Operationalizing and Implementing the Concept of Responsiveness in a Management Tool

by

Kwang Seok Lee

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Industrial and Systems Engineering

APPROVED:

Harold A. Kurstedt, Chairman

C. Patrick Koelling

John W. Roach

D. Scott Sink

Paul E. Torgersen

January, 1991

Blacksburg, Virginia

Operationalizing and Implementing the Concept of Responsiveness in a Management Tool

by

Kwang Seok Lee

Harold A. Kurstedt, Chairman

Industrial and Systems Engineering

(ABSTRACT)

This research studies the concept of responsiveness in management tools to improve the match between a management tool and managers using it. This research first operationalizes the concept of responsiveness. The concept involves two components: a management tool and managers using it. Responsiveness is a property of the tool for matching the varying and evolving needs of the managers. It requires certain capabilities of the tool: functionality, user interface capability, and adaptivity. Responsiveness implies a certain process: observe and understand the manager, and interpret and implement his or her needs.

This study combines the concept with artificial intelligence technology and introduces the responsive system. A responsive system mainly consists of three parts: the application program, the user interface, and the responsive layer. The responsive layer is the core of a responsive system, employing the blackboard architecture and performing the responsiveness process.

To realize the structure of a responsive system, this study builds a prototype responsive management tool called MSLTRAIN, which helps managers schedule their training needs for computer packages. MSLTRAIN observes the behavior of the user, infers the user's preferences and decision patterns, reasons the user's goals, and responds appropriately based on the user's goals.

To evaluate the effect of responsiveness in MSLTRAIN on user performance and user satisfaction, this study conducts a laboratory experiment involving twenty human subjects. The subjects use and compare the prototype responsive management tool (R-MSLTRAIN) with the original

MSLTRAIN (O-MSLTRAIN) that doesn't have responsive features. The results support responsiveness has positive effect on user performance and user perception to a limited degree.

This study opens the door to management tools responsive to the varying and evolving needs of the managers. I believe responsiveness in management tools will improve success and sharing of the management tools. The concept of responsiveness and the structure of a responsive system can be applied to other domains.

Acknowledgements

Once I heard about a wise man in a desert who could hear the sound of water flowing far below the desert floor. For the last four and a half years, I was like a man digging alone in a desert to find water. Without the help of others, I might have been drowned in the desert. I would like to take this opportunity to express my appreciation to those who helped me in this research.

First, I thank Dr. Kurstedt. It was a great opportunity for me to work with him. He gave me insight and provided me critical directions on this research. Furthermore, he gave me an opportunity to work at MSL. It was a great experience for me. Also I thank Dr. Koelling, Dr. Roach, Dr. Sink, and Dr. Torgersen for their encouragement, time, ideas and feedback. I thank Dr. Williges for his review of my experimental design. His feedback was valuable for me. I thank Pedro Mendes. He always gave me good feedback on my study. His feedback helped me get ideas and directions. I thank the staff and graduate students at MSL: Jim Hughes, Anne Doss, Tom Polk, Kathryn Welch, Ellen Tomchin, Martin Jones, Teresa Simpson, Marcia Murawski, Martin Grunau, Jeff Keeling, John Polk, Seung-il Shin, Jana Moore, Asif Taiyabi, and Tim Kotnour. They helped me in terms of administration, formative evaluation, experimental design, pilot studies, and valuable discussions.

Finally, I want to dedicate this dissertation to my family. My grandfather and my father who died several years ago were always with me during the study, encouraging me. My mother, mother-in-

law, and father-in-law all supported me. I give special thanks to them. Above all, I thank my wife. For the last four and a half years, she has endured so much and received so little. I give her my thanks and my love, for I couldn't have done it without her. Last thanks go to my two sons born during the study.

Table of Contents

List of Illustrations

List of Tables

Chapter 1. The Problem and Its Setting

The purpose of this chapter is to put this research in perspective. In this chapter, I explain the problem and discuss the type of research, the delimitations, the assumptions, the significance, and the contributions of this research to the area of management systems engineering (MSE).

1.1 The Problem Statement

This research is about the transfer of a concept from the human world to the world of management tools[1]. The conceptual basis for this research is:

Question: How can we implement the concept of responsiveness into management tools?

Purpose: To provide a way for a management tool to match the varying and evolving needs of managers using it so ultimately to increase the tool's success and sharing.

---

[1] By *management tool* I mean anything we use to manage with; for example, information systems, plans, forecasts, organizational structures, models, expert systems, etc. (Kurstedt, 1985a).

Goal: Operationalize[2] and implement the concept of responsiveness into a management tool.

In this section, I explain the purpose in terms of motivation and the goal in terms of stages and subproblems.

1.1.1 Motivation

We help managers most by providing management tools to support their decision making. Often these tools fail. Kurstedt (1985a) claims 70 percent of all management information systems fail and Martin (1988) reports that "less than 5 percent of the money put into the nine software developments resulted in software which could be used as delivered or with minor changes" based on a Government Accounting Office study. I suspect this high failure rate of management tools is largely due to the lack of a total management system perspective. The success of management tools depends on the success of a management system the management tools support. Management tools are means to achieve all management system objectives. If the management system fails, the management tool fails whether it appears successful in itself. Often people are confused between their ends and means and tend to bend their needs (ends) to fit the tools (means) rather than vice versa. This *forced fit* causes the management system to fail in meeting system objectives. To rectify the confusion between ends and means, Kurstedt (1985a) has developed the Management System Model (MSM) shown in Figure 1 on page 3.

The MSM represents a structured approach we need for understanding and applying management tools. The MSM provides a simple framework describing a management system, its components, and their relationships. The MSM consists of three essential components: *who manages* (a manager), *what is managed* (operation), and *what is used to manage* (management tools). The *who*

---

[2] By *operationalize* I mean to come up with an operational definition that puts communicable meaning into a concept, that people agree on and can do business with (Deming, 1986).

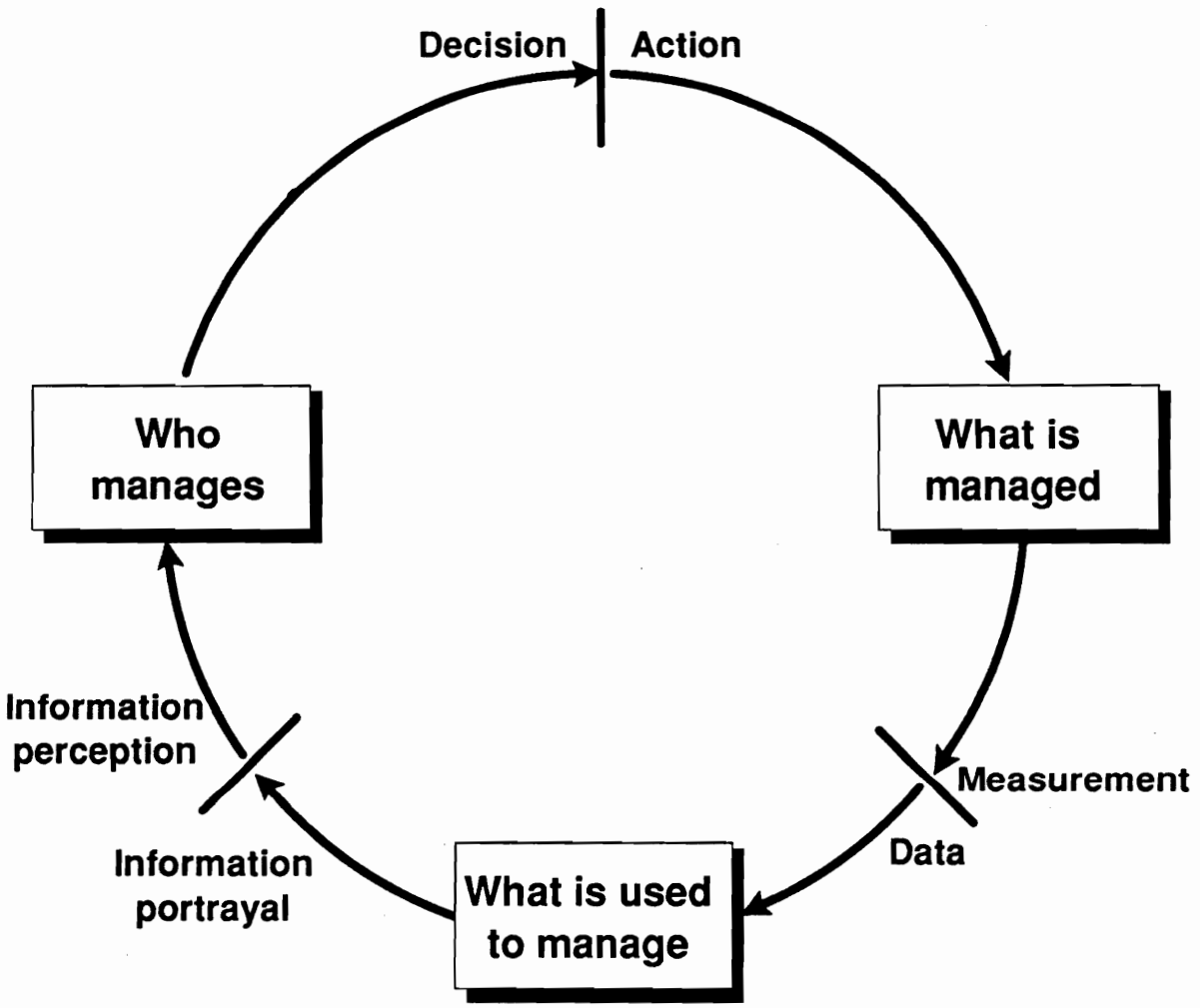


Figure 1. The Management System Model has three essential components. (Source: Kurstedt, 1985a)

manages component is anyone who uses information to make decisions resulting in actions affecting *what is managed.* The *what is managed* component includes the tangible physical things the manager is responsible for. The *what is used to manage* component comprises the tools with which we manage, such as organizational structures, information systems, plans, etc. Explicitly, the MSM separates means (what is used to manage) from ends (what is managed). A common source of management tool failure is the emphasis on *what is used to manage* as the end rather than the means. The MSM implies the focal point of managerial concern should be the physical things that constitute *what is managed.*

We balance the three essential components by matching both sides of each interface between the components (Kurstedt, 1985a). Kurstedt (1985a) argues lack of match in the MSM interfaces is another major source of management tool failure. Management tools must reflect *what is managed* and must be acceptable (comfortable and useful) to *who manages.* If our operation provides customer-oriented goods or services, a technical-function-oriented organizational structure will not work for us. If the manager is systematic and prefers definition and details, a colorful graphics package may not be appropriate.

The MSM highlights *who manages*, a component neglected in many management information system (MIS) research models. The MSM implicitly recognizes that different managers or a manager at a different time may 1) need different kinds of information in different formats, 2) perceive the same information differently, and 3) go through different decision making processes. The recognition of individual differences between managers implies management tools should reflect these individual differences to match the interface between a manager and his or her management tools. To be more successful, management tools should respond to changes in *who manages.* Usually conventional management tools have not reflected this changing nature of managers' needs. This is another source of management tool failure (Kurstedt, 1985a).

I'm also concerned with sharing management tools among different managers in terms of the converter of data, the information the tool produces, and the data the tool converts. We've experienced

a low sharing rate of management tools (MSL, 1986). I suspect one of the reasons is those management tools didn't have the ability to respond differently to different managers. In other words, they couldn't satisfy different needs of different managers. They might satisfy some of the managers they fit. For the rest of the managers, those management tools are useless. Therefore, if management tools could respond differently to different managers, those management tools will have higher sharing rates among managers.

In the context of success and sharing of management tools, we can imagine management tools *responsive* to the varying and evolving needs of the managers. Responsiveness is a valuable property human beings have shown with their intelligence for their success. For example, responsiveness is the credo of Management Systems Laboratories (MSL)[3]. Responsiveness characteristics mean MSL's ability and willingness to listen and understand the needs of sponsors and to provide good and timely solutions to the problems of sponsors. Responsiveness is a key and requisite characteristic we need for our success. So, why shouldn't successful management tools be responsive to us — the user ? Responsiveness implies intelligence. The rapid development of artificial intelligence (AI) supports our thoughts of intelligent management tools responsive to individual managers' needs. This research transfers the concept of responsiveness in the human world into management tools with AI technologies.

This research focuses on operationalizing and implementing the concept of responsiveness into an existing management tool. Therefore, the **research problem** can be stated as *operationalizing and implementing the concept of responsiveness into an existing management tool to provide a way for a management tool to match the varying and evolving needs of managers using it.*

---

[3] MSL is a research arm of Virginia Tech as a part of the Department of Industrial and Systems Engineering within the College of Engineering. MSL has been creating new methods to resolve persistent management questions.

1.1.2 Research Stages and Subproblems

Brinberg & McGrath (1985) divide research into three stages: *preparatory, exploratory,* and *confirmatory*. In the preparatory stage, we develop, clarify, and refine the problem. In the exploratory stage, we combine some content, some ideas, and some techniques to arrive at some findings. Usually, when we talk about "doing a study," we mean the exploratory stage. In the confirmatory stage, we explore the scope and limits of the set of findings. According to Brinberg & McGrath (1985), I'm explaining the preparatory stage of this research — defining the problem. In this section, I explain the exploratory stage and the confirmatory stage of this research by stating the subproblems (goals) in each stage.

Exploratory Stage. This stage is further divided into three steps:

1. **Operationalize** the concept of responsiveness in management tools.: The purpose of this step is to understand, in communicable terms, what constitutes responsiveness in management tools. The output of this step is the operational definition of responsiveness in management tools and frameworks that distinguish and clarify the concept of responsiveness.

2. **Reflect** the concept of responsiveness in the design of a responsive system.

   The purpose of this step is to provide a mechanism for interpretation of the concept of responsiveness. The output of this step is a structure of a responsive system — elements of the system, relationships between the elements, and a working mechanism.

3. **Realize** the structure of a responsive system by building a prototype responsive management tool in a representative domain of responsibility.[4]

---

[4] This step includes validation of the prototype. By *validate* I mean to ensure that a product or a program functions and contains the features as prescribed by its requirements and specifications (Deutsch, 1982).

The purpose of this step is to see the results of the interpretation in the previous step. The output of this step is a prototype responsive management tool realizing the structure of a responsive system.

Confirmatory Stage. This stage has one step:

1. **Evaluate**[5] the prototype responsive management tool in terms of fit between the management tool and the managers using it compared to a reference management tool (the same tool without responsive features).

   The purpose of this step is to evaluate the effect of responsiveness in a management tool. The output of this step is to see whether the prototype responsive management tool fits users better than the reference management tool.

1.2 The Type of Research

Brinberg and McGrath (1985), from a social science perspective, define three interrelated but analytically distinct domains research involves:

- Substantive domain — real-world systems and phenomena we're interested in,
- Conceptual domain — ideas that give meaning to substantive phenomena, and
- Methodological domain — some techniques or procedures with which we can study those ideas and phenomena.

---

[5] By *evaluate* I mean to determine the merit, worth, or value of something (Scriven, 1981).

This distinction is useful in defining my type of research. To define the research we start with one domain (conceptual, substantive, or methodological) and have *basic research, applied research,* or *technological research*, respectively. Depending on which domain we bring as a second domain, we have the *theoretical path*, the *experimental path*, or the *empirical path.* I modify the term *methodological domain* to *technological domain* to explain my research more properly.

This research is ***basic research.*** It starts from the conceptual domain — responsiveness in management tools. This research takes the ***experimental path.*** It starts from the conceptual domain, combines it with the technological domain, and applies that combination to the substantive domain.

Conceptual Domain: This research is about the concept of responsiveness in management tools. The concept of responsiveness involves two components: a management tool and a manager using it. Responsiveness is a property of the management tool including the manager. This research starts from the speculation that responsiveness in a management tool will improve the fit between the management tool and the manager using it, so ultimately increasing the success and sharing rate of the management tool. Most designers of management tools ignore dynamic characteristics of managers. Those tools don't respond to the managers' needs varying among different managers and evolving over time. Consequently, many management tools fail. The concept of responsiveness counts the varying and evolving aspects of the managers' needs.

Technological Domain: The technological domain provides a way (technique) to realize concepts and/or relations in the conceptual domain into a real-world system (infra-structure) in the substantive domain. The technological domain of this research is artificial intelligence (AI), especially the blackboard architecture. This research applies the blackboard architecture to reflect the concept of responsiveness into the structure of a responsive system. The blackboard architecture was first developed between 1971 and 1975 in HEARSAY-II speech understanding system research (Erman et al, 1980) and has been used in several areas. The blackboard architecture is useful to solve a

complex problem that needs integrated knowledge of several experts. This research combines the concept of responsiveness with the blackboard architecture to come up with the structure of a responsive system.

Substantive Domain: The substantive domain of this research is an area of management – identifying and scheduling training needs on computer packages in an organization. However, any other substantive domain would be proper, which is a characteristic of experimental research. Kurstedt (1986) developed a five-step instrument that helps managers with their training by addressing the organization's information needs and then helps identify which people should be trained on which computer package at what times. The worksheet was transformed into a computer package called MSLTRAIN, into which this research implements the structure of a responsive system.

1.3 The Delimitations

This research has boundaries as follows.

This research concerns only *computer-based* management tools, which is a part of management tools. Responsiveness doesn't necessarily imply computer-based. However, management tools such as plans, notebooks, organization charts, etc. can't be smart until we have such technology. We can begin to make a computer smart, so I'll deal with computer-based management tools.

This research focuses only on the reasoning aspect of responsiveness. This study doesn't concern the issue of response time, which is, however, important in the concept of responsiveness. I assume interactive computer systems respond in a reasonable time. After we understand the reasoning aspect of responsiveness, we can focus on the issue of response time.

This research looks at the fit between a management tool and the managers it supports only from an individual perspective, not an organizational perspective. For example, this study doesn't concern organizational structure, power distribution in an organization, or organizational environment. In terms of Markus & Robey (1983), this study focuses on User-System fit.[6]

The responsive features of the prototype management tool will be limited to its application domain, the capabilities of the development tool, and current technologies. In other application domains, the implementation of responsiveness might be different in terms of responsive features. However, the structure could be applied to other domains. The capability of the development tool limits the responsive features of the prototype. The prototype can't have features the development tool can't support. Current AI technologies also limit the responsive features. If we have more advanced technologies, the prototype can have more advanced responsive features.

The prototype management tool can't be responsive to all managers in every situation. Even in the human world, no person is responsive to all people in every situation. The prototype is responsive within its *notional world*. The notional world is a fictitious world in which a system would be perfectly adapted (Bechtel, 1985). A notional world of a system represents the beliefs the system holds about the user, task, and application characteristics of its environment (Totterdell et al, 1987).

1.4 The Assumptions

The assumptions[7] of this research are as follows.

---

[6] Refer to "2.2 The Concept of Fit in Management Tools" on page 21.

[7] I follow Leedy's (1980) definition of *assumption*: "a condition which is taken for granted and without which the research effort would be impossible."

The better a management tool fits individual managers it supports, the less likely it will fail. Based on this assumption, I can argue a responsive management tool will decrease the failure rate of management tools if responsiveness in a management tool improves the fit between the management tool and the managers it supports. Therefore, this study tests just whether or not responsiveness in a management tool improves the fit between the management tool and the managers using it.

If we know enough about a manager, we can build a management tool that fits the manager. We can elicit that knowledge from human experts or empirical research findings. This assumption is critical. Extracting the knowledge is not the objective of this research. I borrow the knowledge from literature or build the knowledge from my speculation.

Managers don't misuse a management tool intentionally. Users may misuse the management tool due to their misconceptions about it, but not intentionally.

The cost in terms of money and time to develop a responsive system is not important at this stage. The cost to be responsive may be more than what we can accept. However, I believe technical development will decrease the cost to a reasonable level at some time.

Interactive computer systems have the ability to respond in a reasonable time. This research doesn't concern the issue of response time. However, I'm not saying response time isn't important in the concept of responsiveness in management tools. I focus on the reasoning side of responsiveness rather than response time. We might study further the issue of response time. For example, reasoning takes time and we need timely response. Which one is more important? There is a problem of trade-off between right reasoning and quick response time.

1.5 The Significance of the Research

This research is significant in terms of the following aspects.

First, this research studies the concept of responsiveness in management tools extensively for the first time. Several people used the term *responsive* or *responsiveness*.[8] However, they didn't study the concept of responsiveness extensively. The concept of responsiveness is critical for the success and sharing of management tools. Many management tools have failed due to their inability to respond to changes in managers' needs (Kurstedt, 1985b). Changes may be caused by personnel changes or evolution of managers. Changes in the *who manages* component of the MSM cause the whole management system to be out of balance, especially at the information portrayal/information perception interface. This causes the management system to fail (Kurstedt, 1985a). Therefore, to significantly increase the success rate of management tools, we need to develop management tools reflecting the changing aspects of *who manages*. This research responds to that need and introduces the concept of responsiveness to management tools. I expect responsive systems to provide more effective support to managers and result in higher user satisfaction, higher user performance, and ultimately, higher success rates and more sharing of management tools.

Second, this research is integrative. This research transfers a concept in the human world to the world of management tools and combines the concept with existing technology. This research integrates studies in decision support systems (DSS), human computer interaction (HCI), and artificial intelligence (AI) areas into the concept of responsiveness.

Third, this research provides a real prototype responsive management tool developed onto an existing management tool. This research doesn't stop at the stage of discussing the concept and the

---

[8] For examples of these, see "2.1 Overview" on page 19.

structure. Furthermore, the prototype was developed onto an existing system. This implies we can implement the structure of a responsive system to other existing systems.

