made extensive use of my knowledge of the programming language(s) and data base system in which the software is written.
- I asked a colleague for technical information during this project.
- I consulted manuals to obtain information about the programming language(s) and / or data base system.
- I examined samples of the input data.
- I obtained information about the system from the data base schema.
- I learned a great deal about the system by mentally processing parts of the system code.
- I learned a great deal about the system by examining the JCL code.
- I had to weigh and evaluate a large volume of information about the system I was maintaining

**Planning Activity** ( $\alpha=0.68$ )

- I frequently re-evaluated my plan of action.
- I did not hesitate to change my approach to solving a problem if it appeared I was at a "dead-end".
- I frequently had alternative approaches to solving a problem.
- I had a number of choices to make regarding which source of information to consult in order to solve a particular problem.

## Task - Technology Fit Items (Direct Measurement)

### Accuracy of Information ( $\alpha=0.76$ )

The information that I obtained from the software maintenance tools about the software that was maintained was accurate enough for my purposes.

There were accuracy problems in the information I obtained from the software maintenance tools about the software system that was maintained.

### Access to Information ( $\alpha=0.77$ )

I could quickly get information about the system being maintained.

It was easy to get access to information that I needed on the system being maintained.

It was difficult to obtain the information on the system being maintained which I needed to complete this project.

### Assistance in Use of Tools ( $\alpha=0.85$ )

I got the help I needed in accessing and understanding information on the system being maintained.

It was easy to get assistance if I had trouble finding or using information on the system being maintained.

### Tool Compatibility ( $\alpha=0.79$ )

When it was necessary to compare information obtained from two or more different sources by the software maintenance tools sources about the system being maintained, there were unexpected or difficult inconsistencies.

There were times when supposedly equivalent information derived by the software maintenance tools from two different sources about the system being maintained was inconsistent.

Sometimes it was difficult or impossible to compare information derived by the software maintenance tools from two different sources about the system being maintained because the information from the different sources was defined differently.

### Confidence ( $\alpha=0.83$ )

There were so many different types of software maintenance tools available, it was hard to know how to use them effectively.

There were so many different software maintenance tools, each with different capabilities, which could be used to obtain information on the software being maintained, that it was hard to understand which one to use in a given situation.

**Currency of Information** ( $\alpha=0.86$ )

I couldn't get information about the system being maintained from the software maintenance tools that was current enough to meet my needs.

I needed some information on the up-to-the minute status or state of the system being maintained but could not get it from the software maintenance tools.

The information obtained through the use of the software maintenance tools on the software being maintained was up to date enough for my purposes.

**Ease of Tool Use** ( $\alpha=0.70$ )

It was easy to learn how to use the software maintenance tools that give me access to information on the system being maintained.

The software maintenance tools that gave me access to information on the system being maintained were convenient and easy to use.

**Level of Detail** ( $\alpha=0.83$ )

Sufficiently detailed information about the system I maintained was available from the software maintenance tools.

Information about the system that was being maintained, was available from the software maintenance tools at an appropriate level of detail for my purposes.

**Locatability** ( $\alpha=0.78$ )

It was easy to locate information about the system to be maintained, even if I had not used that information before.

It was easy to find what information was available on the system that was maintained.

**Clarity of Meaning** ( $\alpha=0.76$ )

On the reports or output produced by the software maintenance tools I dealt with, the exact meaning of information I will obtain was either be obvious, or was easy to find out.

The exact definition of the information items I obtained from the software maintenance tools on the software being maintained was easy to find out.

**Overall Fit of Tool to Task** ( $\alpha=0.65$ )

All in all, the software maintenance tools were satisfactory in meeting my needs.

Overall I believe there were some important problems with the way the information about the systems or software being maintained is managed and made available that made it harder to do my job.

The information about the software to be maintained and the way it was provided adequately met my needs.

**Information Presentation** ( $\alpha=0.82$ )

The information obtained from the software maintenance tools on the system being maintained was displayed in a readable and understandable form.

The information obtained from the software maintenance tools on the system being maintained was presented in a readable and useful format.

**Tool Reliability** ( $\alpha=0.40$ )

The software maintenance tools were subject to frequent systems problems and crashes.

I could count on the software maintenance tools to be "up" and available when I needed them.

## Task Complexity

### Task Ambiguity ( $\alpha=0.81$ )

To what extent did multiple opinions exist of how the final system should look?

