December 1998

# Extending a Model of Task Technology Fit to CASE Tool Utilization in Design Maintenance Tasks

Mark Dishaw
*University of Wisconsin Oshkosh*

Diane Strong
*Worcester Polytechnic Institute*

# Extending a Model of Task Technology Fit to CASE Tool Utilization in Design Maintenance Tasks

**Mark T. Dishaw**
College of Business Administration
University of Wisconsin, Oshkosh

**Diane M. Strong**
Department of Management
Worcester Polytechnic Institute

## Abstract

*We present an experiment that tests for differences in software maintenance at the design level and at the program level. The experiment examines what maintenance task activities are performed, what maintenance tool functionality is used, and the fit between maintenance tasks and tool functionality. The experiment will be run this spring and results will be available for the conference.*

## General Statement of the Research Problem

One of the goals of the research program of which this project proposal forms a significant part is to develop an understanding of the utilization of CASE tools in the software maintenance process. This study is concerned with extending our model of Task-Technology Fit, which we applied previously to conventional software maintenance, in a new type of software maintenance situation.

Software tool support for the maintenance process has been identified as a key to achieving maintenance productivity gains (Schneidewind, 1987) as well as quality gains (Kim & Westin, 1988). Although software tools for maintenance offer potential quality and productivity increases, our knowledge of how the maintenance task can be, and currently is, supported by software engineering tools is limited. Organizations are adopting software engineering tools, and if they are used, productivity and quality benefits are being achieved (see Iivari (1996) for a brief review of this literature). A significant problem, however, is low utilization of these tools (Iivari, 1996; Kemerer, 1992).

Conventional software maintenance is concerned with changing existing program source code in a third generation programming language (3GL) such as COBOL. With the introduction of the newer types of Computer Aided Software Engineering (CASE) Tools, the maintenance process has begun to change. These newer CASE tools often include the ability for a programmer or analyst to create executable programs automatically from fairly high-level design specifications. The programmer in such environments no longer works directly in a 3GL, but in a much higher level environment. Such environments often include complex graphical representations of the systems functions as well as text-based representations. This has the effect of merging the traditional process of programming with the software design process. It is argued that this type of development may lower the cost of subsequent maintenance of these systems. This conjecture has yet to be addressed empirically.

It is extremely important for software authors and managers of the software users to understand how the customers and end-users of software actually choose to use certain functions or in some instances choose to forgo using automated methods and revert to manual methods. We believe that the key to understanding software use decisions lies in understanding how the functions provided by the software fit the perceived needs of the user.

We believe that the actual process of maintaining software at the design level differs from the process that occurs at the program level. Specifically, maintenance performed on 3GL code will have more use of transformation tool functionality and more software modification activities than will maintenance performed on designs created in a CASE tool environment. Use of representation tool functionality and performance of understanding activities will be higher in the CASE tool environments.

## Research Model and Hypotheses

Our research model, shown in Figure 1, was constructed from three models from the general MIS literature. The general Task/Technology Fit (TTF) Model (Goodhue, 1988b; Goodhue, 1992) provides the basic framework for examining tool usage. To adapt the general TTF model to our research context, we augmented the Software Debugging Model (Vessey, 1986), with the software understanding literature (Letovsky, 1987; Letovsky & Soloway, 1986). A software maintenance tool functionality model was developed based on the problem solving literature and the Functional CASE Technology Model (FCTM) (Henderson

& Cooprider, 1990). The integration of these models results in a more comprehensive framework relating maintenance task and technology characteristics to software tool usage. The use of a TTF model allows us to examine the linked behaviors of tool use and maintenance task together.