Fourth, this research involves evaluation of a responsive system. There has been little study on evaluation of adaptive computer systems even though there has been lots of literature on developing adaptive computer systems. This research provides a procedure and an instrument asking the user's response to the system's performance.

1.6 The Contributions of the Research

This research doesn't directly contribute to the knowledge of the technological domain, and contributes marginally to the substantive domain. This research primarily contributes to the body of knowledge of the conceptual domain — management systems engineering (MSE). This research adds knowledge of what we should consider in the design of management tools to balance the whole management system, and how we can actually design and develop such management tools. I suspect this knowledge will help decrease the failure rates of management tools and increase the sharing of management tools. This study brings artificial intelligence (AI) and human computer interaction (HCI) concepts and techniques to bear on the concept of responsiveness in management tools.

The output of this research is 1) the operational definition of responsiveness, 2) the structure of a responsive system, 3) a prototype responsive management tool realizing the operational definition of responsiveness, and 4) an evaluation result of the prototype in terms of fit. The operational definition of responsiveness will help decision support system (DSS) researchers and designers understand the concept of responsiveness, which provides a new perspective for considering individual users. The structure and the building procedure can be used to build other responsive systems, leading to a generalization of how to build the concept of responsiveness into management tools.

The evaluation procedure can contribute to the knowledge of evaluating responsive (or adaptive) systems in terms of methodology.

1.7 Terminology

This section provides the definitions of terms and abbreviations used in this document.

1.7.1 Definitions of Terms

adaptation any process whereby a structure is progressively modified to give better performance in its environment

application domain a domain or domains of responsibility a management tool is supposed to support

application program a functional part of software that performs and contains those functions which provide what the user is to accomplish

assumption a condition taken for granted and without which the research effort would be impossible

cognitive style the characteristic, self-consistent mode of functioning which individuals show in their perception and intellectual activities

context circumstances in which an event occurs

decision support support to help managers make decisions

decision support systems (DSS) all management tools integrated to support managers' decision making, which are not necessarily computer-based

domain of responsibility a bounded domain a manager is responsible for

evaluation the process of determining the merit, worth, or value of something; or the product of that process

hypothesis a logical supposition, a reasonable guess, an educated conjecture which may give direction to our thinking with respect to a problem and thus aid in solving it

interaction support support to help the user interact with the system

management system the set of responsibilities of any decision maker bounded as a system. For example, organization is a management system as a radio is an electrical system.

management tool anything we use to manage with; for example, information systems, plans, forecasts, organizational structures, models, expert systems, etc.

mental model an internal representation within the human brain of aspects concerning phenomena in reality

metadecision decision about a decision; for example, decision of how to decide, what to decide, which information to use, which model to use, etc.

metacommunication communication about communication between the user and the system, which includes help messages, user manuals, error messages, etc.

needs anything without which that existence of mode or level of performance would fall below a satisfactory or acceptable level

notional world a fictitious world in which a system would be perfectly adapted

operationalize to come up with an operational definition that puts communicable meaning into a concept, that people agree on and can do business with

predecision see *metadecision*

prototype first example from which others have been or will be copied or developed

user a person using a computer system to solve a problem in a domain of responsibility within a certain context

user interface the part of the system with which users interact

user model a collection of information about a user a computer system keeps for its response to the user

user modeling the process of building and maintaining user models

validation ensuring that a product or a program functions and contains the features as prescribed by its requirements and specifications; or the process of assessing the degree to which a design achieves the objectives of interest

1.7.2 Abbreviations

AI Artificial Intelligence

ANOVA ANalysis Of VAriance

CAI Computer-Assisted Instruction

DSS Decision Support System

DWIM Do What I Mean

HCI Human Computer Interaction

IFE Intelligent Front-End

ITS Intelligent Tutoring System

KS Knowledge Source

KSAR Knowledge Source Activation Records

MBTI Myers-Briggs Type Indicator

MIS Management Information System

MSL Management Systems Laboratories

MSM Management System Model

SAUI Self-Adaptive User Interface

SDLC System (or Structured) Development Life Cycle

1.8 *The Organization of the Document*

This document parallels the research stages[9] of this research. Chapter 1 and Chapter 2 correspond to the preparatory stage; Chapter 3, Chapter 4, and Chapter 5 correspond to the exploratory stage; and Chapter 6 and Chapter 7 correspond to the confirmatory stage of this research. The following summarizes each chapter in this document.

Chapter	*Description*
One:	explain the problem and discuss the type of research, the delimitations, the assumptions, the significance, and contributions of this research to the area of management systems engineering.
Two:	review literature related to this research, focusing on the concept of fit in management tools and how to get management tools to fit the managers.
Three:	provide a conceptual breakthrough on responsiveness in management tools, borrowing the concept from the human world. Explain a conceptual model of responsiveness, three constructs of responsiveness, and the responsiveness process, and finally define the concept of responsiveness in management tools.
Four:	combine the concept of responsiveness with technology. Define the responsive system and provide operational and structural specifications for a responsive system.
Five:	explain the prototype responsive management tool, MSLTRAIN, developed to realize the structure of a responsive system.
Six:	explain the laboratory experiment I did to evaluate the prototype responsive management tool and discuss the result.

[9] See "1.1.2 Research Stages and Subproblems" on page 6.

Seven: review the goals of this research, discuss generalization and limits of this research, and suggest further research in this area.

Chapter 2. The Review of Related Literature

2.1 Overview

The concept of responsiveness has not been fully studied and applied in the area of management tools. Several people use the term *responsiveness* or *responsive* in computer-based Decision Support Systems (DSS)[10] or Human-Computer Interaction (HCI)[11] areas. For example,[12]

> ... a greater effort will be required of the scientist to transcend his own individuality to develop aids *responsive* to the manager's needs (De Waele, 1978).
>
> ... This definition clearly places a greater burden on such systems to be easy and *responsive* to the needs of users (Rich, 1979).
>
> One of the major considerations when developing computer software has been to produce a *responsive system.* In the past emphasis has tended to be on the speed of response, ... the quality of the response has become a major concern (Fitter & Sime, 1980).
>
> If information systems are to be truly *responsive* to users' needs, change itself must be considered in the process of system design (Penniman & Dominick, 1980).

---

[10] I mean by DSS all management tools managers use to manage with. The key to DSS is integration of all these tools. DSS, in my context, is not always computer-based. This is similar to Katzan's (1984) *management support system* (MSS). I use the term *computer-based DSS* for the specific area of management tools referred as DSS by Keen & Scott Morton (1978) and Sprague & Carlson (1982).

[11] I mean by HCI a discipline studying the user interface — the part of the system with which the users interact. I'm using it in the same context as *man-computer interface, computer-human interface, or man-machine interaction.* HCI is a subdiscipline of DSS, as DSS is a subdiscipline of management. But HCI people treat HCI as an independent discipline.

[12] The italic emphasis is mine.

> How can interactive systems be made more helpful and *responsive*? This tool-independent system cuts down frustration by incorporating many of the communications shortcuts people use naturally (Hayes et al, 1981).
>
> It is widely agreed that systems should be *responsive* and flexible to support the changing needs of different users (Ein-Dor & Segev, 1981).
>
> Interactive intelligent systems often suffer from a basic conflict between their computationally intensive nature and the need for *responsiveness* to a user. An acceptable response time is needed ... (Gerring & Shortliffe, 1982).
>
> The second most important aspect of the human-computer interaction is *responsiveness*, but not merely that which is due to computer response time. ... Four other characteristics of responsiveness that affect user performance are listed in Figure C-6: interruption control, immediate status information, error recovery, and immediate acknowledge of input (Bair, 1984).

Although there exists some common ground in the use of the term *responsive*, there is no clear-cut definition on what it means. The implicit common ground in this literature is that the term *responsive* is used with the term *user's needs*; in other words, responsiveness implies fitting the user's needs. Management tools are meaningless unless they *fit* the managers using them. Therefore, the literature review focuses on the issue of *fit* in management tools. This literature review gives a basis for discussing the concept of responsiveness.

As to the issue of fit, basically I'm interested in the following two issues:

1. What does *fit* mean in management tools ?
2. How can we make management tools *fit* the users ?

For the first issue, I review the basic concept of fit and its meaning in management tools in Section 2.2. For the second issue, many researchers in computer-based DSS or HCI have tried to fit their systems to the user by incorporating adaptive design strategies, or building flexible, adaptable, or adaptive systems. In Section 2.3, I review evolutionary design and development strategy as a way to build a management tool to fit the managers using it. In Section 2.4, I review efforts to define adaptiveness and to build adaptive systems that by themselves fit the user. In Section 2.5, I review user modeling efforts to incorporate user models as a guide for a system to fit the user.

2.2 *The Concept of Fit in Management Tools*

We often say management tools should fit the managers using them. I think no one would argue about this. But, until we clearly understand fit, fit has no meaning in the design of management tools. First, I want to delimit the concept of fit. The term *fit* can be used at different levels such as

1. A system **S** fits a user group **U**,
2. A system **S** fits a user U_i,
3. A system's response fits a user's needs, and
4. A system's characteristic fit a user's characteristics.

This implies necessary or sufficient conditions for each level of fit. That is, 4) of the above list is a necessary condition of 3), 3) is a necessary condition of 2), and 2) is a necessary condition of 1). When I say the goal of responsiveness is to fit the system to the users, I mean 1) of the above list.

When managers use computer-based management tools, they deal with two different domains:

- Domain of the original problem they are trying to make a decision about, and
- Domain of interaction with the computer.

The main role of management tools is to support the first domain. However, supporting the second domain is also important in computer-based management tools. If the manager can't interact effectively and efficiently with the management tool, its good functionality means nothing. Here we need to distinguish the meaning of fit between these two domains. In literature, most DSS people have focused on the first domain and emphasized the fit of functionality of a system to the managers it supports. On the other hand, most HCI people have focused on the second domain and em-

phasized the fit of the user interface of a system to users. I review the concept of fit from the two perspectives separately.

2.2.1 Decision Support Perspective

Management tools help managers make decisions most by providing information they need. Thus, the concept of fit from the decision support perspective includes the following issues:

- Content — what information to present,
- Format — how it should be presented, and
- Time — when it should be presented.

The fit to the manager's needs or wishes, in terms of information content, presentation format, and presentation time is the bottom line of management tools (Rouse, 1984; Hollnagel, 1986). Fitting the needs or wishes implies two roles of decision support: confirming (wishes of) and complementing (needs of) the user. Sometimes these two roles are contradictory; what the user wants may not be what he needs. De Waele (1978), while defining criteria of fitting the manager's style, mentions this point. His first two criteria are: 1) the extent to which the decision support assists or augments the manager's preferred functions, and 2) the extent to which the decision support can fill the manager's secondary functions. This implies the concept of responsiveness should differentiate the user's needs and wishes and resolve the difference between needs and wishes. The issue of needs and wishes means the user's projection on the interface in Figure 7 on page 64 should be considered as multidimensional.

Ein-Dor & Segev (1981) talk about both individual fit and organizational fit of information systems. They define:

Individual fit	the extent to which different people need different information and the ability of a given system to meet those varying needs
Organizational fit	the extent that an information system is adapted to the particular organization in which it is used.

They view information system success as contingent on the concept of fit. Information systems that are successful in one organization (or individual) may not fit another organization (or individual) at all. They say:

> One may think of the individual and organizational environments of MIS as a pattern. The structure of an information system is another pattern. If the patterns interlock, success will result. Incompatible patterns, however attractive each may be alone, will achieve no results (Ein-Dor & Segev, 1981, p.227).

Markus & Robey (1983) describe the issue of fit as organizational validity. Organizational validity, for them, refers to the fit (or matching) between an information system and its organizational context of use. They identify four kinds of fit: 1) User - System fit, 2) Organizational Structure - System fit, 3) Power Distribution - System fit, and 4) Environment - System fit. Markus & Robey (1983) seem to be on the same track as Ein-Dor & Segev (1981) in terms of a contingency approach. They view fit as a relative and a descriptive concept rather than an absolute and a normative one. They argue fit doesn't always lead to effective use. Thus, an organizationally fit information system may be easily implemented but fail to bring any significant benefits. They find out there is some relationship between organizational validity, ease of implementation, and resistance to the system. However, they're not sure the fit (or validity) guarantees effective system use. They leave it as a question. The point is that fit and effective system use are different constructs.

Mallak (1986) describes the issue of fit of components within a management system as *balance* of a management system. By balance, he means "the system's ability to regain stability at an equal or higher level after accommodating change." He investigates the matching between the three components in the management system model and defines balance at three levels:

- At each COB (Condition Of Balance) — characteristics from the neighboring components match according to criteria such as the manager's reading level and the reading ease of reports, the manager's portrayal preference and the portrayal format of reports, the manager's decision style and the structure of the organization[13], etc.;
- At each interface — all COB's at that interface are balanced; and,
- For the entire management system — there exist desirable matches of important criteria (COB) among components of a management system, resulting in the overall success of that system.

To help identify the manager's needs, several researchers have developed several frameworks of decision support delimiting the domain where a management tool is applied (Simon, 1960; Anthony, 1965; Gorry & Scott Morton, 1971; Forrester, 1961; Blumenthal, 1969; Macintosh, 1981; Peterson, 1977; Mintzberg, 1980; Kurstedt, 1985a).

Sometimes managers fit themselves or their management systems to management tools — *forced-fit*. I should note here forced-fit is not really a fit. Kurstedt (1985b) warns the danger of forced-fit:

> Sometimes, in an effort to get aboard the computer bandwagon or shore up management inadequacies, some managers will force the components of their management system to fit an improperly developed MIS or an off-the-shelf MIS package. The new management system, forced to fit the MIS, may appear to be balanced, giving an illusion of success. However, because of the *forced fit*, the new management system probably no longer addresses the original system goals and objectives. The computer-based MIS may appear successful while the entire management system fails. Such an illusory success portends dire long-term consequences for real MIS success.

Many people have empirically studied fit between information systems and their users. One of the most frequently studied variables is cognitive style. There exist several kinds of cognitive style constructs. For example, Myers-Briggs Type Indicator (MBTI), cognitive complexity, impulsive/reflective, operation-learning/comprehensive-learning, risk handling style, locus of control, degree of dogmatism, convergent/divergent, degree of anxiety, assertive/passive, tolerance for ambiguity, degree of motivation, degree of compulsiveness, brain orientation, field dependence/field

---

[13] From his point of view, the structure of an organization is a management tool we manage with. We manage the organization, not the structure.

independence, heuristic/systematic, etc. (Sage, 1981). Many researchers, in the 70's and early 80's, have tried to relate cognitive styles to the design and development of information systems (McKenney & Keen, 1974; Driver & Mock, 1975; Kilmann & Mitroff, 1976; Bariff & Lusk, 1977; Barkin & Dickson, 1977; Dickson et al, 1977; Vasarhelyi, 1977; Benbasat & Taylor, 1978; De Waele, 1978; Lusk & Kersnick, 1979; Zmud, 1979; Henderson & Nutt, 1980; Robey & Taggart, 1981). But Huber (1983), in the review of these works, concludes:

- The currently available literature on cognitive styles is an unsatisfactory basis for deriving operational guidelines for MIS and DSS designs, and
- Further cognitive style research is unlikely to lead to operational guidelines for MIS and DSS designs.

Responding to Huber (1983), Robey (1983) agrees with Huber's assessment of the state of the art of cognitive style research but opposes the abandonment of cognitive style research in DSS design. He argues knowledge of cognitive style can be used as part of the design process such as team-building. Ramaprasad & Mitroff (1984) oppose the categorical view of cognitive styles. Relating Piaget's model of development of logico-mathematical structures (LMSs) to Jungian personality typology, they suggest the functional view of Jungian typology: sensing, intuiting, feeling, and thinking. Ramaprasad & Mitroff (1984) emphasize the strength of preference for all four functions, not merely by the dominant form of perception and of judgment. Ramaprasad (1987) suggests cognitive process research rather than cognitive style research. His point is similar to De Waele's (1978) idea of complementing the user's cognitive style, not just confirming it. As Huber (1983) suggests, the difference in individual cognitive styles can be accommodated by developing flexible, adaptable, or adaptive systems (Mann et al, 1986).

The discussion of the concept of fit in terms of decision support provides a basis for the concept of responsiveness in management tools. The goal of responsiveness is to fit the user's needs. The literature review in this section provides what we mean by fit and what we should consider to fit the user's needs, which frames the concept of responsiveness.

2.2.2 Interaction Support Perspective

To get help from computer-based management tools, managers should use (operate) computers. Using a computer is another task for managers. Often managers have frustrating experiences when using computers because of non-working commands, unkind error messages, slow response time, low reliability, need to learn too much, irrelevant help messages, etc. (Nickerson, 1981; Schneider & Thomas, 1983; Rubinstein & Hersh, 1984). These experiences result in less frequent and ineffective use of the system. Stewart (1986) makes this point with empirical findings. He defines:

Task fit — a measure of the correspondence between the service provided by the computer and the needs arising out of the manager's task

Ease of use — the degree of difficulty experienced by the manager in attempting to use the computer system

System potential — a judgment of the capability of the system to serve its users in terms of the flexibility and interrogative capacity it places at the disposal of its users

He suggests task fit will be increased as the system potential increases but the increase depends on ease of use. Even if the system potential is high, the task fit can be poor when it's not easy to use the system. This exemplifies the lack of fit in terms of human-computer interaction. To emphasize the importance of the user interface, Gaines (1979) uses the term *peopleware* compared to *hardware* and *software*. According to the degree a system is peopleware, the computer system can be an *intelligence amplifier* or *ignorance amplifier* (Gaines, 1979). Many user interface techniques have been developed to count the importance of user interface such as direct manipulation, pull-down menus, mouse, etc. (Schneiderman, 1987). It is the trend of 80's to separate the user interface from the main application program (Edmonds, 1981) and build domain-independent user interface management systems (UIMS) (Hayes et al, 1983; Buxton et al, 1983; Olsen et al, 1984; Foley et al, 1988).

Several researchers have suggested models or frameworks of human-computer interaction to help user interface designers figure out what to consider in their design to fit the users (Moran, 1981; Norman, 1984; Dix & Runciman, 1985; Nielsen, 1986; Clarke, 1986; Kammersgaard, 1988). Nielsen (1986) proposes a conceptual virtual protocol model incorporating seven levels: goal, task, semantics, syntax, lexical, alphabetic, and physical. All contact between the user and the system occurs at the physical level, but that is only to realize communication on the higher levels. He argues there should be a close connection between the dialogue protocol on the different levels. Norman (1984) defines four stages of user activities in human computer interaction: intention, selection, execution, and evaluation. He argues the importance of appropriate interface aids according to each stage.

There have been several approaches to discovering the concept of fit in human computer interaction, such as user-friendliness, usability, mental models, and human computer interaction models or frameworks.

User-friendliness: User-friendliness has been a buzzword for the human-computer interface. Most software companies claim their products are user-friendly. The concept of user-friendliness has driven computer designers to consider users and made it possible to use computers without sophisticated knowledge of the system. However, the definition of user-friendliness is not straightforward. People define the term in different ways such as *low subjective operating complexity, ability of the system to react as expected by the user, facilitating the user's access to the computer, satisfying all persons in the environment of the system,* etc. (Dehning et al, 1981). Dehning et al (1981) develop an operational definition of user-friendliness through a hierarchy of realizable system objectives: dialog flexibility, transparency, ease of learning, and reliability. I think their operational definition moves away from what is generally considered user-friendliness.

Norman et al (1986) show even a negative view of user-friendliness; they see user-friendliness as verbose or chatty interface with many long prompts, messages, and menus. Their point is that current user-friendly systems don't actually fit their users. Users may appreciate kind, detailed help messages the first time they use a system, but not every time. It's like "short-term friendly and

long-term hostile" (Klensin, 1982) or "novice-friendly" (Zachary, 1985). What is user-friendly to one person may not be user-friendly to others, and what is user-friendly to a person at one time may not be user-friendly at another time. I think we'd better define a new concept to address this issue, such as responsiveness, rather than try to enlarge the definition of user-friendliness like Dehning et al (1981).

Usability: To get over the negative view of user-friendliness, several researchers describe the issue of fit in terms of usability (Bennett, 1979; Shackel, 1984; Bennett, 1984; Whiteside et al, 1988). Shackel (1984) describes the advent of the concept of usability:

> It seems reasonable to suggest that the first industrial revolution created the concept of usability, or rather its underlying necessity, by the complexity and growing pace of technological change. This change has caused ever greater distance in time and space between designers and users, and so inevitably has resulted in the making of products which may be useful but are certainly not usable (p. 45).

He distinguishes between utility and usability:

Utility the capability in machine functional terms to fulfill the specified range of tasks within the specified range of environmental scenarios

Usability the capability in human functional terms to be used easily (to a specified level of subjective assessment) and effectively (to a specified level of performance) by the specified range of users, given specified training and user support, to fulfill the specified range of tasks, within the specified range of environmental scenarios

The concept of usability extends the concept of user-friendliness practically. It focuses on system design processes as well as system design goals. In this sense, Shackel (1984) defines four components of usability: user, task, tool, and environment. According to him, usability depends on the dynamic interplay of these four components. Gould & Lewis (1985) suggest three principles of system design for usability: "early and continual focus on users; empirical measurement of usage; and iterative design whereby the system (simulated, prototype, and real) is modified, tested, modified again, tested again, and the cycle is repeated again and again." Furthermore, Whiteside et al

(1988) use the term *usability engineering* to emphasize the design process for usability. However, the purpose of usability is still to make the system usable for more users, not to fit the system to individual users. The concept doesn't incorporate the issue of fit fully.