During the system maintenance process, to what extent did you find that information used in making decisions meant different things to different people?

To what extent did multiple views exist of how the final system should be maintained?

During the system maintenance process, to what extent could information be interpreted in several ways, which could have led to different but acceptable solutions?

### Task Interdependence ( $\alpha=0.72$ )

To what extent do the people on your group have one-person jobs; that is, in order to get the work out, to what extent do group members independently accomplish their own assigned tasks?

To what extent did group members meet together to discuss how each activity should be performed in order to do the work of the group?

To what extent is (was) yours a one-person job; is (was) there little need for checking or working with others.

### Scale of Task ( $\alpha=0.72$ )

To what extent were the technical problems for this system particularly complicated?

To what extent would you characterize this project as being large?

How technically complex is the system that was maintained?

### Task Uncertainty ( $\alpha=0.48$ )

While developing this system, how often did you come across specific but difficult problems that you didn't know how to solve, and you had to take some time to think through by yourself or with others before you could take any action?

In general, how much actual "thinking" time do you usually spend trying to solve such specific problems?

### Task Variability ( $\alpha=0.41$ )

In some aspects of the system maintenance process, activities were fairly predictable. In others, you were often not sure what the outcome will be. To what extent would you say that you were generally sure what the outcome of your efforts would be?

To what extent did requirements and design change requests occur in your work?

## Experience Items

### General Experience ( $\alpha=0.74$ )

To what extent does this programmer have experience in managing software maintenance projects?

To what extent does this programmer have experience in software maintenance?

To what extent does this programmer have experience in software development?

To what extent does this programmer have formal training in maintenance or development?

### Tool Experience ( $\alpha=0.71$ )

How many total hours have you used this tool

How frequently do you use this tool?

How much experience do you have with this tool?

### System Experience ( $\alpha=0.79$ )

How many time has the PA has maintained this system before?

How much experience does the PA have in maintaining the system?

Relative to other PA's in the group. how much experience does this PA have in maintaining the system?



## **TRA Items**

### **Attitude Towards Tool Use ( $\alpha=0.96$ )**

I think it would be very good to use the software maintenance tools rather than manual methods for this task.

In my opinion it would be very desirable to use the software maintenance tools rather than manual methods for the project.

It would be much better for me to use the maintenance tools rather than manual methods.

### **Control Over Tool Use ( $\alpha=0.89$ )**

I have much more control over the use of the software maintenance tools as compared to manual methods.

Given the resources, opportunities, and knowledge it takes to use the software maintenance tools versus manual methods, it would be easier for me to choose to use the software maintenance tools.

I would be much more able to use the software maintenance tools rather than manual methods because of differences in the resources, opportunities and knowledge it takes to use each one.

### **Intention to Use Tool ( $\alpha=0.92$ )**

I will use the software maintenance tools rather than manual methods to complete this project.

My intention is to use the software maintenance tools rather than manual methods in completing this project.

In completing this project, I would rather use the software maintenance tools than use manual methods alone.

### **Subjective Norms ( $\alpha=0.98$ )**

Those people who are important to me would strongly support my using the software maintenance tools rather than using totally manual methods.

I think that those people who are important to me would want me to use software maintenance tools rather than use totally manual methods.

People whose opinions I value would prefer me to use the software maintenance tools rather than manual methods.

## **TAM Items**

### **Perceived Ease of Use ( $\alpha=0.97$ )**

I will find it easy to get the software maintenance tools to do what I want them to do.

My interaction with the software maintenance tools will be clear and understandable.

I will find the software maintenance tools to be flexible to interact with.

I will find the software maintenance tools easy to use.

### **Perceived Usefulness ( $\alpha=0.98$ )**

Using the software maintenance tools will enable me to accomplish my tasks more quickly.

Using the software maintenance tools will enable me to improve my performance on this project

Using the software maintenance tools will enable me to increase my productivity on this project

Using the software maintenance tools will enable me to enhance my effectiveness on this project

Using the software maintenance tools will make it easier to do this project.

I will find the software maintenance tools useful in this project.