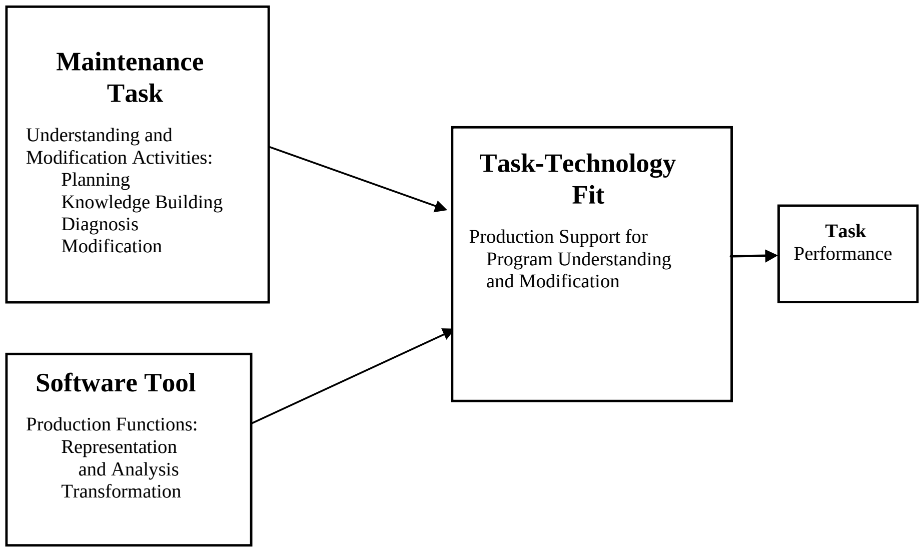

**Figure 1. Research Model**

A fundamental argument of our model is that software will be used if the functions available to the user support the activities of the user. The ability of software to support a task is expressed by the formal construct known as Fit. Fit is the matching of the capabilities of the technology to the demands of the task. Although a universal definition of task-technology fit does not exist, the literature contains several similar definitions, e.g., (Vessey & Galletta 1991, Goodhue, 1992). Essentially the TTF model states that "Fit" between task and technology is the extent to which a software tool is appropriate for a certain task. In this paper we employ the following definition of TTF (Dishaw & Strong, Forthcoming): "*Task-Technology fit is the matching of the functional capability of available software with the activity demands of the task.*"

The dimensions of fit between the maintenance task and software tool functionality are based on the matching of task activities with appropriate tool functionality. These constructs and the items and scales employed in their measurement are fully described in Dishaw (1992), Dishaw & Strong (1997), and Dishaw & Strong (Forthcoming). A very brief description follows.

**Maintenance Task** Our model of the maintenance task is based on the work of Vessey (1985, 1986) who developed a description of the actual types of actions engaged in by programmers during software maintenance. She identified Planning, Knowledge Building, Diagnosis, and Modification Activities in the maintenance process.

**Maintenance Technology** Henderson and Cooprider (1990) provide a description of the basic functions present in design support software (CASE). They identified two major dimensions of tool functionality: Production and Coordination functionality. Production functionality is the functionality that supports an individual programmer developing or changing software, which is the focus of this study. It includes representation, analysis, and transformation technology.

**The Nature of Fit between Maintenance Task and Technology** The task - technology fit (TTF) variable captures how well the set of production functions in the software maintenance tool supports the set of understanding and modification activities needed to accomplish a particular maintenance task. In this study as in our previous study (Dishaw & Strong, Forthcoming), the TTF variable is computed as the interaction term between a Task and Technology dimension. Our previous research demonstrated the use of a TTF model for professional software maintainers using 3GL languages. In this study, we investigate differences between 3GL maintenance and design level maintenance. Our research hypotheses pertaining to the model variables are:

| | |
|---|---|
| Hypothesis 1a) | 3GL software maintenance is dominated by the use of transformation tool functionality. |
| Hypothesis 1b) | Design level software maintenance is dominated by the use of representation tool functionality. |
| Hypothesis 2a) | Task-Technology fit for Modification-Transformation fit dimension will be higher for the 3GL environment. |
| Hypothesis 2b) | Task-Technology fit for Understanding-Representation & Analysis fit dimension will be higher for the Design-level environment. |

## Research Method

This project will involve the collection of data from two groups of undergraduate MIS students performing a maintenance activity. The first group of students will perform maintenance on a COBOL program. The second group will perform maintenance on a CASE based system design. The students will complete two short surveys. The first is an evaluation of the CASE tool environment and will be done prior to, and independently of the actual task assignment. A second survey will be completed regarding the task variables immediately after the assignment is turned in for grading. The survey instruments for this project have been previously developed and have established statistical properties (Dishaw & Strong, Forthcoming). We expect to be able to collect data from approximately 40 students from each group. Data collection is planned for the last two weeks in April.

Data analysis will be accomplished using SPSS for Windows. We will perform regression analysis and also path analysis using the AMOS structural equation modeling package integrated into SPSS. We will conduct a series of analyses that parallel those performed in our earlier studies. These results will be compared with those obtained in our field studies of working professional maintainers as well as with those that will be obtained from students maintaining software at the program level. These analyses will be completed before the August AIS conference.

## Conclusion

This paper describes a project arising from an ongoing research program on the nature and support of the software maintenance process. The research described here extends our previous work by examining maintenance at the design level. The goal of the overall research program is the development of our understanding of the underlying models of software tool utilization. By better understanding the circumstances which drive software utilization, we will be able to make recommendations to software tool builders as to those characteristics which are most needed in software tools which are designed to support the program development and maintenance processes. In addition, we will be able to make recommendations to managers and programmers regarding the support of the programming processes.

### *References*

References available on request from the first author (dishaw@uwosh.edu).