Mental Models: Cognitive psychologists have focused on mental models users are developing while using computers (Card et al, 1983; Saja, 1985; Norman et al, 1986). According to them, during an interaction with a computer system, the user will form a mental model of the interaction that contains what he or she believes are the functions, capabilities and limitations of the system (Shackel, 1984). Cognitive psychologists argue we should consider these mental models when designing user interfaces. The design criteria may be: promoting 1) confidence in the user towards the ability and correctness of the system, 2) satisfaction in the user, and 3) efficiency by the user in completion of the task (Saja, 1985); and reducing the user's misconceptions towards the system (Miller et al, 1987).

Norman (1983) points out four different things we need to consider when we talk about mental models: the *target system* the person is using; the *conceptual model* of the target system invented by designers to provide an appropriate representation of the target system; the user's *mental model* of the target system, which will be constrained by such things as the user's technical background, previous experiences with similar systems, and the structure of the human information processing system; and the *Scientist's conceptualization* of a mental model.

The intent of this category of researchers has been quite ambitious, but I don't think we've had significant progress to incorporate mental models fully into the design of interactive systems. As Norman (1983) points out, often mental models are too complex and too dynamic to model. I agree with the idea of considering mental models in the design process, but the idea needs much more development.

The discussion in this section implies the concept of responsiveness in management tools should consider the issue of user interface. In computer-based management tools, the user interface is

critical because it provides a way for the user to interact with the system. Concepts and models reviewed in this section provide what we mean by fit and what we should consider to fit the user's needs in terms of interaction support, which constitutes the concept of responsiveness.

2.2.3 Operational Measures of Fit

Researchers have operationalized the concept of fit largely based on two perspectives:

1. User performance perspective, and
2. User perception perspective.

In the evaluation experiment of this research, I operationalize the fit in the interface from both user performance and user perception perspectives.

User Performance Perspective: The user performance perspective is based on the proposition that if the user performs better in a system **S1** than in a system **S2**, we say the system **S1** fits the user better than the system **S2**. Many people have used user performance as the criteria of fit such as:

- Task completion (Benbasat et al, 1981);
- Time – decision time (Maskery, 1984; Benbasat et al, 1981; Ambardar, 1988; Dickson et al, 1977; Vasarhelyi, 1977; Tyler, 1986), training time (Ambardar, 1988), scanning time (Greenberg & Witten, 1985);
- System use – number of errors (Maskery, 1984; Ambardar, 1988), error rates (Greenberg & Witten, 1985; Tyler, 1986), number of calls for help (Maskery ,1984); and
- Performance – performance scores (Lusk & Kersnick, 1979); productivity, quality of user performance (Cats-Baril & Huber, 1987).

These measures can be said to be *objective* in the sense that they don't depend on a person's perception. However, as Markus & Robey (1983) point out, it is questionable that fit and user performance are the same construct.

User Perception Perspectives: The user perception perspective is based on the proposition that if the user likes a system **S1** better than a system **S2**, we say the system **S1** fits the user better than the system **S2**. The assumption of the proposition is the user is the best one to determine whether a system fits him or her. Therefore, this perspective mostly uses Likert-scale questionnaires which ask the user about his or her perception of the system's performance or responses (Dzida et al, 1978; Bailey et al, 1983; Ives et al, 1983; Blaylock & Rees, 1984; Potosnak, 1984; Stewart, 1986; Cats-Baril & Huber, 1987; Chin et al, 1988; Hiltz & Johnson, 1990). These measures can be said to be *subjective* in the sense that they depend on a person's perception. I think this perspective suffers the problem of validity because of this subjectivity and the lack of the user's ability to evaluate the system.

2.3 Evolutionary Design and Development Strategy

A design and development strategy includes how we can build and implement decision support systems successfully. The issue here is which strategy is effective and efficient for getting systems to fit to the user. The possible questions might be:

- How can we effectively extract what the user needs or wants ?
- How can we effectively and efficiently implement them ?

I review two basic approaches — the traditional approach, or system (or structured) development life cycle (SDLC) design and evolutionary (or iterative, adaptive) design or prototyping.

2.3.1 Traditional Approach

The System (or Structured) Development Life Cycle (SDLC) is a traditional approach in the sense that it has been in practice for almost three decades. Though the literature shows different definitions and terms, SDLC is quite standard (Sprague & Carlson, 1982). SDLC is highly structured, consisting of the following phases:

1. Planning and programming,
2. Conceptual design,
3. Detailed design, and
4. Implementation (Murdick & Munson, 1986).

In the SDLC approach we start actual implementation after considering the whole life cycle of the system beforehand. This increases the effectiveness of project management. How the system designer identifies user requirements before implementing a system is one of the top issues in the SDLC approach. The degree to which the system reflects the user's real requirements depends on the communication between the user and the designer. Often the communication before implementation is not easy and not effective (Martin, 1988). Furthermore, users often don't know what they need or want before they see the system. Even if the designers identify the real requirements of the users, the requirements can change because the users are evolving over time. The result is users are adapting their requirements to the system. This is not always bad. But this is one of the reasons why so many information systems have failed.

2.3.2 Evolutionary Design Approach

Different from the traditional approach, the evolutionary design[14] approach, on the other hand, tries to build a system through an iterative and adaptive process of learning and evolution. The idea is the users should be involved actively in the design process to enhance user satisfaction and system utilization.

Frameworks

Keen (1980) suggests a framework to describe what we should consider in an evolutionary (adaptive) design process. Simply his framework includes the system, the user, and the builder (Figure 2 on page 34). The arrows in the figure mean some action or influence. Keen's point is all components mutually affect each other, so the DSS design should be an adaptive process taking all links into account. According to him, there is no final DSS. The design process is a never-ending process. His framework is descriptive and normative. The framework describes continuous, dynamic interactions between the user, the system, and the designer and suggests the design process should incorporate this.

The evolutionary design process was motivated from the fact that the environment, the tasks, and users of DSS are subject to frequent change. To respond to the change, DSS must be flexible. Sprague & Carlson (1982) suggest four levels of flexibility[15] desirable for a DSS as shown in Figure 3 on page 35:

1. Flexibility to solve – give the user the ability to confront a problem in a flexible, personal way;

---

[14] Instead of the term *evolutionary*, Keen (1980) used the term *adaptive* and Sprague & Carlson (1982) used the term *iterative*. But the meaning is the same. Each term emphasizes a different aspect of of the evolutionary design. Maybe *prototyping* is a more familiar term to some people.

[15] They use the term *flexible* in a generic sense.

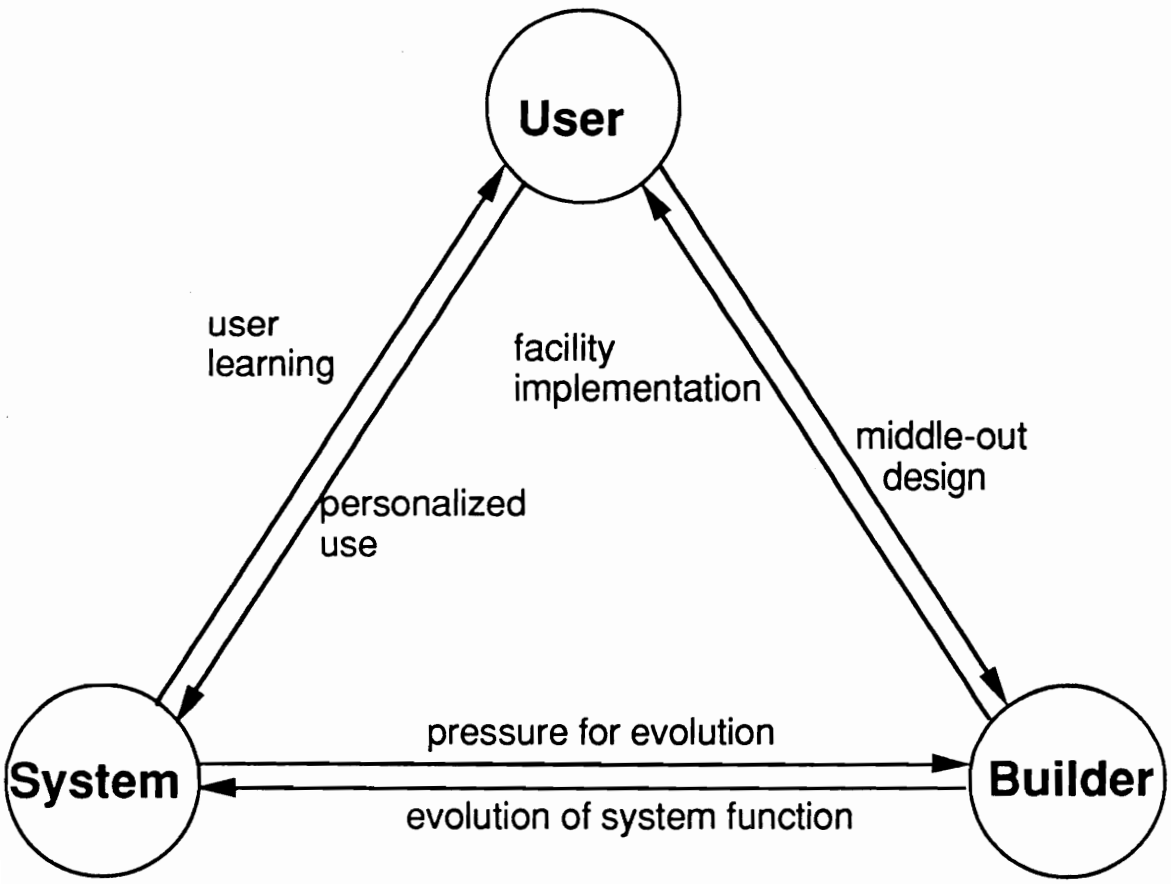


Figure 2. The evolutionary design process links the user, the system, and the builder. (Source: Keen, 1980)

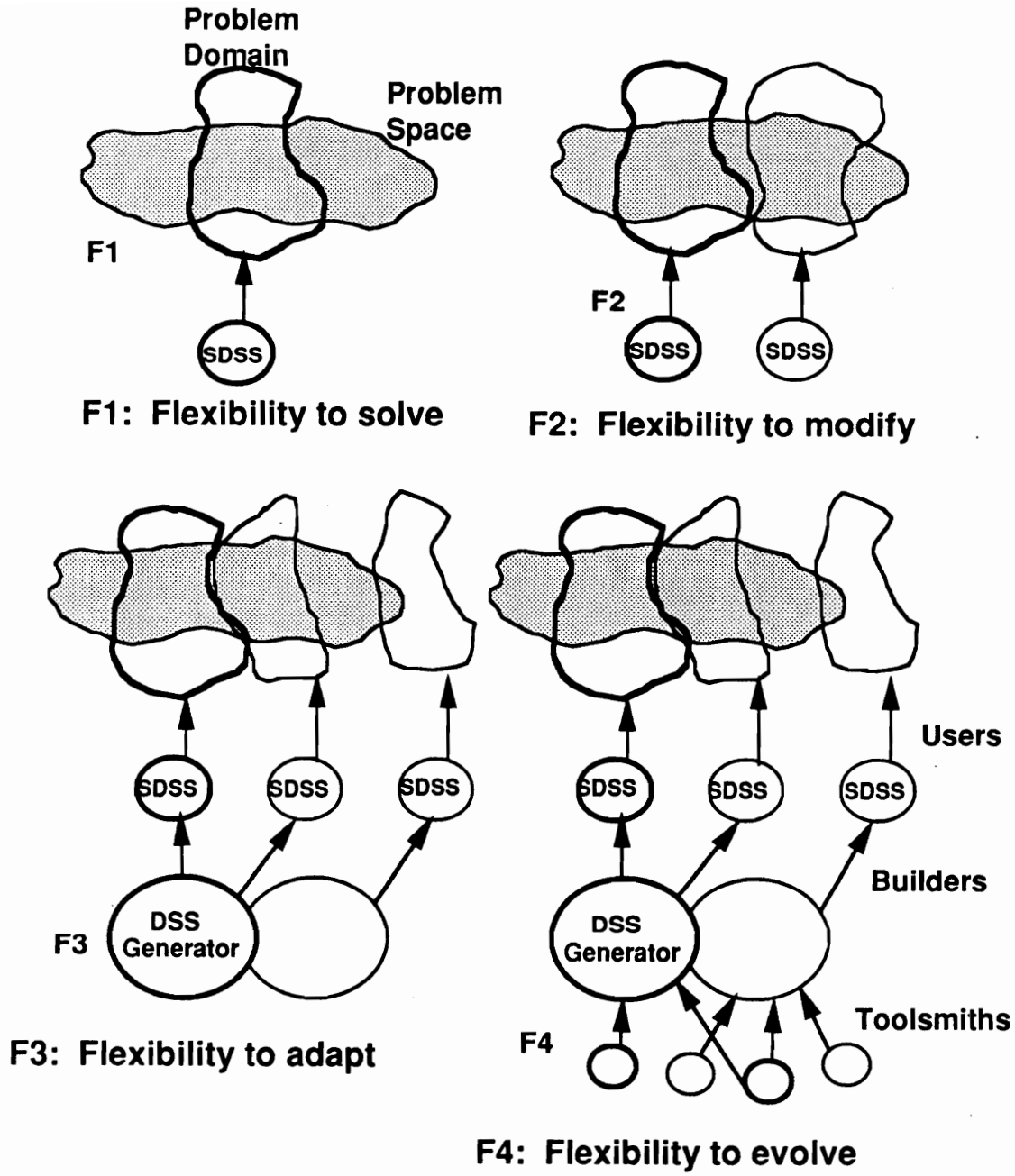


Figure 3. Flexibility is the key to meet the changes. (Source: Sprague & Carlson, 1982)

2. Flexibility to modify — the ability to modify the configuration of a specific DSS (SDSS) to handle a different or expanded set of problems;

3. Flexibility to adapt — the ability to adapt to environmental changes that require a completely different SDSS; and

4. Flexibility to evolve — the ability to evolve in response to changes in the technology.

Their concept of flexibility is very broad. They include problem domain and technology in their context. Keeping a system flexible, in their context, is crucial to the evolutionary design process.

Steps: The steps of evolutionary design include:

1. Identify an important subproblem,
2. Develop a small but usable system (prototype) to assist the decision maker,
3. Refine, expand, and modify the system in cycles, and
4. Evaluate the system constantly (Sprague & Carlson, 1982).

Williges et al (1987) present two kinds of evaluation in the process in terms of human computer interaction:[16]

- Formative evaluation — dealing with techniques for obtaining user feedback to aid the designer in making decisions for design revisions during the iterative design, and

- Summative Evaluation — testing the final design configuration to ensure it is functioning properly.

---

[16] Scriven (1967) used the terms *formative evaluation* and *summative evaluation* in the context of education. The purpose of formative evaluation is to improve the system while it is still fluid. Summative evaluation of a program is conducted after completion and for the benefit of some external audience or decision maker (Scriven, 1981).

General Considerations: Although the result is limited to some extent, Alavi & Handerson (1981) report evolutionary design is more effective than SDLC in both system utilization and user satisfaction based on their experiments. Sprague & Carlson (1982) also criticize the traditional SDLC approach in terms of design effort and effectiveness. They propose evolutionary design as an alternative and characterize it as a short development cycle, repeated many times. Many researchers support the evolutionary design approach (Lucas, 1978; Naumann & Jenkins, 1982; Nosek, 1984; Liang, 1987; Sauter & Schofer, 1988). But the evolutionary design strategy is not always the best for every situation. Lucas (1978) argues:

> This approach is best where the goals and objectives of the system are unclear. Systems most amenable to evolutionary design are probably those in the management control and strategic planning cells of the Gorry and Scott Morton framework. Evolutionary design is also appealing for applications in unstructured cells of the framework. For a clearly defined accounts-receivable system, evolutionary design may not be necessary (p. 51).

Often the evolutionary design results in less efficient systems and development processes more difficult to manage and control (Pliskin & Shoval, 1987).

The evolutionary design approach provides a process of fitting the system as the user evolves. If the system takes the role of a designer, the system should go through the same process. Therefore, the evolutionary design approach provides a clue to the responsiveness process in management tools.

2.4 Adaptive System Approaches

Another possible approach to fit the system to the user is to build adaptive systems that intelligently fit themselves to the user without the designer's or the user's intervention. The rapid development of artificial intelligence supports the idea of adaptive systems. Many people have been trying to build adaptive systems. In this section, I review the adaptation perspective in general and several adaptive system approaches in computer-based DSS and HCI areas.

2.4.1 Adaptation Perspective

Fitting a management tool to the user is like an adaptation process, whether it's done by tool designers (custom-fit), by users (flexible or adaptable), or by the tool itself (adaptive). Holland (1975) defines *adaptation* as "any process whereby a structure is progressively modified to give better performance in its environment" and Conrad (1983) as "a kind of optimization process coping with uncertainty in the environment." Many people view adaptation as a general and fundamental process and try to come up with a unified theory of adaptation (Holland, 1975; Staddon, 1983; Conrad, 1983; Totterdell et al, 1987). Especially Totterdell et al (1987) relate the adaptation process and levels of adaptivity to the design of adaptive computer systems. Reviewing the work on adaptation will help us understand the fitting process of management tools.

2.4.1.1 General Setting

The term *adaptation* (*ad* + *aptare*, to fit to) is used in fields as diverse as evolution, ecology, economic planning, control, artificial intelligence, computational mathematics, sampling, inference, organizational development etc. (Holland, 1975; De Greene, 1982). As a starting point on the study of adaptation, I rephrase Holland's (1975) fundamental questions on adaptation:

- To what is the organism (organization, or computer system) adapting ?
- What in the organism is adapting to the environment ?
- How is the organism adapting to the environment ?

In this section, I review answers to these questions.

Environment: Adaptation relates a system to its environment. When we talk about an adaptation, we are making at least an implicit reference to the corresponding environmental object(s) or

function(s) to which it is adapted (Totterdell et al, 1987). To understand to what part of the environment the organism is adapting, I'll use the concepts of niche and notional world. *Niche* is hard to define but can be understood well through examples. Staddon (1983) describes *niche*:

> Organisms are machines designed by their evolution to play a certain role. This role, together with the environment within it is played, is called the organism's *niche*. For example, most cats — tigers, leopards, mountain lions — play the role of solitary hunters; wolves and wild dogs are social hunters; antelope are social grazers; and so on. The niche defines the patterns of adaptive behavior to an animal's survival and reproduction. (p. 1)

Every system (organism, organization, or computer system) has its own niche. Furthermore, relating it to the design of adaptive interfaces, Totterdell et al (1987) introduce Dennett's concept of *notional world* to resolve the difficulty of explaining the possibility of false beliefs which don't relate the system to its environment. They quote Bechtel (1985):

> A notional world is a fictional world that would be a perfect *niche* for a system with a particular set of internal states (i.e., a world in which the system would be perfectly adapted). The notional world of a system will usually partly correspond to the world the system actually inhabits (otherwise the subject would not survive on its own) but it may be differ in some respects ... what we are doing in such cases is tantamount to saying that the internal states of the system make it best adapted to a world which differs in various ways from the one it inhabits (p. 484).

Totterdell et al (1987) suggest specifying the notional world for each adaptive computer system, which requires specification of the beliefs that the system holds about the user, task, and application characteristics of its environment. They argue the advantages of the use of notional world in adaptive computer system design are:

1. The problem is simplified by virtue of the fact that we no longer attempt to define a generic adaptive system that can cope with all environments. Instead we look at manageable, albeit incomplete, versions of the real world.
2. By comparing notional worlds we may be able to determine the generality of an adaptive system to other user groups and applications.
3. Once the notional world has been specified, failures of the system to meet ideal rationality in this world can be explained by referring to the design level of the system (p. 719).

The concept of notional world can be applied to the design of a responsive system. A responsive system can't be responsive to all users in every domain. To define the notional world of a responsive system will be critical to its design.

Structures and Operators: For the second question, we can think of two things: structures that are modified during adaptation and operators that yield the modification. Adaptation implies a change in system structure. Some of the structures are undergoing adaptation according to the

system's niche. The change can't occur without operators. Holland (1975) describes the structures and operators:

> We can see at once that adaptation involves a progressive modification of some structure or structures. These structures constitute the grist of the adaptive process, being largely determined by the field of study. Careful examination of successive structural modifications generally reveals a basic set of structural modifiers or operators; repeated action of these operators yields the observed modification sequences. ... A system undergoing adaptation is largely characterized by the mixture of operators acting on the structures at each stage (p. 3).

Adaptation Mechanism: Environmental changes cause the system not to be in the state of fit. The adaptation process starts with information on the environment received by the system. The information is a seed to the adaptive plan of the system. Holland (1975) describes the adaptive plan[17] :

> A system undergoing adaptation is largely characterized by the mixture of operators acting on the structures at each stage. The set of factors controlling this changing mixture — the adaptive plan — constitutes the works of the system as far as its adaptive character is concerned. The adaptive plan determines just what structures arise in response to the environment, and the set of structures attainable by applying all possible operator sequences marks out the limits of the adaptive plan's domain of action. Since a given structure performs differently in different environments — the structure is more or less fit — it is the adaptive plan's task to produce structures which perform "well" (are fit) in the environment confronting it. "Adaptations" to the environment are persistent properties of the sequence of structures generated by the adaptive plan (p. 4).

An adaptive plan produces a sequence of structures by selecting operators from a set of operators. The crux of the problem for the adaptive plan is that initially the system has incomplete information about which structures fit best. To reduce this uncertainty, the plan must test the performance of different structures in the environment, i.e., the fitness of the structures for the environment. The *adaptiveness* of the plan enters when different environments cause different sequences of structures to be generated and tested (Holland, 1975). Therefore, the scope for adaptation in any given system is a function of the ability of the system to interpret its environment (observe), to display behavioral changes (behave), and to usefully reorganize its internal structure in response to a change in the environment (learn) (Totterdell et al, 1987).