## **Appendix B**

### **Fit Variable Cross-Reference**

	Analysis	Representation	Transformation	Control	Cooperation
Bug-Related	F11A	F11B	F12B	F13A	F13B
Planning	F11E	F11F	F12C	F13E	F13F
Knowledge Building	F11C	F11D	F12A	F13C	F13D
System Modification	F21A	F21B	F22	F23A	F23B
Control Compliance	F31A	F31B	F32B	F33B	<i>not computed</i>
Cooperative Activity	F31C	F31D	F32A	<i>not computed</i>	F33A

Variable Name Matrix

Name	Description
F11a	Fit of Bug Related with Analysis
F11b	Fit of Bug Related with Representation
F11c	Fit of KB with Analysis
F11d	Fit of KB with Representation
F11e	Fit of Planning with Analysis
F11f	Fit of Planning with Representation
F12a	Fit of KB with Transformation
F12b	Fit of Bug Related with Transformation
F12c	Fit of Planning with Transformation
F13a	Fit of Bug Related with Control
F13b	Fit of Bug Related with Cooperation
F13c	Fit of KB with Control
F13d	Fit of KB with Cooperation
F13e	Fit of Planning with Control
F13f	Fit of Planning with Cooperation
F21a	Fit of Modification with Analysis
F21b	Fit of Modification with Representation
F22	Fit of Modification Activity and Function
F23a	Fit of Modification with Control
F23b	Fit of Modification with Cooperation
F31a	Fit of Control with Analysis
F31b	Fit of Control with Representation
F31c	Fit of Cooperation with Analysis
F31d	Fit of Cooperation with Representation
F32a	Fit of Cooperation with Transformation
F32b	Fit of Control with Transformation
F33a	Fit of Cooperation Activity and Function
F33b	Fit of Control Activity and Function

Variable Name Table

## References

- Adams, D. A., Nelson, R. R., & Todd, P. A. (1992). "Perceived Usefulness, Ease of Use, and Usage of Information Technology: A Replication". *MIS Quarterly*, 16(2), 227-247.
- Ajzen, I. (1985). "From Intentions to Actions: A Theory of Planned Behavior". In J. Kuhl, & J. Beckmann (Ed.), *Action Control: From Cognition to Behavior* (pp. 11-39). New York: Springer Verlag.
- Ajzen, I., & Fishbein, M. (1980). *Understanding Attitudes and Predicting Social Behavior*. Englewood Cliffs, NJ: Prentice Hall.
- Alexander, J. W., & Randolph, W. A. (1985). "The Fit Between Technology and Structure as a Predictor of Performance in Nursing Subunits". *Academy of Management Journal*, 28(4), 844-859.
- Araki, K., Furukawa, Z., & Cheng, J. (1991). "A General Framework for Debugging". *IEEE Software*, 1991(May), 14-20.
- Bagozzi, R. P. (1982). "A Field Investigation of Causal Relations Among Cognitions, Affect, Intentions, and Behavior". *Journal of Marketing Research*, 19(11), 562-584.
- Bailey, J. E., & Pearson, S. E. (1983). "Development of a Tool for Measuring and Analyzing Computer User Satisfaction". *Management Science*, 29(5), 530-545.
- Bendifallah, S., & Scacchi, W. (1987). "Understanding Software Maintenance Work". *IEEE Transactions on Software Engineering*, SE-13(3), 311-323.
- Bennett, K. H. (1993). *Understanding the Process of Software Maintenance. Proceedings of Second Workshop on Program Comprehension Capri, Italy*: IEEE Computer Society Press.
- Bourgeois, L. J., III. (1985). "Strategic Goals, Perceived Uncertainty, and Economic Performance in Volatile Environments". *Academy of Management Journal*, 28(3), 548-573.
- Brooks, R. E. (1977). "Towards a Theory of the Cognitive Processes in Computer Programming". *International Journal of Man-Machine Studies*, 9, 737-751.
- Brooks, R. E. (1983). "Towards a Theory of the Comprehension of Computer Programs". *International Journal of Man-Machine Studies*, 18, 543-554.

- Campbell, D. T. (1988). "Task Complexity: A Review and Analysis". *Academy of Management Review*, 13(1), 40-52.
- Cleveland, L. (1989). "A Program Understanding Support Environment". *IBM Systems Journal*, 28(2), 324-344.
- Corbi, T. A. (1989). "Program Understanding: Challenge for the 1990's". *IBM Systems Journal*, 28(2), 294-305.
- Cronbach, L. J. (1951). "Coefficient Alpha and the Internal Structure of Tests". *Psychometrika*, 16, 297-334.
- Davis, F. D., Jr. (1985). *A Technology Acceptance Model for Empirically Testing New End-User Systems: Theory and Results*. Unpublished Ph.D. Thesis, Massachusetts Institute of Technology.
- Davis, F. D. (1989). "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology". *MIS Quarterly*, 13(3), 318-339.
- Davis, F. D., Bagozzi, R. P., & Warshaw, P. R. (1989). "User Acceptance of Computer Technology: A Comparison of Two Theoretical Models". *Management Science*, 35(8), 982-1003.
- Davis, R. (1984). "Diagnostic Reasoning Based on Structure and Behavior". *Artificial Intelligence*, 24, 347-410.
- DeLone, W. H., & McLean, E. R. (1992). "Information Systems Success: The Quest for the Dependent Variable". *Information Systems Research*, 3(1), 60-95.
- DeVellis, R. F. (1991). *Scale Development: Theory and Applications*. Newbury Park, California: Sage Publications.
- Elstein, A. S., Shulman, L. S., & Sprafka, S. A. (1978). *Medical Problem Solving*. Cambridge, MA: Harvard University Press.
- Fedorowicz, J., Oz, E., & Berger, P. D. (1992). "A Learning Curve Analysis of Expert System Use". *Decision Sciences*, 23, 797-818.
- Fishbein, M., & Ajzen, I. (1980). "Predicting and Understanding Consumer Behavior: Attitude-Behavior Correspondence". In I. Ajzen, & M. Fishbein (Ed.), *Understanding Attitudes and Predicting Social Behavior* (pp. 148-172). Englewood Cliffs, NJ: Prentice Hall.

- Fishbein, M., Ajzen, I., & Hinkle, R. (1980). "Predicting and Understanding Voting in American Elections: Effects of External Variables". In I. Ajzen, & M. Fishbein (Ed.), *Understanding Attitudes and Predicting Social Behavior* (pp. 173-195). Englewood Cliffs, NJ: Prentice Hall.
- Fuerst, W., & Cheney, P. (1982). "Factors Affecting the Perceived Utilization of Computer-Based Decision Support Systems in the Oil Industry". *Decision Sciences*, 13(October), 554-559.
- Gode, D. K., Barua, A., & Mukhopadhyay, T. (1990). On the Economics of the Software Replacement Problem. *Proceedings of Eleventh International Conference on Information Systems* (pp. 159-170). Copenhagen, Denmark: ACM.
- Goodhue, D. L. (1988a). "I/S Attitudes: Toward Theoretical and Definitional Clarity". *Data Base*, 19(3/4), 6-15.
- Goodhue, D. L. (1988b). *Supporting Users of Corporate Data: The Effect of I/S Policy Choices*. Unpublished Ph.D. Thesis, Massachusetts Institute of Technology.
- Goodhue, D. L. (1992a). *A New Instrument for User Evaluations of Information Systems: Toward Organizational Assessments of MIS Effectiveness*. Working Paper . University of Minnesota.
- Goodhue, D. L. (1992b). User Evaluations of MIS Success: What Are We Really Measuring? *Proceedings of Hawaii International Conference on System Sciences* (pp. 303-314). Kona, HI: IEEE.
- Goodhue, D. L. (1994). "Understanding User Evaluations of Information Systems". *Management Science*, Forthcoming.
- Gopal, R., & Schach, S. R. (1989). Using Automatic Program Decomposition Techniques in Software Maintenance Tools. *Proceedings of IEEE Conference on Software Engineering* (pp. 132-141).
- Gould, J. D., & Drongowski, P. (1974). "An Exploratory Study of Computer Programming Debugging". *Human Factors*, 16(3), 258-277.
- Gremillion, L. L. (1984). "Determinants of Program Repair Maintenance Requirements". *Communications of the ACM*, 27(8), 166-172.
- Guinan, P. J. (1992). *Personal Communication - Research in progress* .
- Hanna, M. A. (1990). "Defining the "R" Words for Automated Maintenance". *Software Magazine*, May, 1990, pp. 41-48.