---

[17] Totterdell et al (1987) call the adaptive plan *adaptation strategy*. According to them, an adaptation strategy is a mechanism "which changes the state of a system in response to a change in the environment such that the system is better adapted than it would have been had it remained in its original state."

Adaptation perspective provides a direction on how a management tool can perform the responsiveness process and what we should consider: environment (the user), structures (the tool itself), operators, and the adaptation mechanism.

2.4.1.2 Levels of Adaptivity

Totterdell et al (1987) suggest a taxonomy for adaptive systems. They distinguish between tailorable systems and adaptive systems. They say tailoring is probably better seen as a parallel dimension which runs through all the levels and is concerned with the amount of control a system has in negotiating change. At the lower adaptivity levels, the user will simply be able to switch the system between different states; whereas, at the higher levels, the tailoring procedure will require more than intelligence on the part of the system in negotiating change.

According to Totterdell et al (1987), the first level adaptive (*self-adaptive*) system has the ability to detect a change in the input it receives from the environment and as a result make a change in its output behavior. The system has adaptation strategies to generate a different response for each event it discriminates. But the system doesn't have the ability to learn about the environment. Its behavior is like a *knee-jerk* response. The system is purely reactive.

The second level adaptive (*self-regulating*) system has the ability to evaluate its behavior against the outcomes based on the purpose of the system. The system has a multiple output potential for a single input, and the selection of the actual output is determined by the environment and the system's evaluation function. That is, the system learns about the environment. But the system doesn't internalize the learning.

The third level adaptive (*self-mediating*) system has the ability to generate and test its hypotheses on the environment, and internalize the model of the environment. By using this internal representation of the environment, the system has the ability to pre-evaluate the outcome of various

courses of its behavior. That is, the system evaluates the effects the system has on the environment. The locus of control is not only on the environment side but on the system side.

The fourth level adaptive (*self-modifying*) system has the ability to reason about its environment and reorganize its internal rules in response to changes in the environment.

These levels of adaptivity provide a taxonomy for levels of intelligence we can put into a responsive system.

2.4.1.3 Implications to the Design of Adaptive Systems

The adaptation perspective provides some implications to the design of adaptive computer systems. Totterdell et al (1987) say:

> There is a parallel between evolution and an iterative computer system design methodology in which a prototype design is tested on potential end users before being modified by an external agent such as the designer in order to improve the system's adaptedness. This opens up the interesting possibility that with the advent of self-adaptive computer systems some of the iterative design decisions made by the designer will be transferred to the system. It will no longer be necessary to identify the design that best fits the generic set of users; instead it will become the responsibility of the run-time system to decide on the design or strategy that best fits the individual user at any particular time. The designer's role will be to make decisions about the range, content and control of strategies to be built into the system.

Totterdell & Cooper (1986) identify prerequisites to the design of the adaptive system:

- Define the purpose of the adaptation,
- Specify the environment (user, application, etc.) to which the system is adapting,
- Identify the sources of evidence on which the system can adapt, and
- Select what to adapt within the system.

Once we have specified these prerequisites and the notional world, we can proceed to specify the design level functionality that will enable the system to respond rationally in its world.

The prerequisites apply to the design of a responsive system. We should consider what a management tool is to accomplish as a result of being responsive, specify the notional world (user, domain, and time period) of the management tool, identify which user attributes are critical, and decide which tool attributes can be adapted to the user's attributes.

2.4.2 Several Approaches to Adaptive Systems

In computer-based management tools, adaptation can occur from two sides: the user and the system. The user may adapt to the system by learning how to use the system to accomplish a task, or the system may adapt to the user by modifying itself appropriately to the user. The system adaptation to the user seems to be the current trend. Innocent (1982) mentions three ways of system adaptation to the user: 1) by designers based on user feedback, 2) by the user, and 3) by the system itself based on the acquired knowledge about the individual user. Many researchers adopt the third approach and build some kind of adaptive systems.

2.4.2.1 Intelligent Front-End (IFE)

When we use different languages, we need a translator who understands each language. Users often have difficulty communicating with computers. To help users, the system needs a translator, which is often called an *intelligent front-end*[18] (IFE) (Bundy, 1984; Clowes et al ,1985; Sleeman, 1985; Totterdell & Cooper, 1986; Alty & McKell, 1986). The role of IFE is to translate the user's input into terms the main application can understand and the main application's output into terms the user can understand. Therefore, the IFE is located between the user and the system.

---

[18] The term *front-end* is from the term *front-end processors* in computer science. A front-end processor is a small, limited capability, digital computer that is programmed to replace the hard-wired input and output functions of a central computing system. The front-end processor thereby permits the host computer to perform its primary functions with little regard for the slower input/output activities associated with large-scale multiprogrammed or time-shared computing systems (Ralston & Reilly, 1983).

Bundy (1984) defines an intelligent front-end as a user-friendly interface to a software package[19] which otherwise would be technically incomprehensible and/or too complex to be accessible by many potential users. According to him, an IFE basically does two things:

1. Specifies the user's problem and synthesizes instructions to run the package, and
2. Interprets the results of the package into the language of the task specification and explains the the results to the user.

To do these things, an IFE should have 1) an explicit representation of the user's task or problem, 2) a user model representing the user's understanding of the package, and 3) a model of the package describing what kind of task the package can and can't cope with (Bundy, 1984). To emphasize the user modeling aspect in an IFE, Sleeman (1985) suggests the idea of user modeling front-end (UMFE) in computer-assisted instruction (CAI). In terms of interaction support, Interlisp's Do What I Mean (DWIM; Teitelman & Masinter, 1981) facility can be seen as an IFE.

In information retrieval systems, some people have used the term *intelligent intermediary* instead of IFE (Brooks & Belkin, 1983; Smith, 1985; Chignell & Hancock, 1989). The intelligent intermediary should have the ability to *understand* rather than just decode utterances of the user — to infer the goals or intentions of the user — and the ability to adapt the system's advice to the different needs of the user (Allen & Perrault, 1980; De, 1986).

2.4.2.2 Predecision Support Systems

Managers usually make decisions on how to make decisions before they actually make decisions. For example, before making actual budgeting decisions, managers decide what kind of information they need and how they'll approach the problem. Wedley & Field (1984) call these decisions *pred-*

[19] He uses the term *package* in a general sense. It might be a traditional software package, a database, a compiler, or a computer network.

ecisions.[20] These predecisions are influenced by situational variables that interact with and, in turn, affect an ultimate decision, like a contingency approach. Extending Vroom & Yetton's (1973) situational leadership model, Wedley & Field (1984) build a computerized predecision support system helping managers come up with possible decision styles and decision methods. Their contribution is in the computerization of the models, which reduces the manager's perception of the complexity of the models and provides useful on-line help to the manager.

The concept of predecision support can be applied to other areas such as selection of statistical methods or mathematical models based on the characteristics of the problem, selection of problem solving methods, selection of simulation models and techniques, etc. If we develop situational models for these cases and incorporate these models into a computer system, the system can automatically select a method or methods appropriate for the problem situation. Remus & Kottemann (1986) apply this concept to an Artificially Intelligent Statistician (AIS), which 1) parses the query, 2) determines data appropriate to answering the query, 3) determines correspondingly appropriate statistical technique(s), 4) determines how to handle issues such as outliers, multicolinearity, and transformations, and 5) produces output to the decision maker which minimizes possible biasing effects.

The concept of predecision support can be applied to the concept of responsiveness. A predecision support system can be a front-end to main application systems. It's like an expert on methodology.

2.4.2.3 Self-Evolving DSS

In Figure 2 on page 34, if the system itself takes the role of the builder, we can call that system self-evolving to some degree. Liang (1987) develops this concept and implements it into a prototype. The idea of his prototype is simple; the prototype itself selects a default policy of presenting

---

[20] Mintzberg et al (1976) call them *metadecisions.*

information that matches the evolution of the user's preference of presentation format. The prototype presents information in one of four formats (bar chart, line chart, pie chart, table). If the user wants another format, the user can select one he wants. The prototype focuses on the evolution of the user's preference of presentation format. He defines three types of evolution of the users' behavior: uniform, systematic, and random. His prototype has three default policies of presenting information: fixed, dynamic, and non-default. The prototype assigns a default policy by comparing the performance of each default policy based on the user's previous usage records and selecting the best one. Liang (1987) defines performance in terms of additional actions the user should take. Liang (1987) also proposes an architecture of a self-evolving DSS comprising five major components: database subsystem, model base subsystem, user interface subsystem, evolution management system, and a control mechanism for coordinating those subsystems. The key components in his architecture are the evolution management system (EMS) and the control mechanism. The EMS handles system usage data of the user and rules that determine an appropriate default policy for a specific version of the DSS. His self-evolving path for a specific user is determined by preset rules based on the user's system usage data. But his prototype is so simple it doesn't seem to satisfy his notion of self-evolving ideas and the proposed mechanism. His self-evolving idea helps me figure out how a responsive system incorporates the evolution process, and the architecture he proposes provides some idea of the structure of a responsive system.

2.4.2.4 Intelligent Tutoring Systems

For effective teaching, teachers should know what students know and don't know. Moreover, knowing the individual characteristics of the students will increase the effectiveness of teaching. This idea can be transferred to Computer-Assisted Instruction (CAI). It has long been the objective of CAI people for the computer to understand the status of the student and give instructions based on understanding — intelligent tutoring systems (ITS) (Sleeman & Brown, 1982).

The two important issues in ITS are how to model the student and how to design and execute appropriate instructions. For the first issue, lots of user modeling literature exists (Goldstein, 1982; Clowes et al, 1985; Sleeman, 1985; Zissos & Witten, 1985; Brecht & Jones, 1988). I review this literature in "2.5 User Modeling Approach" on page 52. For the second issue, Peachery & McCalla (1986) apply planning techniques in AI — goals and operators. In their plan-based CAI, a planner develops a teaching plan fitting the particular student based on instructional goals and preconditions of the student. An executor uses the teaching plan to guide the student through the course. Macmillan & Sleeman (1987) develop a self-improving instruction planner (SIIP) that improves its planning behavior based on a student's response to instruction. They use the blackboard architecture for the architecture of SIIP.

2.4.2.5 Self-Adaptive User Interfaces

Innocent (1982) suggests the self-adaptive user interface (SAUI) concept and an architecture of a self-adaptive interface system that includes soft facade, monitor, control, and expert modifier (Figure 4 on page 48). The soft facade means a facade[21] that can be easily tailored. The expert modifier mimics a system designer engaged in adapting a facade to individual users. The monitor collects and sends information about the facade components to evaluation agents in the expert modifier. The control modules take the information from control agents in the expert modifier and change the components of the facade accordingly.

Instead of SAUI, some people use the term *intelligent interfaces* (Rissland, 1984; Daniels, 1986; Quinn & Russell, 1986; Rouse et al, 1987; Chignell & Hancock, 1989) or *smart interfaces* (Smith, 1985) to emphasize intelligent characteristics of user interfaces. Intelligent interfaces, for example,

---

[21] The facade is the system as it appears to a user. The facade of a system contains aspects such as documentation and user-help facilities, as well as those aspects to protect privacy, system integrity, etc. (Innocent, 1982).

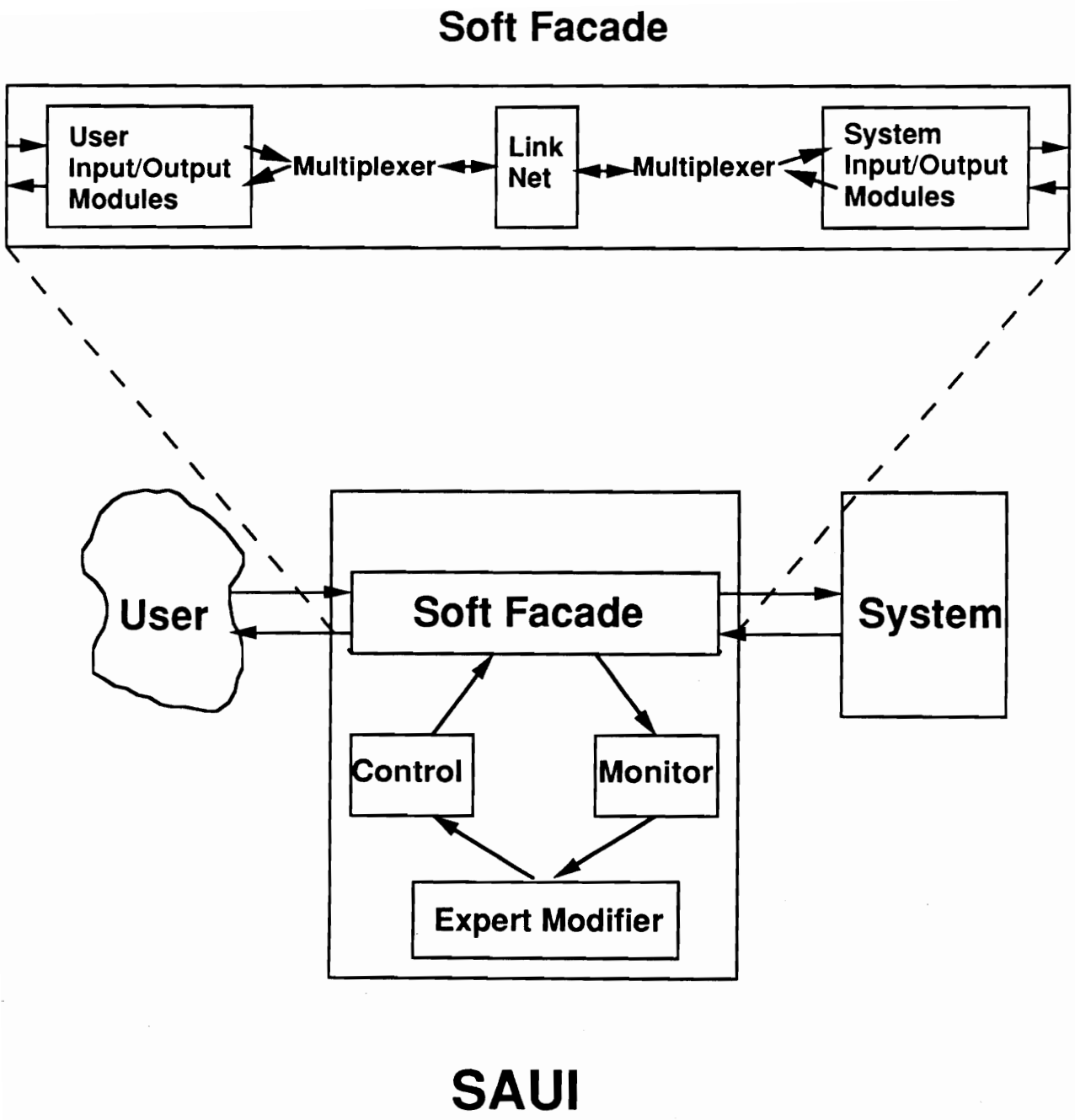


Figure 4. An architecture realizes the Self-Adaptive User Interface. (Source: Innocent, 1982)

attempt to intelligently bridge the gap between users and computers as a intermediary or a translator (Chignell & Hancock, 1989).

Trevellyan & Browne (1987) suggest the concept of self-regulation[22]. While evaluating the work of Greenberg & Witten (1985), they find the personalized directory works well only in some situations. The *notional world*, explained in "Environment" on page 38, of the personalized directory (Greenberg & Witten, 1985) includes only those individuals whose pattern of phone calls follows the Zipf distribution (Trevellyan & Browne, 1987). In other words, sometimes adaptation brings a worse result than no adaptation. They argue the system should evaluate the adaptation and should feedback the evaluation result to regulate the system itself.

The concept of self-adaptive user interface and the proposed structure provides a clue for the structure of a responsive system.

2.4.2.6 Active Decision Support Systems

Decision support systems support the user in making decisions. The initiation of support may be passive or active. Manheim (1988) argues conventional DSS are passive because they do only what the user explicitly tells them to do, and proposes the concept of *Active DSS* (ADSS). He defines an ADSS as "a DSS operating in part almost completely independent of explicit direction from the user to provide support which the user finds useful." He emphasizes the symbiotic nature of the user and the system and proposes an architecture of ADSS comprising the user interface, user-directed processes, user-directed process manager, computer-directed processes, computer-directed process manager, history recorder, history inference processor, and display manager (Figure 5 on page 50).

---

[22] Totterdell et al (1987) call this as the second level of adaptivity. See "2.4.1.2 Levels of Adaptivity" on page 41.

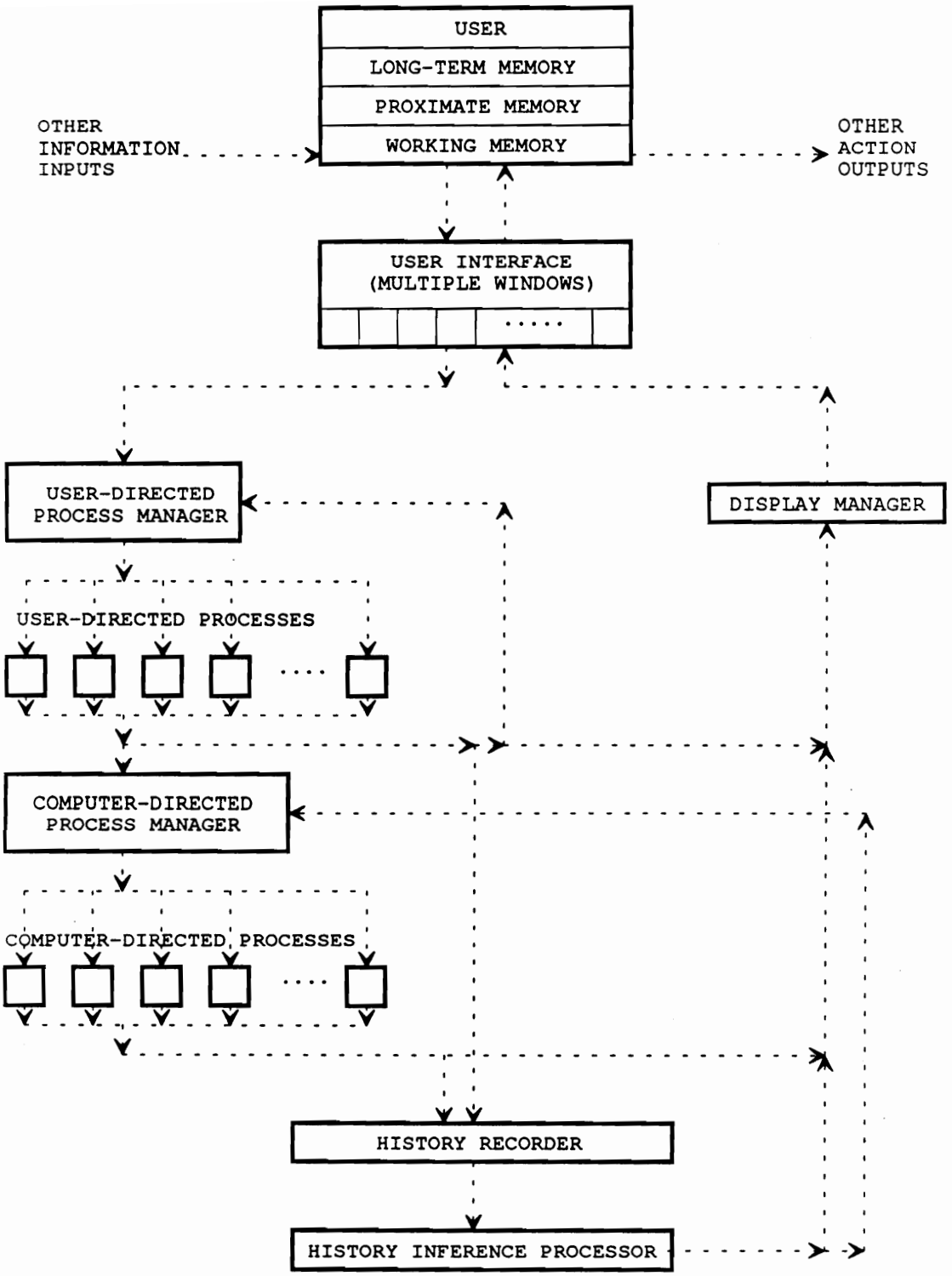


Figure 5. An architecture realizes Active DSS. (Source: Manheim, 1988)

Raghavan & Chand (1989) illustrate some aspects of ADSS:

- monitoring the decision making process of the user and detecting inconsistencies and problems,
- automatically scheduling and carrying out the necessary activities by understanding the intent and the context of the decision maker,
- alerting the decision maker to the aspects of the problem and problem-solving process he or she may be unconsciously ignoring,
- carrying on conversations with the decision maker that can lead to proper formulation of decision problems,
- stimulating creative ideas,
- serving as a sounding board for ideas, and
- criticizing the decision maker's actions and decisions from various perspectives.

The idea of ADSS implies implicit identification of the user's needs, which should be an operational characteristic of a responsive system. A responsive system should be active in Manheim's (1988) terms.

2.4.3 Arguments against Adaptive Systems

There exist arguments against adaptive systems. Adaptive systems have their own problems. Greenberg & Witten (1985) argue:

- Adaptive systems can be unstable because, while the system is trying to adapt to the user, the user is also trying to adapt to the user. This may undermine the user's confidence in the system.

- Adaptive systems, by definition, have control over some aspects of the interaction that the user may want to have himself or herself.

- Adaptive systems are hard to implement because of their complexity.

- Adaptive systems may generate inaccurate user models, which lead to wrong responses to the user.