- Hartwick, J., & Barki, H. (1994). "Explaining the Role of User Participation in Information Systems Use". *Management Science*, 40(4), 440-465.
- Hausler, P. A., Pleszkoch, M. B., Linger, R. C., & Hevner, A. R. (1990). "Using Function Abstraction to Understand Program Behavior". *IEEE Software*, 1990(January), 55-63.
- Henderson, J. C., & Cooperider, J. G. (1990). "Dimensions of I/S Planning and Design Aids: A Functional Model of CASE Technology". *Information Systems Research*, 1(3), 227-254.
- Ives, B., Olson, M. H., & Baroudi, J. J. (1983). "The Measurement of User Information Satisfaction". *Communications of the ACM*, 26(10), 785-793.
- Joyce, W., John W. Slocum, J., & Glinow, M. A. V. (1982). "Person-Situation Interaction: Competing Models of Fit". *Journal of Occupational Behaviour*, 3, 265-280.
- Kim, C., & Westin, S. (1988). "Software Maintainability: Perceptions of EDP Professionals". *MIS Quarterly*, 12(2), 167-185.
- Kleinbaum, D. G., Kupper, L. L., & Muller, K. E. (1988). *Applied Regression Analysis and Other Multivariable Methods*(2nd. ed.). Boston: PWS-Kent.
- Lee, S., Goldstein, D. K., & Guinan, P. J. (1989). *Informant Bias in I/S Design Team Research*. Working Paper 89-25. Boston University.
- Letovsky, S. (1987). "Cognitive Processes in Program Comprehension". *Journal of Systems and Software*, 7, 325-339.
- Letovsky, S., & Soloway, E. (1986). "Delocalized Plans and Program Comprehension". *IEEE Software*, 1986(5), 41-49.
- Lientz, B. P., Swanson, E. B., & Tompkins, G. E. (1978). "Characteristics of Application Software Maintenance". *Communications of the ACM*, 21(6), 466-471.
- Littman, D. C., Pinto, J., Letovsky, S., & Soloway, E. (1987). "Mental Models and Software Maintenance". *Journal of Systems and Software*, ,
- Mathieson, K. (1991). "Predicting User Intentions: Comparing the Technology Acceptance Model with the Theory of Planned Behavior". *Information Systems Research*, 2(3), 173-191.
- Melone, N. P. (1990). "A Theoretical Assessment of the User Satisfaction Construct in Information Systems Research". *Management Science*, 36(1), 76-91.



- Miller, D. (1992). "Environmental Fit versus Internal Fit". *Organization Science*, 3(2), 159-178.
- Moad, J. (1990). "Maintaining the Competitive Edge". *Datamation*, pp. 61-66.
- Nance, W. D. (1992). *Task / Technology Fit and Knowledge Worker Use of Information Technology: A Study of Auditors*. Unpublished Ph.D. Thesis, University of Minnesota.
- Pennington, N., & Grabowski, B. (1990). "The Tasks of Programming". In J.-M. Hoc, T. R. G. Green, R. Samurcay, & D. J. Gilmore (Ed.), *Psychology of Programming* (pp. 45-62). London: Academic Press.
- Robertson, D. C. (1990). *CAD Systems and Communication in Design Engineering: A Test of the Information Processing Model*. Unpublished Ph.D. Thesis, Massachusetts Institute of Technology.
- Robson, D. J., Bennett, K. H., Cornelius, B. J., & Munro, M. (1991). "Approaches to Program Comprehension". *Journal of Systems and Software*, 14, 79-84.
- Schaffner, K. F. (Eds.). (1985). *Logic of Discovery and Diagnosis in Medicine*. Berkeley: University of California Press.
- Schneberger, S. L. (1993). Software Maintenance: An Update on Issues and Research. *Proceedings of 31st. Annual Southeast Conference* Birmingham, Alabama: Association for Computing Machinery Press.
- Schneidewind, N. F. (1987). "The State of Software Maintenance". *IEEE Transactions on Software Engineering*, SE-13(3), 303-310.
- Sejwacz, D., Ajzen, I., & Fishbein, M. (1980). "Predicting and Understanding Weight Loss: Intentions, Behaviors, and Outcomes". In I. Ajzen, & M. Fishbein (Ed.), *Understanding Attitudes and Predicting Social Behavior* (pp. 101-112). Englewood Cliffs, NJ: Prentice Hall.
- Sperber, B. M., Fishbein, M., & Ajzen, I. (1980). "Predicting and Understanding Women's Occupational Orientations: Factors Underlying Choice Intentions". In I. Ajzen, & M. Fishbein (Ed.), *Understanding Attitudes and Predicting Social Behavior* (pp. 113-129). Englewood Cliffs, NJ: Prentice Hall.
- Sproull, L., & Kiesler, S. (1986). "Reducing Social Context Clues: Electronic Mail in Organizational Communication". *Management Science*, 32(11), 1492-1512.
- Swanson, E. B. (1976). The Dimensions of Maintenance. *Proceedings of Second International Conference on Software Engineering* (pp. 492-497). San Francisco.