- Adaptive systems are hard to evaluate empirically because of their complexity.

Even in the human world it is very hard to assess a human's mind. We need the balance between the need for a consistent and uniform interface and the need for change to suit user, task, and system environments (Innocent, 1982).

We should consider negative aspects of adaptive features of a responsive system. The negative aspects should be resolved in a responsive system.

2.5 User Modeling Approach

To effectively communicate with others, we should know them. That knowledge is a model of the person we communicate with. With the model we understand the person and direct our behavior to him or her. Likewise, computer systems maintain a user model or user models, whether deliberately or accidentally.[23] The kind of user model a computer system has is critical to the fit of the system to the user, especially in adaptive systems. A system based on a wrong user model can't fit the user. In this section, I review the literature on the following topics:

- What is a user model ?

---

[23] A user model may be a generic model usually of what the designer would want or what the designer thinks a representative person would want. And a representative person usually represents everyone a little bit and no one a lot.

- How and from where can we (or the system itself) get information to fill the user model ?
- How can we represent the user model within the system ?

2.5.1 User Models

2.5.1.1 Definition and Use

A user model is a collection of knowledge and/or information keeps that contains explicit assumptions on all aspects of the user relevant for the dialog behavior of the system (Kass & Finin, 1989). In this sense, all computer programs have a user model, encoded by the programmer when writing the program. Kass & Finin (1989) argue user models can be used to support the tasks of "1) recognizing and interpreting the information-seeking behavior of a user, 2) providing the user with help and advice, 3) eliciting information from the user, and 4) providing information to the user" (Figure 6 on page 54).

2.5.1.2 Dimensions

Many researchers have specified a broad definition of a user model by describing a user model in different dimensions.

a. Ownership: Rohr & Tauber (1984) define a model as "a representation of an object where relevant properties of the object are mapped onto a specific substrata different of that from the object (symbolic signs, electrical circuits, etc...)" and argue the selection of relevant properties assumes a *subject* deciding the relevance. Murray (1987) extends this notion and defines the ownership of user models. She differentiates among:

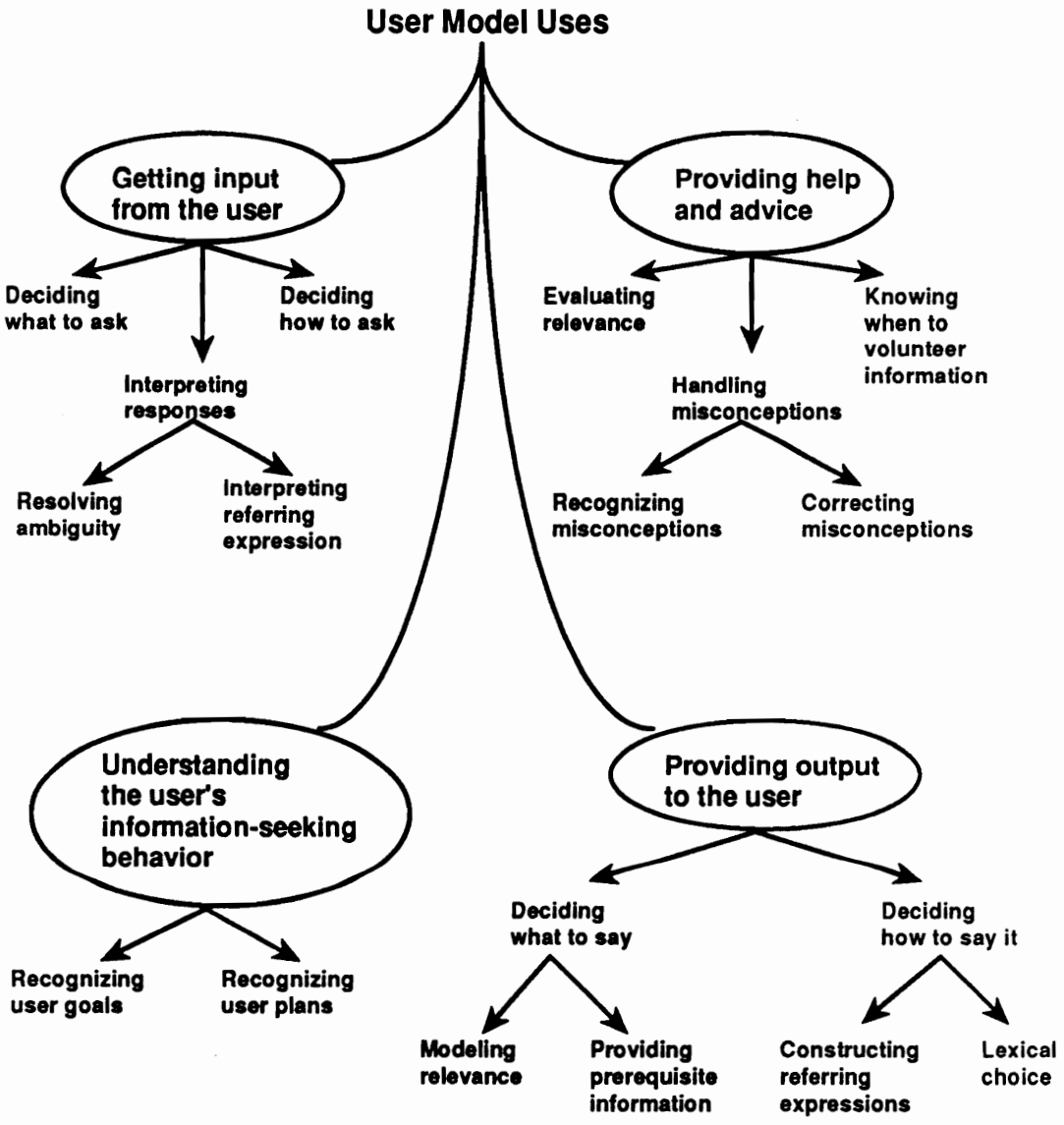


Figure 6. User models can be used for several purposes. (Source: Kass & Finin, 1989)

- *Designer's model* of user characteristics for the purpose of interface and system design;
- *User's mental model* of the computer system, derived from interaction with that system and from prior experience, for the purpose of achieving or understanding a specific task; and
- *System's model* of user characteristics for the purpose of tailoring the interaction or making the dialogue between user and system adaptive.

In this sense, I mean by user model a system's model of user characteristics.

b. Representation Coverage: The representation coverage of the user model means how many users a user model represents. Rich (1979) identifies two kinds of coverage of user models:

Canonical a user model represents all users, and

Individual a user model represents an individual user.

c. Content: The content of the user model is information about the user that may be useful to an application. Rich (1979) categorizes content of a user model into:

Short-term relatively short-term user characteristics such as the problem the user is trying to solve, or goals and plans specific to the problem; and

Long-term fairly long-term user characteristics such as areas of interest, expertise, beliefs, or attitudes.

Spark Jones (1986) categorizes content of a user model into:

Objective user characteristics that can be described objectively such as age, sex, name, etc.; and

Subjective user characteristics that can't be described objectively such as beliefs, goals and plans, intentions, etc.

d. Purpose: The purpose of a user model means where to use the user model. Kass & Finin (1989) and Murray (1987) categorizes purpose into:

Descriptive treat the user model simply as a data base of information about the user, and

Prescriptive simulate the user; for example, check the generated response of the system against the user model to see whether the user finds it misleading or ambiguous.[24]

Kass & Finin (1989) argue this dimension is an important, but overlooked point.

e. Modifiability: I mean by modifiability whether or not user models can be changed. Murray (1987) distinguishes between:

Static the user model is fixed over time, and

Dynamic the user model represents the user changing over time.

f. Attitude: I mean by attitude how to respond to other system modules. Kass & Finin (1989) categorize attitude into:

Passive the user model only responds to information requests, and

Active the user model, on its own initiative, volunteers information to another module.

g. Others: Sleeman (1985) adds one dimension that he called "the nature and form of the information contained in the user model and the inference engine needed to interpret that information." Sridharan, in Sleeman et al (1985), suggests other dimensions such as richness of response space, who bears responsibility, and deterministic versus probabilistic.

---

[24] Murray (1987) uses the term *predictive* instead of *prescriptive*.

Several dimensions of user models imply what kind of user model a responsive system should maintain. They also provide some guidance on what to observe and understand in terms of the user in a responsive system.

2.5.2 User Modeling

User modeling is a verb meaning actually building and maintaining the user models. The focus of user modeling is: 1) acquiring information to fill the user model, and 2) representing that knowledge in the system. Related to these two functions, Rich, in Sleeman et al (1985), raises two questions:

- How can models of users be inferred from their behavior and used in reasoning to improve the performance of a target system ?

- How can models of a large number of users be maintained efficiently so that each is available when necessary but system performance does not degrade even if the user population is very large ?

For the first question, AI comes in. For the second question, database technology comes in. I now review these two functions as acquisition and representation of user models.

2.5.2.1 Acquisition of User Models

Acquisition of user models is an important and interesting problem. Many researchers have argued the availability of a rich user model would enhance the performance of their systems. The major problem is the difficulty in obtaining the knowledge that belongs to the user model. There may be several ways to acquire knowledge about the user. Designers might get the knowledge from the user

and implement it in the system. Users themselves might fill the values of the system's user models, such as user profiles. In both cases, the user models are provided from outside of system. Rich (1979) calls this kind of user model *explicit user models*. However, the system itself might infer the user models based on keys users provide or the behavior of users. Rich (1979) calls this kind of user model *implicit user models*. Here, Rich's concern is *who finally specifies* the user models. I'm interested in implicit user models because this research starts from the recognition that managers are different. As to *implicit user models*, Kass & Finin (1989) are interested in *how to specify* the model. They categorize the system's user model acquisition techniques into:

Explicit Acquisition identifying pieces of pre-encoded information that apply to the user by observing the user's behavior. It's a recognition or classification process. It includes overlay modeling, perturbation modeling, stereotyping, plan recognition, etc.

Implicit Acquisition no explicit pre-encoding of information about the potential users. The system itself builds the user models implicitly.

I now review these user model acquisition techniques because they provide some guidance on how to observe and understand in a responsive system.

Overlay Model: An overlay model is one of the techniques used for student modeling in intelligent tutoring systems. It describes the learner in terms of a subset of the expert's skills and is a set of hypotheses regarding the student's familiarity with the expert's skills (Brecht & Jones, 1988). Overlay models highlight the differences between the expert's and the student's knowledge (Sleeman, 1985). Goldstein (1982) applies the overlay approach to a *genetic graph* to record the student's progress at the game of WUMPUS. However, the overlay model can't correctly model a user who has misconceptions or information sources other than the system (Kass & Finin, 1989).

Perturbation Model: A perturbation model is also one of the techniques used in intelligent tutoring systems. The perturbation model still uses the student's knowledge in terms of the expert's, but

augments additional information, such as misconceptions, which might differ from the expert's knowledge (Kass & Finin, 1989).

Stereotyping: When we have little information of a person, often we judge him or her by assessing his or her stereotypes.[25] Rich (1979) implements the idea of stereotyping into user modeling. She builds a system, called GRUNDY, whose purpose is to recommend library books to users that they might enjoy reading, based on what the system knows of the user's characteristics. The system uses stereotypes to build a specific user model for a specific user. We can get the information by asking the user; but too many questions may frustrate the user. That is why she used stereotypes. First, the system requests adjectives and nouns from the user that might be self-descriptive. These terms *trigger* certain stereotypes predefined in the system which contain some assumed values for certain attributes relevant to book topic choices. After building the model of an individual user by combining all activated stereotypes, the system then recommends books congruent with predictions generated from the model.

Stereotyping enables a system to develop a large set of beliefs about a new user very quickly. But it has some problems. Kass & Finin (1989) points out the problems of stereotyping: inappropriate level of stereotypes, difficulty of building stereotypes, and lack of justification.

Plan Recognition: Plan recognition is a technique to identify a plan the user might have to accomplish goals when using the system. It includes taking as input a sequence of actions performed by a user, inferring the goal pursued by the user, and organizing the action sequences in terms of a plan structure (Schmidt et al, 1978). This technique is based on a *planning paradigm* in artificial intelligence. Many people apply the idea of planning to user modeling (Boguraev, 1985; Davenport & Weir, 1986; De, 1986; Desmarais et al, 1987; Gilbert, 1987; Thomas et al, 1987).

---

[25] McCauley et al (1980) define *stereotypes* as "generalizations about a class of people that distinguish that class from others" and stereotyping as "differential trait attribution or differential prediction based on group membership information." For example, if someone is a judge, he or she is probably over forty, well-educated, fairly affluent, honest, and well-respected in the community.

Implicit Acquisition: Kass & Finin (1989) suggest the idea of implicit acquisition in general user modeling. They point out some problems the explicit acquisition techniques might have: 1) need of a significant amount of knowledge about users to be hand-coded by the system designer, 2) time consuming pre-encoding, and 3) domain dependency. They define some acquisition rules independent of domains such as:

- The user modeling module can assume that a statement made by the user is believed by him or her.
- The user modeling module can assume that presuppositions of statements made by the user are believed by him or her.
- If the user says P, the user modeling can assume that the user believes that P, in its entirety, is used in reasoning about the user's current goal or goals.

However, as they admit, the implicit acquisition technique has not been well studied, and may be slow and lead to wrong user models.

2.5.2.2 Representation of User Models

Zissos & Witten (1985) classify three forms of user model representations: a parametric form, in which a small set of values is identified to characterize the user for a particular task; a discrete-event form, in which the command or keystroke sequence is massaged into a finite-state or finite-context model; and a frame-like form, in which domain knowledge is used to identify explicitly the user's performance with each concept.

A responsive system maintains individual user models. The representation techniques provide guidance on how a responsive system incorporates user models.

Chapter 3. The Concept of Responsiveness in Management Tools

3.1 Overview

Responsiveness is a valuable property human beings or groups of human beings have shown. We can think of lots of examples: responsive secretaries to their bosses (Belker, 1981), responsive nurses to their patients (Murray, 1980), responsive salespeople to their customers (Grikscheit et al, 1981), responsive governments to their people (Zeigler & Tucker, 1978), responsive schools to the parents of their students (Johnston, 1988), responsive teachers to their students (Sleeman & Brown, 1982), etc. A responsive secretary observes the boss, anticipates what the boss needs, and implements it. Belker (1981) exemplifies:

- The secretary scans an incoming letter to see what file needs to be pulled for the boss to process the matter with thoroughness, and attaches the file, putting the latest letter on top.

- The secretary looks at the boss's calendar, sees what meetings and appointments the boss has scheduled for the next week, and asks himself or herself what items the boss is likely to need for these appointments and what he or she can do to help the boss prepare for them.

The responsive secretary has the ability and willingness to listen and understand the boss's needs and provide appropriate and timely services. This implies three things:

- The responsive secretary matches his or her response to the boss's needs — provide *appropriate* and *timely* services.

- The responsive secretary goes through some process to match — *listen*, *understand*, *interpret*, and *provide*.

- The responsive secretary has some abilities to perform the match — *ability* and *willingness*.

In this chapter, I discuss these three things in more detail: a conceptual model of matching, the responsiveness process, and three constructs of responsiveness; and integrate those into the definition of responsiveness in management tools.

3.2 A Conceptual Model of Matching

When we talk about responsiveness, we implicitly assume there are two components: one that is responsive such as a secretary, a nurse, a salesperson, a government, a school, and a teacher; and the other to which the one is responsive such as a boss, patients, customers, people, parents, and students, respectively. I call them *server* and *sponsor*, respectively. Responsiveness is a property of the server towards the sponsor. Figure 7 on page 64 represents a simple model describing the re-

lationship between the server and the sponsor.[26] The model includes a sponsor, a server, and an interface screen between them. To put this into the perspective of management tools, the server is a management tool (*what is used to manage*) and the sponsor is the manager using the tool (*who manages*). The interface screen in Figure 7 on page 64 represents an interface in the Management System Model (MSM) shown in Figure 1 on page 3: decision/action, data/measurement, or information portrayal/information perception interface. A successful management system balances the three components, by focusing on these three interfaces.[27] I explain the model first from a static point of view, and then from a dynamic point of view.

3.2.1 Static View of the Model

First, assume one sponsor. In the model, the sponsor projects its state to the left side of the interface screen as *needs* and the server projects its state to the right side of the interface screen as *responses*. Here, the interface is not a physical one, but a conceptual one. The size and position of each projection depends on the states of the sponsor and the server. Conceptually, the intersection of the projection areas defines the degree of match between the sponsor and the server. In the secretary example, think of the boss seeing incoming letters the secretary prepared. The boss's characteristics, preferences, situation, etc define the boss's state. This state is projected to the interface screen as *needs* such that the boss 1) wants to process recent letters first, 2) needs to make some decisions, and 3) wants some information showing the trend of recent sales. On the other hand, the secretary's capabilities, resources, preparation, etc define the secretary's state. This state is projected to the interface screen as *responses* such that the secretary puts the latest letter on top and for the letter requiring it, attaches a line chart of the company's recent sales. In this case, a match occurs

---

[26] This model with Figure 8 on page 68, Figure 9 on page 72, and Figure 10 on page 75 represents the conceptual model of this research.

[27] In this research, I'm interested primarily in the information portrayal/information perception interface and secondarily in the measurement/data interface.

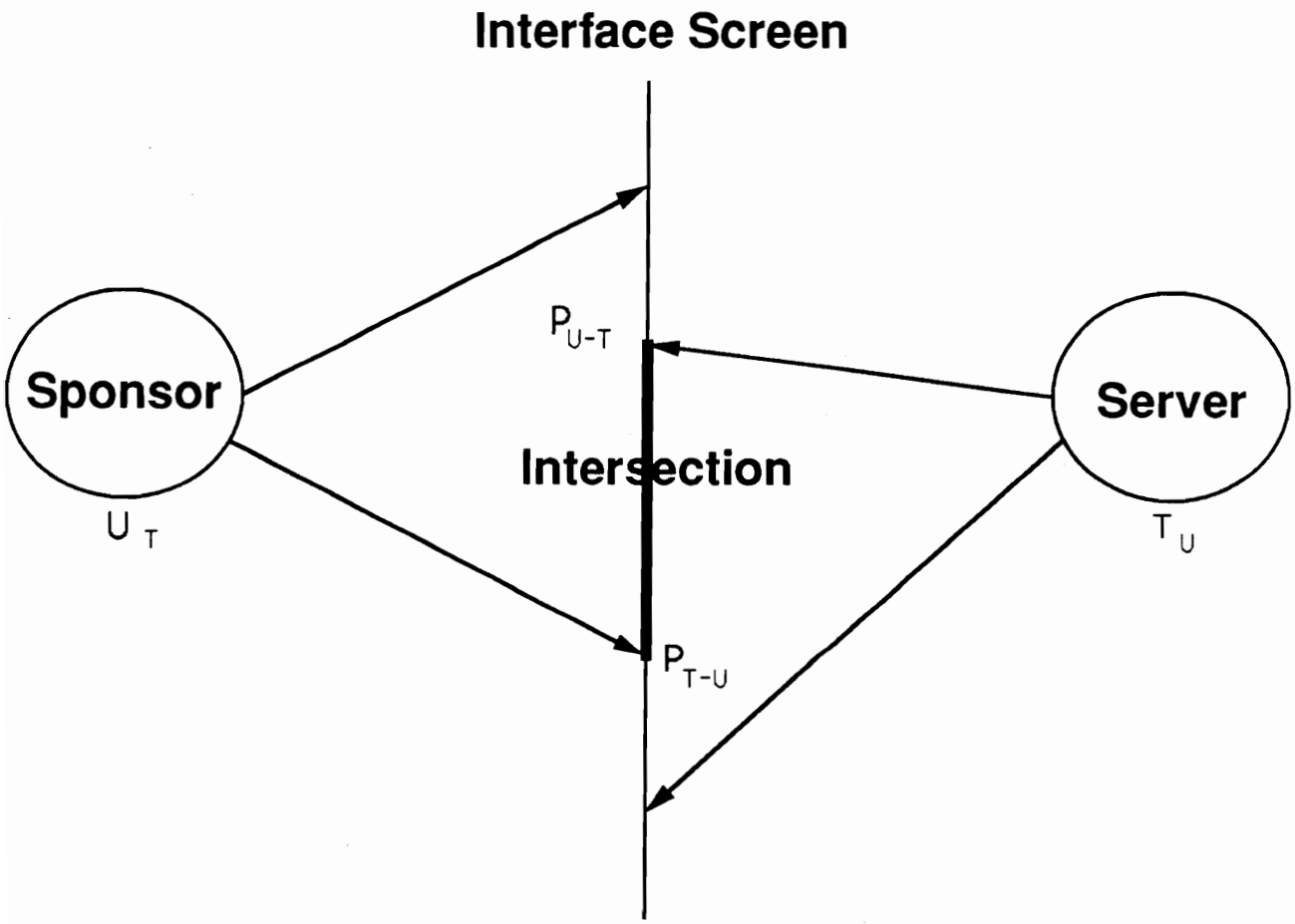


Figure 7. Each component projects its state to the interface screen.

between the boss' needs and the secretary's response. If the secretary attached no information or attached a pie chart, a mismatch occurs because it doesn't satisfy the boss's needs.

I'll put the model into the perspective of management tools. The manager has his or her characteristics, preferences, experiences, etc. The manager's characteristics determine the projection to one side of the screen. The management tool also has its design purpose, functionality, user interfaces, etc. The tool characteristics determine the projection to the other side of the screen. The central issue of this interface is information flow from the tool to the user. The flow of information through the interface is related to the size of the intersection area. No intersection area means no flow of information through the information portrayal/information perception interface in the MSM. That is, the tool is valueless if the manager doesn't use it or doesn't get help from it. The gap between projection areas may be defined by the unmatched areas and the difference between the two centroids. Perhaps we can project and affect the future usefulness of the tool by measuring and watching changes in the intersection area over time.

Figure 7 on page 64 implies a set of variables, in a broad sense, in terms of management tools:

- U_T[28] — user attributes,
- T_U — system attributes,
- P_{U-T} — U_T's projection on the interface,
- P_{T-U} — T_U's projection on the interface.

U_T: This represents variables that define the status of the user (sponsor). I categorize these variables into:

---

[28] If the MSM had but one interface, I would use U instead of U_T. When considering more than one interface, I distinguish the variables of the user for the perception interface from the variables of the user for the decision interface.

- **User characteristics** — attributes of a user considered relatively constant for a specific period of time. For example, age, gender, cognitive styles, skills, knowledge and beliefs, preferences, preunderstanding and background (Kammersgaard, 1988), physical attributes, etc.

- **Problem characteristics** — attributes of a problem the user is trying to solve and/or a task the user is trying to accomplish when using the system. For example, types of decision making tasks such as situation assessment, planning and commitment, and execution and monitoring (Rouse, 1984); structuredness (Simon, 1960), levels of managerial activity such as strategic planning, management control, and operational control (Anthony, 1965); classes of the decision making situation such as familiar and frequent, familiar and infrequent, and unfamiliar and infrequent (Rouse, 1984); etc.

- **Decision environment** — attributes of the environment where the user is trying to solve a problem. For example, the degree to which subunits of an organization are organic or mechanistic, the degree to which decision-making is centralized or decentralized, environments of an organization (Markus & Robey, 1983), etc.

- **User context** — attributes of a user subject to change during the use of the system. For example, goals, plans, intentions (Allen & Perrault, 1980; Nielsen, 1986; Kass & Finin, 1989); operative cognitions (Kammersgaard, 1988), misconceptions; etc.

These variables are not independent, but interrelated. As a whole, these variables define the state of the user.

T_U: This represents variables that define the state of the management tool. I categorize these variables into:

- **System parameters** — attributes of a system considered constant for a specific period of time. For example, problem solving methods or models (Remus & Kottemann, 1986), database management functions, interaction type, interaction style, interaction medium, etc.

- **System variables** — attributes of a system subject to change during the use of the system. For example, information content, information portrayal format, output frequency, level of help messages, etc.

The combination of these variables specifies a response of the system to the user.

$\mathbf{P_{U\text{-}T}}$: This variable represents the projection of U_T and can be explained as the *needs* of the user. We can view needs in terms of decision support or interaction support as reviewed in "2.2.1 Decision Support Perspective" on page 22 and "2.2.2 Interaction Support Perspective" on page 26. If the manager is making decisions on investments, he or she needs some information on investment alternatives and corresponding returns on investment. If the manager is a sensing person in terms of MBTI (Myers-Briggs Type Indicator)[29], he or she may need more detailed information. As such, P_{U-T} is multi-dimensional and is considered in terms of the user.

$\mathbf{P_{T\text{-}U}}$: This variable represents the projection of T_U and can be explained as the *responses* of the management tool. It can be specific information, a help message, an error message, etc. This is also multi-dimensional and is considered in terms of the tool.

Figure 8 on page 68 illustrates what can happen in the interface screen. If $P_{U-T} = P_{T-U}$, it means *perfect matching*. If $(P_{U-T} \cap P_{T-U}) = \emptyset$, it means *no matching*. If $P_{U-T} \supset P_{T-U}$, it means the system satisfies only a part of the user's expectation. If $P_{U-T} \subset P_{T-U}$, it means the system exceeds the user expectation or the user only expects parts of the tool's capability. This implies the idea of a management tool's potential or maturation. If the tool can do no more than the user expects, the user

[29] See Myers (1980).

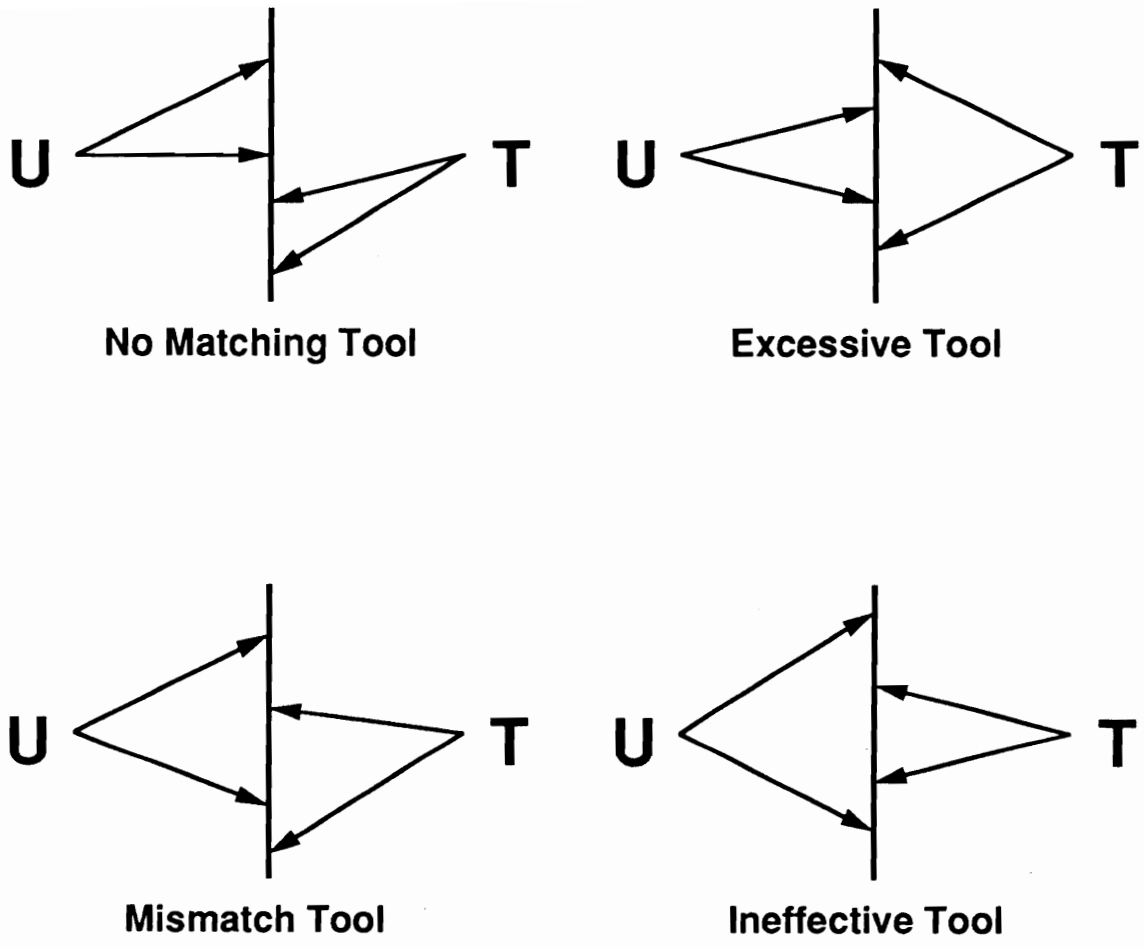


Figure 8. The interface screen exemplifies the matching between the manager and the tool.

will outgrow the tool. If the tool's projection is slightly larger than the user's projection, the user will improve his or her skill step by step without being embarrassed or being bored.

3.2.2 Dynamic View of the Model

Now let's consider time and assume more than one sponsor. Then the state of the sponsor changes depending on when and whom the server is serving. The change of the state of the sponsor causes its projection to change. The change in the projection again causes a mismatch in the interface. To put the mismatch into the perspective of the MSM, the state of the *who manages* component changes. We can think of two kinds of situations in which the individual user's projection area changes: 1) different users are using the tool, and 2) the same user is using the tool at different times. If the state of *who manages* changes, the resulting projection to the interface screen changes correspondingly in size or location. If the tool's projection doesn't evolve as the user's projection area changes, the two projection areas may no longer be matched. Then we need another matching process: changing the state of the *what is used to manage* component to match the changed projection of the *who manages* component. Usually system designers do the change process, which limits the capability of the tool to respond to changes in the user's projection (Figure 2 on page 34). If we transfer the role of the designer to the system, it will improve the system's capability to respond to the changes. Responsiveness, from a dynamic point of view, implies the system's automatic change in the system's projection (P_{T-U}), by changing its state (T_U) to match the change in the user's projection (P_{U-T}) caused by the change in the user's state (U_T).

3.2.3 Definition of Responsiveness in Management Tools

From the discussion of matching in the interface, I draw the following definition of responsiveness in management tools.

Given:

1. A user or user group **U**,
2. A management tool **S**,
3. A time period **T** — from t_0 to t_n, and
4. An application domain **A**,

I say:

S is ***responsive*** to **U** in **A** during **T**

when **S** recognizes the decision support and interaction support **needs** of the user working in **A**, which can vary among different users in **U** and evolve over **T**, and provides responses matching the needs.

Here **U**, **A**, and **T** define the notional world of the management tool **S**.[30] The definition implies a process: 1) observe the user's attributes, 2) reason the user's projection, 3) design the system's attributes that will produce the projection matching the user's projection, and 4) actually project to the interface screen. In the next section, I explain this process, which I call the *responsiveness process*, in more detail.

3.3 Responsiveness Process

In the previous section, I suggested that responsiveness, from the dynamic point view of the model, implies a process, which I call the *responsiveness process*. We can see responsiveness as a process. The responsive secretary listens, anticipates, and goes through some internal reasoning process,

---

[30] A notional world is a fictitious world in which a system would be perfectly adapted. For more detail, see "Environment" on page 38.

which the boss can't see. The only way the boss can see the process is through the result of the process — matching in the interface.

The responsiveness process consists of four major iterative stages: observing, understanding, interpreting, and implementing (Figure 9 on page 72). The first two stages are related to the sponsor side of the interface screen shown in Figure 7 on page 64 and the last two stages are related to the server side of the interface screen. In this section, I discuss each stage in more detail.

3.3.1 Observing

Responsiveness starts from observation. By becoming a better observer, the responsive secretary can draw conclusions that will enable him or her to anticipate certain actions (Belker, 1981). The responsive secretary observes the boss and the surrounding environment. From the observation, the secretary gets data about the boss. Observation is gathering data about the sponsor with all perception devices. The issue here is what to observe and how to observe. The secretary knows what is important to the boss and therefore knows what to observe about the boss. The secretary has observing devices such as eyes, ears, feeling, etc. and has observing management processes such as regular review of the boss' calendar.

In computer-based management tools, the primary way of observing is monitoring the user's interaction with the system. Monitored data can be used: in diagnosing and providing the user appropriate aid, in assessing user performance and user satisfaction, etc. (Penniman & Dominick, 1980). Other primitive forms of observing may be possible. For example, Starker & Bolt (1990) use eye-tracking to monitor the user looking at a graphics screen in a *gaze-responsive* information display system. Computer vision and speech recognition will extend the meaning of observation in terms of both what to observe and how to observe in computer-based management tools.

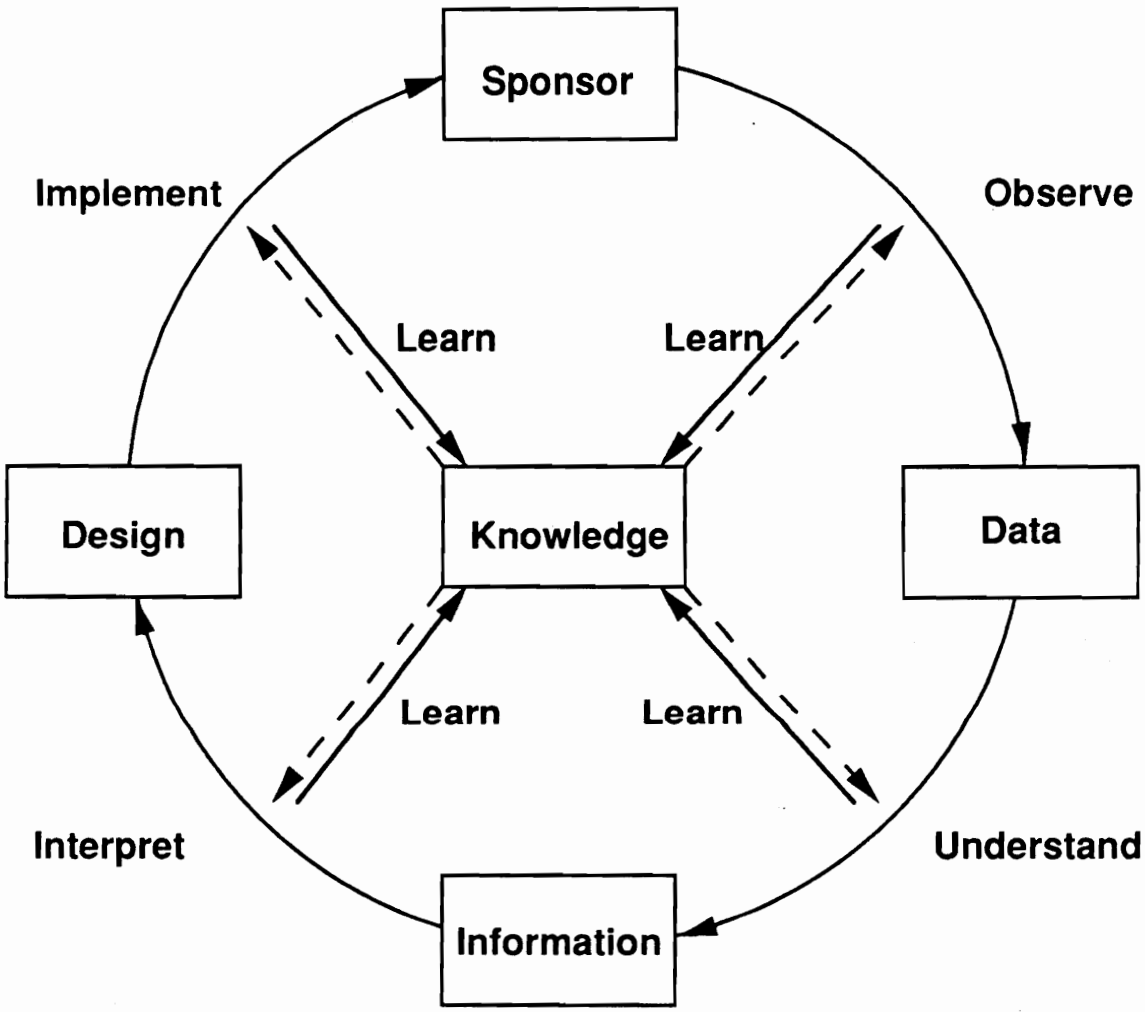


Figure 9. The responsiveness process is cyclic.

3.3.2 Understanding

The second stage is *understanding*. Having observed data and prior knowledge about the sponsor, the server reasons what the sponsor really wants or needs. In this stage, the data about the user are transformed to information about the user. The responsive secretary reasons that the boss needs information showing the trend of recent sales after reviewing the incoming letters and remembering the boss preferred information about the matter in the letters. In computer-based management tools, user modeling effort reviewed in "2.5 User Modeling Approach" on page 52 fits this stage. In terms of the model in Figure 7 on page 64, the ultimate goal of this stage is to figure out P_{U-T}. This implies a series of reasoning steps: from some variables in U_T to some other variables in U_T and then from those variables to P_{U-T}. For example, reasoning the goal of the user from the input of the user to the system and the context and then reasoning the needs of the user from the goal of the user and the characteristics of the user. Here all reasoning occurs in terms of the user (sponsor).

3.3.3 Interpreting

The third stage is *interpreting*. The server reasons what they can do to satisfy the sponsor's needs — the result of understanding. In this stage, a decision on what and how to respond is made based on information about the user. The responsive secretary determines to obtain a specific line chart showing the trend of recent sales. In computer-based management tools, this stage means the design of appropriate responses to the user's needs. In terms of the model in Figure 7 on page 64, the goal of this stage is to change T_U to match P_{U-T}. Gargan et al (1988) suggest an adaptable multi-modal response planner that dynamically designs the appropriate way to present information. Here all reasoning occurs in terms of the system (server).

3.3.4 Implementing

The fourth stage is *implementing*. The server provides the result of *interpreting* — a specific response. The responsive secretary prepares a line chart and attaches it to the letters. This stage implies capabilities of the server actually doing something, such as functionality and user interface capability, which I'll discuss in "3.4 Three Constructs of Responsiveness" on page 74. From the management perspective, if I say the first three stages correspond to information gathering and decision making, then this last stage corresponds to taking action. Without action, the decision is meaningless (Drucker, 1967).

3.3.5 Cyclic Feature of Responsiveness Process

These four stages constitute the responsiveness process. After implementing the designed response, the server observes the sponsor again to see the effects of its implementation. The responsiveness process is continuous and cyclic (Figure 9 on page 72). This cyclic feature of responsiveness implies learning about the sponsor. As the cycles go on, the server learns about the sponsor and uses that knowledge in later reasonings. This learning aspect is vital for responsiveness.

3.4 Three Constructs of Responsiveness

The responsiveness process explained in the previous section requires certain capabilities of the server. I categorize these capabilities into three constructs of responsiveness in computer-based management tools as shown in Figure 10 on page 75:

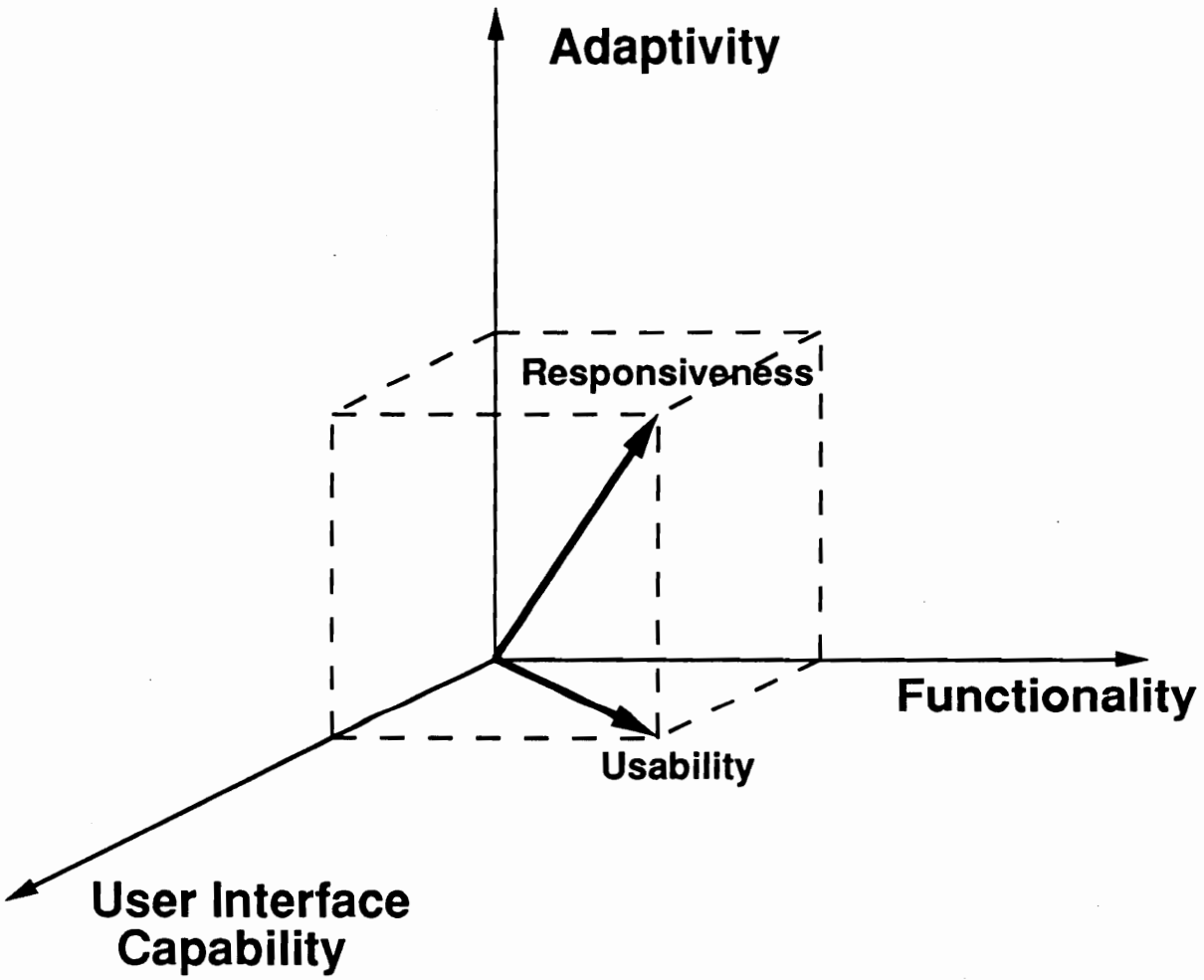


Figure 10. Three constructs constitute responsiveness in management tools.

- Functionality – capability to provide what the user is trying to get or accomplish,

- User Interface Capability – capability to enable the user to interact well with the system, and

- Adaptivity – abilities to adapt itself or to be adapted to the user.

All three constructs, as a whole, define *responsiveness* of a system. However, this doesn't imply the software of a computer-based system can be split into disjoint parts according to the constructs. The three constructs simply define three different perspectives. I explain each construct in more detail.