- Swanson, E. B., & Beath, C. M. (1989). *Maintaining Information Systems in Organizations*. New York: Wiley.
- Torasso, P., & Console, L. (1989). *Diagnostic Problem Solving*. New York: Van Nostrand Reinhold.
- Trice, A. W., & Treacy, M. E. (1988). "Utilization as a Dependent Variable in MIS Research". *Data Base*, 1988(Fall/Winter ), 33-41.
- Venkatraman, N. (1989). "The Concept of Fit in Strategy Research: Toward Verbal and Statistical Correspondence". *Academy of Management Review*, 14(3), 423-444.
- Vessey, I. (1985a). "Expertise in Debugging Computer Programs: A Process Analysis". *International Journal of Man-Machine Studies*, 23, 459-494.
- Vessey, I. (1985b). Expertise in Debugging Computer Programs: Situation-Based versus Model-Based Problem Solving. *Proceedings of International Conference on Information Systems*
- Vessey, I. (1986). "Expertise in Debugging Computer Programs: An Analysis of the Content of Verbal Protocols". *IEEE Transactions on Systems, Man, and Cybernetics*, MSC-16(5), 621-637.
- Vessey, I. (1991). "Cognitive Fit: A Theory-Based Analysis of the Graphs Versus Tables Literature". *Decision Sciences*, 22, 219-240.
- Vessey, I., & Galletta, D. (1991). "Cognitive Fit: An Empirical Study of Information Acquisition". *Information Systems Research*, 2(1), 63-84.
- Vessey, I., & Weber, R. (1983). "Some Factors Affecting Program Repair Maintenance: An Empirical Study". *Communications of the ACM*, 26(2), 128-134.
- Waters, R. C., & Tan, Y. M. (1991). "Toward a Design Apprentice: Supporting Reuse and Evolution in Software Design". *ACM SIGSOFT Software Engineering Notes*, 16(2), 33-44.
- Wilde, N., & Huitt, R. (1991). "A Reusable Toolset for Software Dependency Analysis". *Journal of Systems and Software*, 14(2), 97-102.
- Wilde, N., & Thebaut, S. M. (1989). "The Maintenance Assistant: Work in Progress". *Journal of Systems and Software*, 9, 3-17.
- Yau, S. S., & Collofello, J. (1985). "Design Stability Measures for Software Maintenance". *IEEE Transactions on Software Engineering*, SE-11(9), 849-856.

## Vita

### Mark Thomas Dishaw

#### EDUCATION:

- D.B.A. Boston University, September 1994  
Graduate School of Management  
Major: Management Information Systems
- M.B.A. University of Rochester, Rochester, New York, May 1981  
Graduate School of Management  
Major: Computers and Information Systems
- B.S. State University of New York at Albany, June 1975  
Major: Biological Sciences

#### EMPLOYMENT:

- 8/93 - Present **Saint John Fisher College**, Rochester, New York  
ASSISTANT PROFESSOR (Visiting) Department of Management
- 9/88 - 8/93 **Boston University, School of Management**, Boston, MA  
INSTRUCTOR, Management Information Systems Department
- 9/84 - 8/88 **Saint John Fisher College**, Rochester, New York  
INSTRUCTOR, Department of Management

#### Publications and Working Papers:

*Modeling Tool Utilization in the Software Maintenance Process*,  
with Diane M. Strong, Working Paper , February 1994

*Disciplinary Information Systems: The New Panopticons*, Working  
Paper 1990

*Object-Oriented Databases: A Not Ready for Prime Time Player?*  
Working Paper 1990

*Freeze-Fracture Observations on the Visceral Yolk Sac Placenta of  
Rats, Mice and Hamsters*, with S. J. Carpenter, **Anatomy and  
Embryology**, vol. 157, pp. 255-268 (1979).