3.4.1 Functionality

When we deal with *functionality*, we are dealing with what can be done with a computer system in terms of the set of possible decision supports and the purposes for which the computer system is valuable for the user (Kammersgaard, 1988). Functionality corresponds to functional skills of secretaries or nurses such as typing, filing, treating a patient, etc. For example, issues related to *functionality* include:

- What kind of information is available?
- What kind of statistical or mathematical packages (or functions) are supported?
- What kind of models are supported?
- How big is the memory?
- How fast is the processing?

Functionality is basic to be responsive. The responsive secretary is assumed to be able to do what they design to satisfy the boss' needs. In the early years of computers, the most important issue was functionality. Users were at the mercy of the system. But, as we have enough basic

functionality because of the rapid development of computer technology, attention has shifted to the user. The user interface has become one of the hot issues for designing computer systems.

3.4.2 User Interface Capability

When we deal with *user interface capability*, we are dealing with how to make the user interact well with the system. Many people have suggested the importance of user interfaces. Powerful functionality by itself doesn't guarantee better performance of the user. User interface capability corresponds to communications or inter-personal skills of the responsive secretary or the responsive nurse. As I said in "2.2.2 Interaction Support Perspective" on page 26, people have differentiated user interface from the application program. For example, issues related to *user interface capability* include:

- What kinds of input or output devices are available?
- What kinds of presentation formats are supported, such as graphics, charts, tables, etc.?
- What types of interaction are supported, such as commands, fill-in forms, menus, direct manipulation, windows, natural language processing, etc.?
- What kinds of metacommunication are supported, such as help messages, error messages, manuals, etc.?

The ability to understand and speak natural languages is included in user interface capability. Many user interface techniques have been developed recently such as windows, pull-down menus, pointing devices, icons, etc. (Schneiderman, 1987).

3.4.3 Adaptivity

Powerful functionality and user interface capability alone don't yet guarantee good performance of a system because users are different and evolve over time. Here comes the issue of *adaptivity*. Adaptivity corresponds to anticipation skills of the responsive secretary (Belker, 1981) and intellectual skills of the responsive nurse (Murray, 1980). For example, issues related to *adaptivity* include:

- Can the user select one system mode among alternatives that fits him or her?
- Can the user customize the system ?
- Can the system adapt itself to the user ?

The adaptation perspective reviewed in "2.4.1 Adaptation Perspective" on page 38 fits here. The main focus of adaptivity is to match the user. We can think of several levels of adaptivity:[31]

1. **Custom-fit** systems are designed to match the user and any change in the user impairs the match. Here the matching responsibility is up to the system designers.

2. **Flexible** systems offer a series of fixed alternatives from which the user can choose. Here the matching responsibility is up to the system users.

3. **Adaptable** systems adjust to the user when and as they are told. Here the matching responsibility is up to the system users.

4. **Adaptive** systems adjust to the user without being told. Here the matching responsibility is up to the system itself.

---

[31] These terms are from Kurstedt (1985a) and Totterdell et al (1987). It is arguable that adaptivity includes the first three levels. The reason I put these in adaptivity is to focus on matching the user.

a. **Self-adaptive** systems adjust to the user without evaluating the expected outcome of the adjustment.

b. **Self-regulating** systems adjust to the user according to the evaluation of the expected outcome of the adjustment but without internalizing the learning.

c. **Self-mediating** systems adjust to the user with pre-evaluation of the various expected outcomes of their behavior.

Adaptivity is the key among the three constructs in this research and provides variables for the experiment evaluating a responsive system.

3.5 Conclusion

In this chapter, I presented a conceptual model of matching, the responsiveness process, and three constructs of responsiveness. These, as a whole, represent the conceptual model of this research, which explains the conceptual domain of this research and guides the rest of this research: reflecting and realizing the concept. The discussions in this chapter describe what responsiveness means in management tools. Generically those discussions evolve to the responsive system, a reflection of the concept of responsiveness, which I explain in the next chapter.

Chapter 4. Specification of the Responsive System

4.1 Overview

In Chapter 3, I defined the concept of responsiveness in management tools in terms of the conceptual model of matching and discussed the responsiveness process and three constructs of responsiveness (functionality, user interface capability, and adaptivity). Now the problem is how we can implement the concept into management tools. We need some technology to put the concept to work. Here I introduce the term *responsive system* of Kurstedt (1985a). The responsive system is a reflection of the concept of responsiveness, combined with some technology. Generically the responsive system is responsive. I define:

Responsive System a system that has a **layer** over the application program and the user interface that performs the responsiveness process to be responsive to the user.

In the definition, I emphasize the structural and the operational characteristics of the responsive system. When we talk about expert systems, we implicitly assume some structural characteristics of expert systems such as inference engines, knowledge bases, backtracking, etc as well as replication

of human experts (Harmon & King, 1985). In this chapter, I specify the responsive system operationally and structurally in more detail.

4.2 Operational Specification of the Responsive System

The core of the responsive system is in its responsiveness process and its adaptivity, which I explained in "3.3 Responsiveness Process" on page 70 and "3.4.3 Adaptivity" on page 78. These require certain operational characteristics of the responsive system. I explain these operational characteristics with a cube shown in Figure 11 on page 82. The cube was originally designed to distinguish intelligent systems (Lee et al, 1989). The cube consists of three dimensions: needs identification type, user model type, and system control type. There are 27 cells in the cube and the responsive system is in the (implicit, individual, feedback control) cell. I explain each dimension, relating it with the responsive system.

4.2.1 Implicit Needs Identification

The needs identification type dimension considers how the system identifies the user's needs or wishes to the system — P_{U_T} in Figure 7 on page 64. This dimension is divided into three ranges:

Predefined The system doesn't allow any other needs except the needs we've defined in the design of the system. During the design phase we define all the needs the system can satisfy. That is, the needs should be well-definable. The system can't identify needs we haven't defined. Conventional expert systems usually satisfy this kind of needs. They produce advice by following a decision tree guided by a user's answers to a set of pre-defined questions.

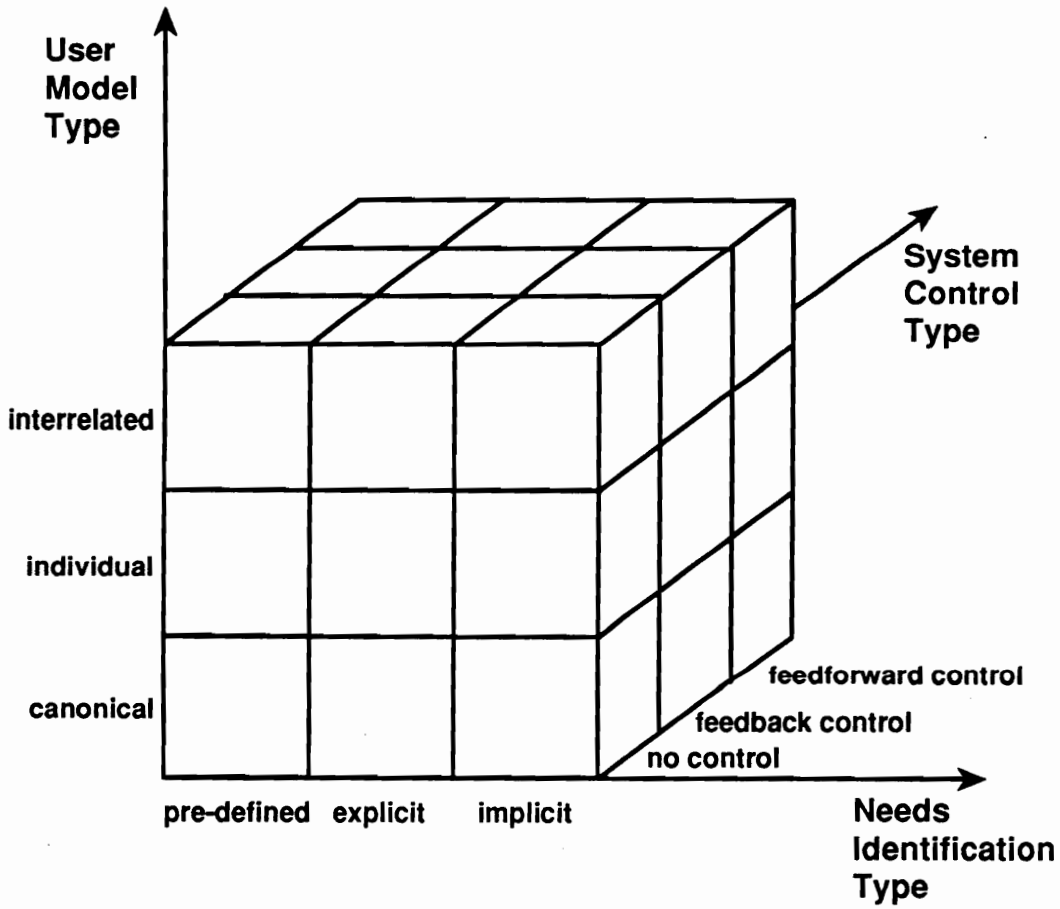


Figure 11. A cube distinguishes intelligent systems.

Explicit The system identifies needs the user specifies. The user can describe their individual needs in their own terms. The system interprets what the user described and identifies the user's needs. Often the user's needs can't be completely defined until they use the system. In this case, the system allows the user to express their needs during the use of the system and then works based on those needs. Volunteering information (Fischer & Stevens, 1987), True Consultants (Shulok, 1988), adaptable, and flexible systems can be placed in this range.

Implicit The system identifies the user's needs, whether they specify their needs or not. The system observes the user and the context, interprets what they mean, and identifies the needs. Often, even the user doesn't notice there is a problem. The needs are extracted by the system.

The responsive system identifies the user's needs implicitly by itself and tries to help the user in terms of the identified needs. For example, the responsive system recognizes the goal or intent of the user by observing the user's inputs to and corresponding context of the system. Recognition of the goal or intent leads to identification of the user's needs.

4.2.2 Individual User Model

The user model type dimension considers how the system views the user. I mean by user model the system's model of the user: a set of rules and/or knowledge about the user the system follows to determine its reaction to the user. This dimension is divided into three ranges[32]:

Canonical The system maintains one identical user model representing all users. That is, the system treats all users exactly alike. It can't notice individual differences of users.

[32] I borrow the first two from Rich (1983).

We might accommodate individual differences in the design phase. Conventional expert systems usually maintain this type of user model.

Individual The system maintains a collection of different user models representing each individual user. The system identifies individual users and treats them differently. The system can't notice relationships between individuals.

Interrelated The system maintains individual user models and their relationships. The system identifies a group among individual users and the interrelatedness of the group. Group decision support systems (Watson et al, 1988) and consensus-gathering systems (Lee et al, 1989) may maintain interrelated user models.

The responsive system maintains individual user models. It recognizes who is using the system. It keeps information or knowledge about individual users separately and retrieves this information when the user starts to use the system.

4.2.3 Feedback System Control

This dimension considers how the system changes its state to adapt to the user. I've divided the system control type dimension into three ranges:

No control The system doesn't change its state. The system can be flexible and provide options the user can select, but the system is fixed. Conventional expert systems have no control strategy.

Feedback control The system changes its state based on the current state of the user. The system observes the current state of the user, and adapts to that. But the system

doesn't predict the user's future state. The adaptation strategy is based on the current state of the user.

Feedforward control The system changes its state based on the future state of the user. The system predicts the future state of the user, and adapts to that. The adaptation strategy is based on the user's future state. The system should have a prediction model.

The responsive system does feedback control. In Figure 7 on page 64, if there occurs some gap between two projections, the system observes the gap and tries to reduce the gap by adjusting the system attributes (T_U). Yet the responsive system doesn't anticipate the user's future state. We can think of a higher level of intelligent systems — anticipatory systems (Rosen, 1985).

4.3 Structural Specification of the Responsive System

A structure of a system should reflect what the system is going to do.[33] I design a structure of the responsive system with existing artificial intelligence technology, especially the blackboard architecture.

4.3.1 The Conceptual Architecture of a Responsive System

Conceptually the responsive system consists of three parts as shown in Figure 12 on page 86:

[33] By *structure* I mean both components and their relationships.

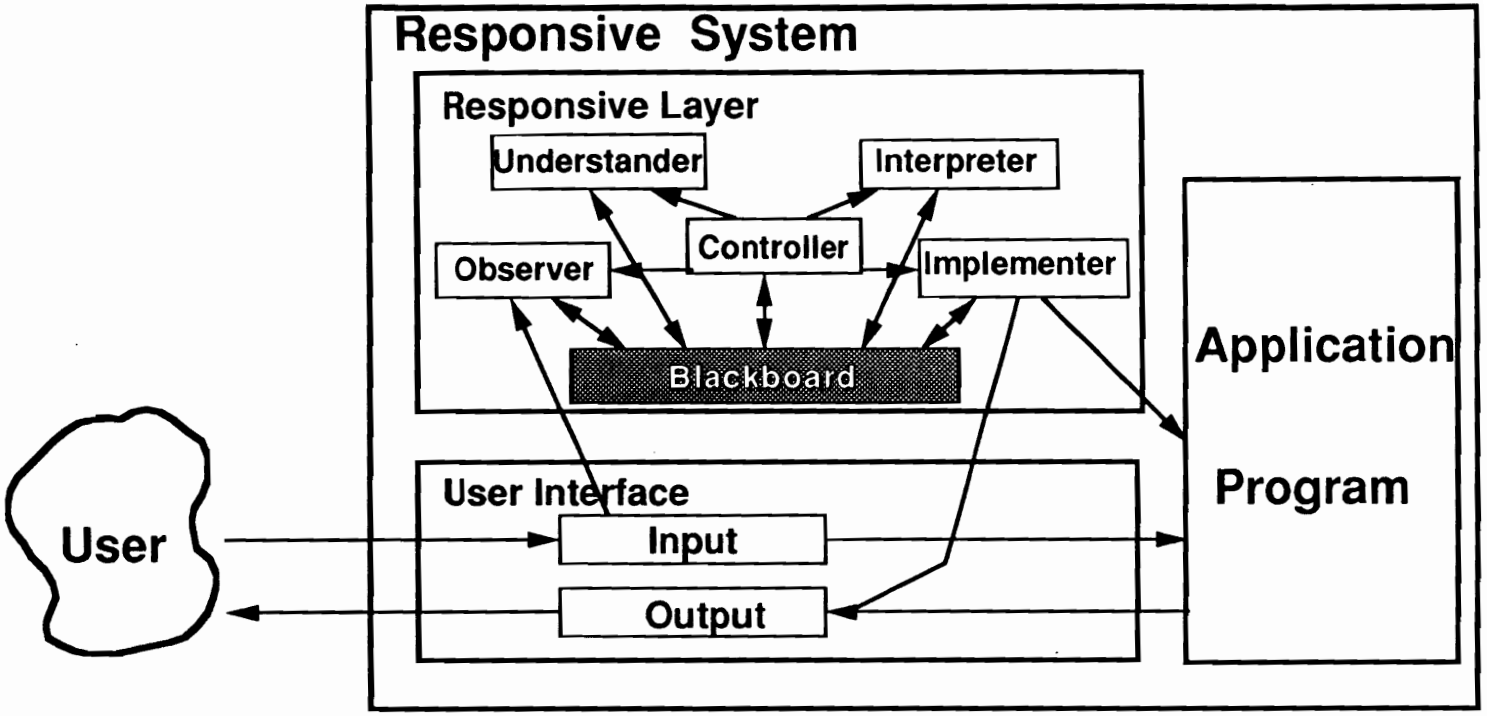


Figure 12. Conceptually a responsive system consists of three parts.

- Application program — main part of the system that prepares outputs the user is trying to get or accomplish;

- User interface — communication part of the system that receives the user's inputs and presents the system's outputs to the user; and

- Responsive layer — layer over the application program and the user interface that performs the responsiveness process.

Each part corresponds to a construct of responsiveness shown in Figure 10 on page 75; the application program to functionality, the user interface to user interface capability, and the responsive layer to adaptivity. The key component of the architecture is the responsive layer. The application program and the user interface may provide highly sophisticated options from which the responsive layer can select. The responsive layer essentially overlays the application program and the user interface and upgrades the system's responsiveness in both decision support and interaction support. For example, if the application program has options and is therefore flexible, the responsive layer adds the ability to observe, understand, and interpret to choose the option and makes the application program look adaptive — it's not really adaptive, it's still limited by the number of options. Therefore, the degree of responsiveness of a responsive system depends on the adaptivity (e.g. maturation of knowledge) of the responsive layer, the functionality of the application program, and the user interface capability of the user interface.

4.3.2 The Responsive Layer

In the design of a structure of the responsive system, the main issue is the design of the responsive layer. I apply the blackboard architecture to the design of the responsive layer. Several researchers support the blackboard architecture for modeling the user and designing responses to the user (Belkin et al, 1984; Smith, 1985; Alty & McKell, 1986; McCalla et al, 1986; Balzert, 1987; Rubin

et al, 1987). In this section, I first review the blackboard architecture and then propose a structure of the responsive layer and how it relates to the application program and the user interface.

4.3.2.1 Blackboard Architecture

The blackboard architecture originates from the HEARSAY-II speech understanding system developed between 1971 and 1976 (Erman et al, 1980). The blackboard architecture has been used in several areas such as speech recognition (Erman et al, 1980), errand planning (Hayes-Roth & Hayes-Roth,1979), cooperative distributed systems (Lesser & Corkill, 1981), cockpit information management (Baum et al, 1988), self-improving instructional planner (Macmillan & Sleeman, 1987), etc. The blackboard architecture is a problem-solving framework based on the distributed problem-solving paradigm, which means the cooperative solution of problems by a collection of problem-solving experts when one expert doesn't have enough knowledge to solve the problem. Distributed problem solving offers several advantages — "speed, reliability, extensibility, the ability to work with uncertain data and knowledge, etc — especially in highly complex or uncertain problem domains" (Smith & Davis, 1981).

The blackboard architecture usually consists of three major components: knowledge sources, blackboard data structure, and a control mechanism (Nii, 1986).

Knowledge Sources: Knowledge sources (KS's) are problem-solving experts working together to solve a problem. Each knowledge source has necessary but not sufficient problem-solving knowledge. However, by working together, knowledge sources exercise full knowledge of problem solving. The objective of each knowledge source is to contribute information that will lead to a solution to the problem (Nii, 1986). A knowledge source generates, combines, and/or evaluates hypothetical interpretation about the problem (Erman et al, 1980) and is represented as procedures, sets of rules, or logical assertions (Nii, 1986). Knowledge sources are kept separate and independent. They communicate through a blackboard by posting and reading messages on it.

Blackboard Data Structure: A blackboard is a structured, globally accessible database containing intermediate or partial results of problem solving that are needed by and produced by the knowledge sources. That is, the blackboard consists of objects from the solution space, which can be input data, partial solutions, alternatives, and final solutions (Nii, 1986). Typically, the blackboard is partitioned for representing hypotheses at different levels of abstraction and mediates the cooperative activities of multiple knowledge sources (Hayes-Roth et al, 1983). Knowledge sources interact with each other only through the blackboard.

Control Mechanism: The control mechanism specifies a general problem-solving behavior. In the blackboard architecture, there is a set of control modules that monitor the changes on the blackboard and decide what actions to take next (Nii, 1986). The control modules are responsible for determining which knowledge source should act at each point in the problem-solving process. The modules use various kinds of information globally available. There exist several kinds of control mechanisms in the blackboard architecture, such as in Erman et al (1980), Hayes-Roth (1985).

4.3.2.2 The Structure of the Responsive Layer

The responsive layer of a responsive system tries to solve a problem: what the system should do to match the user's needs. The problem is very complex and uncertain and needs lots of different kinds of knowledge, suggesting some levels of abstraction such as results of each stage in the responsiveness process. Therefore, I believe the blackboard architecture is appropriate for the responsive layer.

The responsive layer has four groups of knowledge sources that have knowledge required to perform the responsiveness process: observer, understander, interpreter, and implementer as shown in Figure 12 on page 86. Actual knowledge sources will vary in implementation according to the application domain. Each knowledge source is an expert on one of the stages in the responsiveness process. A knowledge source posts its messages (e.g., solution elements such as observed user var-

iables, what the user needs, what the system should do) on the blackboard and reads messages the other knowledges sources posted to continue their own processing.

The responsive layer maintains a blackboard. The blackboard consists of several hierarchical levels, which basically correspond to the result of each stage in the responsiveness process (Figure 13 on page 91). Knowledge sources read messages in a level in the blackboard and post messages to a higher or at the same level of the blackboard. The design of the levels will vary according to the application.

In the responsive layer, a scheduler controls triggering and execution of knowledge sources. The scheduler maintains all information required for control of knowledge sources. The scheduler starts to work when there is some triggering condition in the user interface or in the application program such as the user's input to the system. Then the scheduler goes through the following iterative sequence until it gets some design that matches the user's needs or it recognizes it doesn't have enough knowledge to make the match.

1. A knowledge source makes change(s) to blackboard object(s). As these changes are made, the scheduler keeps an event record in a data structure holding the control information.

2. The scheduler identifies which knowledge sources have potential to contribute to the new solution state by checking the trigger condition[34] of each knowledge source with the event generated in Point 1. The scheduler triggers these knowledge sources as KSAR's (Knowledge Source Activation Records) that contain the solution context at that time.[35] These are called triggered KSAR's (Hayes-Roth, 1985).

---

[34] Trigger conditions specify which events a knowledge source can be triggered on.

[35] A KS can be triggered as several KSAR's containing different contexts.

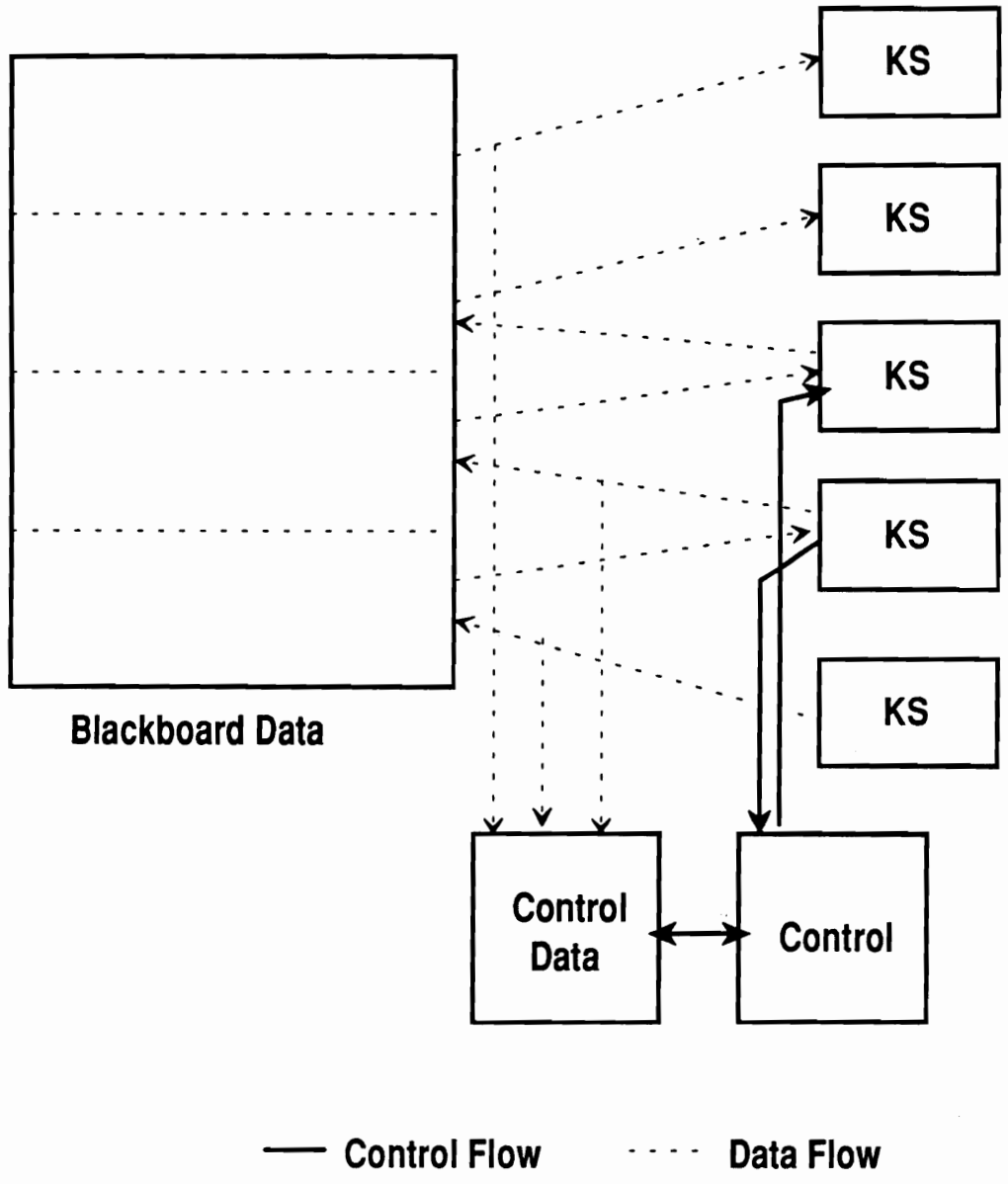


Figure 13. The responsive layer employs the blackboard architecture. (Source: Nii, 1986)

3. The scheduler identifies which triggered KSAR's can contribute to the new solution state by checking the precondition[36] of each KSAR with the solution state at that time. These KSAR's are called invokable KSAR's (Hayes-Roth, 1985).

4. The scheduler selects one KSAR among the invokable KSAR's with some given criteria.

5. The scheduler executes the selected KSAR. At this time, the knowledge source associated with the KSAR contributes to the new solution state by posting messages to the blackboard.

4.4 Conclusion

In this chapter, I specified the responsive system in terms its operational characteristics and its structure. The responsive system identifies the user's needs implicitly, maintains individual user models, and does feedback control. The responsive system mainly consists of three parts: the application program, the user interface, and the responsive layer. The responsive layer employs the blackboard architecture. These specifications help understand how we can implement the concept of responsiveness into management tools. However, still it is questionable whether we can implement the concept into a real management tool. For this reason, I developed a prototype responsive management tool, which I explain in the next chapter.

---

[36] Preconditions specify under which solution state a knowledge source can be executed.

Chapter 5. MSLTRAIN: A Prototype Responsive Management Tool

To realize the structure of a responsive system explained in Chapter 4, I developed a prototype responsive management tool called MSLTRAIN. In this chapter, I overview MSLTRAIN in terms of its functionality and user interface capability explained in "3.4 Three Constructs of Responsiveness" on page 74 and explain it in terms of its responsiveness and detailed structure.

5.1 Overview of MSLTRAIN

5.1.1 Application Domain

The application domain of MSLTRAIN is an area of management — identifying and scheduling training needs on computer packages in an organization. At both management and staff levels, the personnel in an organization are changing. The hardware, and especially the software, feature new

capabilities everyday — faster than managers can keep up. Without proper and timely training, the sophisticated hardware and software can't be properly used; and consequently the office will bog down.

However, identifying and scheduling training needs on computer packages is not simple. For example, often managers select wrong computer packages not helping their decision making, train more people that the organization needs, schedule people when they can't attend training sessions, etc. In this respect, Kurstedt (1986) developed a five-step instrument that helps managers with their training by addressing the organization's information needs and then helps identify which people should be trained on which hardware and software at what times. The worksheet-type instrument provides a structured approach for professional development. It helps managers organize training by considering first what the organization needs and second which people should be trained on which packages at what times. I transformed the worksheet into a computer program called MSLTRAIN. The purpose of computerization was 1) to reduce the burden on the user to manipulate the tables in the worksheet, and 2) to help the user access information more efficiently and effectively.

In MSLTRAIN, the manager goes through the following five decision-making steps to organize the training of people in their organization.

Step 1: Determine generic organizational needs and then the computer packages that support those needs. MSLTRAIN provides a list of computer packages the organization may need and the description of each package.[37] The manager selects packages he or she needs from the list.

Step 2: Set up milestone dates by which the training needs must be met, by month. These dates are used for determining the organizational proficiency in Step 3 as reference time points.

---

[37] In the worksheet, there is a procedure that delimits the domain of responsibility, identifies decisions the manager makes, identifies information the manager needs for those decisions, and chooses appropriate computer packages to provide the information. This procedure is not computerized yet.

Step 3: Determine the organizational proficiency[38]: how many people in the organization need to have novice, regular, or expert level of proficiency on each needed package by each milestone date. The manager should consider training commitment needed to satisfy organizational proficiency. If the manager wants a number of people proficient on a package, they should expect people will be out of the office for a number of days. MSLTRAIN automatically computes the training commitment in days according to the organizational proficiency the manager wants. The manager can change the organizational proficiency after seeing the training commitment information.

Step 4: Determine individual proficiencies: which level of proficiency on each needed package is desired for each person in the organization. In this step, the manager switches their perspective from the organization to the person. The manager matches their needs with their resources (people). The manager considers background, capability, job description, etc of each person in the organization and sets their desired proficiency levels on the packages to meet the desired organizational proficiency. The manager can prevent assigning more people than the organization needs by comparing individual proficiencies with the organizational proficiency determined in Step 3. The shortfall of people to meet the organizational proficiency surfaces the issue of whether they have the right people in their organization. MSLTRAIN automatically computes the gap (shortfall or overage) the manager has in matching their resources to their needs. After seeing the gap information, the manager can revise the individual proficiencies or go back to Step 3 and revise the organizational proficiency.

Step 5: Set up the training schedule: when (in which month) each person in the organization will be trained to the desired level on each needed package. The manager schedules their people based on decisions in Step 3 and Step 4 and the schedules of their people. MSLTRAIN automatically provides the training courses each person should take to satisfy the desired individual proficiencies and computes whether the schedules satisfy the organizational proficiency set up in Step 3.

---

[38] By organizational proficiency I mean a number of people who together have various levels of proficiency on the several packages.

Through successive decisions, the manager finally gets a training schedule for their people. To help the manager's decision making, MSLTRAIN maintains information the manager can access, which is shown in "Appendix A. Information Items in MSLTRAIN" on page 175. Some information can be displayed in two portrayal formats — table or graph. The manager can access information in the format they prefer. Some information can be viewed in two scopes — whole or part. In the *whole* scope, the manager can see information on all packages or people in one form ("Whole"). In the *part* scope, the manager can see only part of the information on some package ("By Package") or some person ("By Person"). For example, the manager can see organizational proficiency in the whole scope showing proficiency on all packages or in the part scope showing proficiency on a specific package.

5.1.2 User Interfaces in MSLTRAIN

A typical screen of MSLTRAIN is shown in Figure 14 on page 97. The screen is divided into five panes:

- Guide pane — displaying some guide on what the user can do,
- Status display pane — displaying what the user is doing,
- Input pane — placing menus and decision windows,
- Information display pane — displaying information, and
- Information menu pane — displaying information items the user can access.

The guide pane can be placed at the top or at the bottom of the screen (Figure 20 on page 200). The user can change the position of the guide pane by hitting the "CHANGE POS" key. The information display pane can be hidden. If the user clicks the mouse on "This Menu Off" in the information menu pane, the pane disappears. If the user hits the "INFO" key, the pane appears again.

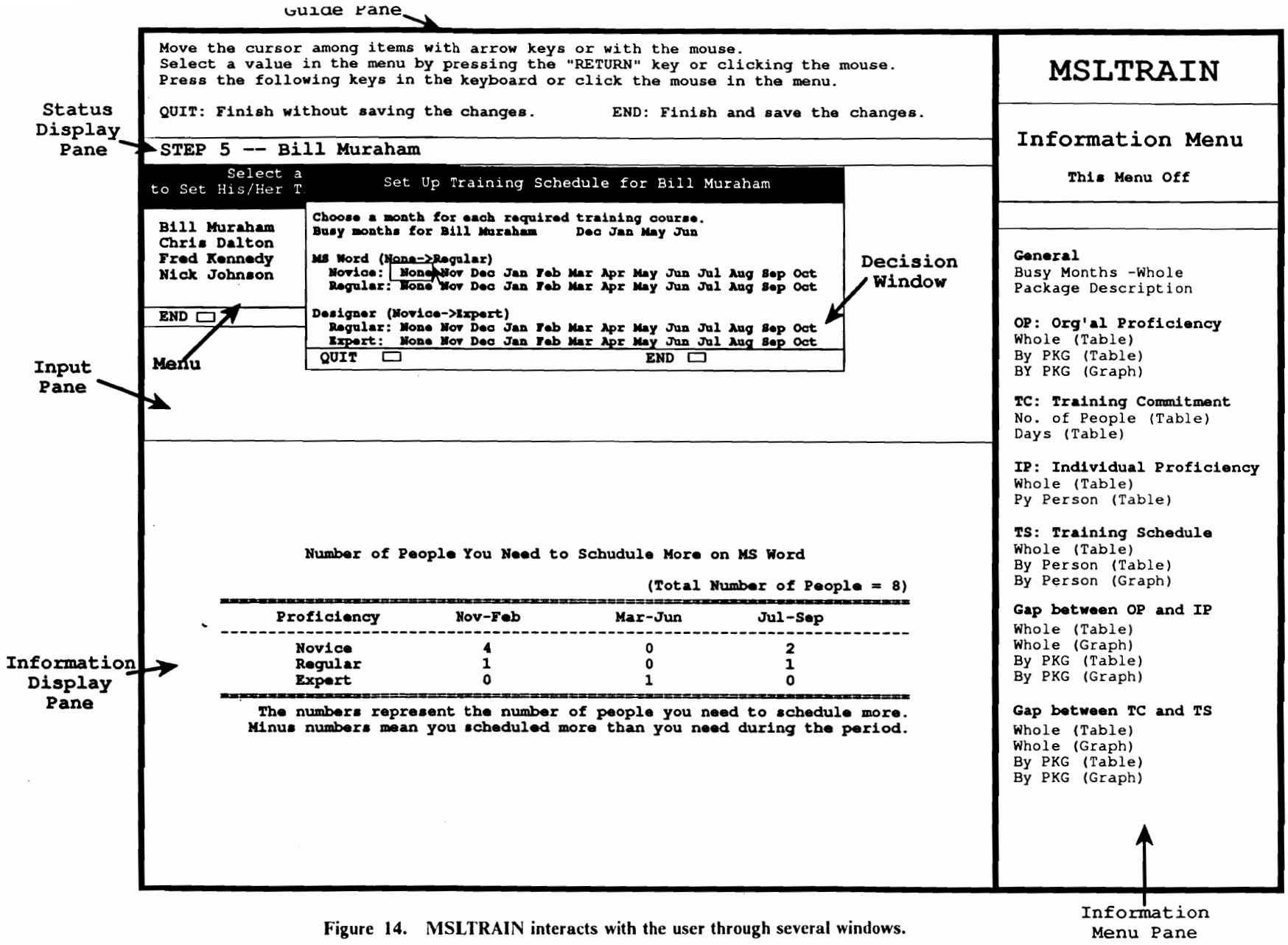


Figure 14. MSLTRAIN interacts with the user through several windows.

There are also several temporary menus and decision windows, which are shown as pop-up windows. The menus list the names of packages or people, from which the user can select. The decision windows provide a mechanism with which the user can input their decisions in each step. "Appendix B. MSLTRAIN Screens in Each Step" on page 198 shows screens of MSLTRAIN in each step.

The user can use the keyboard or the mouse as an input device. The user can select an item in the menus, in the decision windows, or in the information menu pane by hitting the "RETURN" key or clicking the mouse on the item.

MSLTRAIN has three interaction modes:

- Full guide — guide the user through all steps and provide an explanation of each step in the information display window and guidance on what to do in the guide pane (Figure 19 on page 199).

- Partial guide — don't guide the user through all steps but provide guidance on what to do in the guide pane.

- No guide — don't provide any guidance on what to do.

MSLTRAIN automatically sets the interaction mode to one of the modes appropriate to the user. However, the user can change the interaction mode by hitting the "CHANGE MODE" key.

5.1.3 Hardware and Software Environment

I developed MSLTRAIN on a Texas Instruments (TI) Explorer computer, a single-user LISP workstation designed to provide support for development of programs that require symbolic proc-

essing. This computer system consists of the following major components: a large cabinet holding the cpu with 2 megabytes of core memory; two hard disks with a capacity of 112 megabytes; a tape drive with a tape having 60 megabytes; a monochrome monitor, 17-inch diagonal tube, consisting of 1024 pixels horizontally and 808 pixels vertically; a keyboard containing over 100 keys, including the standard QWERTY keys, LISP-specific keys, a number pad, mouse-control keys, and cursor-control keys; and a three-button mouse.

The computer language used for MSLTRAIN implementation is Common LISP based on Steele (1984), which the TI Explorer supports. The TI Explorer provides an additional set of programming tools to support programmers. These include the relational table management system (RTMS); built-in user interface libraries such as windows, menus, graphics; menu-based natural language processing (NLMenu); etc. All these are supported in the object-oriented programming paradigm. Object-oriented programming is a technique relying on objects and messages instead of procedures and data in realizing a given piece of software (Goldberg & Robson, 1983; Cox, 1986; Stefik & Bobrow, 1986). TI Explorer provides the usual features of object-oriented programming, including the ability to create flavors (classes and subclasses) and instances, to define methods associated with the flavors, and to apply the inheritance mechanism. To develop MSLTRAIN, I used user interface libraries and graphics in the object-oriented programming paradigm.

5.2 Responsiveness in MSLTRAIN

In this section, I describe MSLTRAIN in terms of its adaptivity, the final one of the three constructs of responsiveness explained in "3.4 Three Constructs of Responsiveness" on page 74. MSLTRAIN has adaptive features and incorporates the responsiveness process explained in "3.3 Responsiveness Process" on page 70. I explain the adaptive features and the responsiveness process of MSLTRAIN, avoiding technological terms and showing examples.

Note: The adaptive features, in terms of adaptive responses, focus on Step 5 of MSLTRAIN. The purpose of the prototype was to demonstrate we can implement the concept of responsiveness in a management tool. I implemented the concept of responsiveness focusing on Step 5 of MSLTRAIN because I thought that was enough to accomplish the purpose. Further implementation of the concept on the other steps can follow this study.

Let's suppose the following situation.

Today is Oct. 15, 1990. A manager, John Miller, of an organization comprising eight people recently realized his people need more knowledge of computer packages. He is using MSLTRAIN to identify and schedule his training needs. In Step 1, he determined his organization would need three packages: "MS Word," "Lotus 123," and "Designer." In Step 2, he determined February, June, and October as milestone time points. That is, he wants the first milestone, in terms of the number of people proficient on the packages, by February 1991, the second milestone by June 1991, and the final milestone by October 1991. In Step 3, he sets the organizational proficiency. By the way, MSLTRAIN considers three levels of proficiency on each package.

- A "Novice" has little knowledge of the package and uses it occasionally, such as three times or less a week.

- A "Regular" has relatively much knowledge of the package and uses it everyday.

- An "Expert" has knowledge of the package enough to advise other people who have problems on the package.

A person who knows nothing about the package is referred to "None." Currently John Miller's organization has one novice and one regular on "MS Word," two novices on "Lotus 123."

After considering all situations, John Miller decided his organization should have four novices, two regulars, and one expert on each package by the end of October 1991. To get this goal, he set

milestones in terms of the organizational proficiency, like those in the first table in Table 1 on page 102. For example, he wants his organization to have three novices and three regulars on "Lotus 123" by the end of June 1991. One person can have different proficiencies on different packages. The organizational proficiency implies training commitment. The second table in Table 1 on page 102 shows how many people should be trained during each period to satisfy the milestones. For example, he should schedule two novices and one regular on "MS Word" during the period from July 1991 to October 1991.

In Step 4, John Miller changed his perspective to the individuals in his organization. He set the proficiency levels he desired each individual to have by the end of October 1991, like those in the third table in Table 1 on page 102. For example, he wants "Bill Muraham" to be at the regular proficiency level on "MS Word" and at the expert proficiency level on "Designer" by the end of October 1991. "Bill Muraham" is currently a novice on "Designer." Therefore, "Bill Muraham" needs four training sessions: a novice and a regular session on "MS Word" and a regular and an expert session on "Designer."

Now he is going to do Step 5: to determine when each individual in his organization will get the training sessions they need during the following twelve months (from Nov. 1990 to Oct. 1991).

I'll use this example throughout this section. In the example, MSLTRAIN deals with packages ("MS Word," "Lotus 123," and "Designer"), people (eight people in the organization), proficiency levels ("Novice," "Regular," and "Expert"), milestones ("By Feb," "By Jun," and "By Oct"), months (from "Nov" 1990 to "Oct" 1991), information items ("General," "OP: Organizational Proficiency," "TC: Training Commitment," "IP: Individual Proficiencies," "TS: Training Schedules," "Gap between OP and IP," and "Gap between TC and TS"), portrayal formats ("Table" or "Graph"), and information scopes ("Whole," "By Person," or "By Package").

Table 1. The manager made decisions in Step 3 and Step 4.

Desired Organizational Proficiency

(Total Number of People = 8)

Package	By Feb			By Jun			By Oct		
	N	R	E	N	R	E	N	R	E
MS Word	3	2	0	3	1	1	4	2	1
Lotus 123	2	2	0	3	3	0	4	2	1
Designer	3	1	0	4	0	1	4	2	1

N: novice R: regular E: expert

Required Number of People to Be Trained

(Total Number of People = 8)

Package	Nov - Feb			May - Jun			Jul - Oct		
	N	R	E	N	R	E	N	R	E
MS Word	3	1	0	0	0	1	2	1	0
Lotus 123	2	2	0	2	1	0	1	0	1
Designer	3	1	0	1	0	1	2	2	1

N: novice R: regular E: expert

Desired Individual Proficiencies

Name	MS Word	Lotus 123	Designer
Bill Muraham	* → R	*	N → E
Bob Hamilton	* → N	* → N	* → N
Chris Dalton	* → N	* → N	* → R
Dana Keating	*	* → N	* → N
Fred Kennedy	* → N	N → E	* → N
Jane Newmann	N → R	* → R	*
Nick Johnson	* → N	N → R	* → N
Paul Moriarty	R → E	* → N	* → R

*: none N: novice R: regular E: expert

The arrows (→) indicate upgrade.

5.2.1 Adaptive Features

MSLTRAIN adapts to the manager largely in two ways: 1) initial tuning of the system before the main session and 2) providing adaptive responses to the manager during the session.

Initial Tuning: By initial tuning I mean setting the default values of system parameters explained in "3.2.1 Static View of the Model" on page 63 to the values appropriate to the user before actually beginning the session.

1. MSLTRAIN has three interaction modes explained in "5.1.2 User Interfaces in MSLTRAIN" on page 96: full guide, partial guide, and no guide. Experts on MSLTRAIN don't need any guide. They may not like the system's guide. However, novices on MSLTRAIN may prefer the system's guide. Based on the proficiency level of the user, MSLTRAIN initially sets the interaction mode for that user to one of the interaction modes.

2. The guide pane can be placed at the top or at the bottom of the screen. Some people prefer the guide pane at the top; for example, those who are familiar with full-down menus. Some people prefer the guide pane at the bottom; for example, those who are familiar with Microsoft Word that shows menus at the bottom. Based on the user's preference on the guide pane position and input devices, MSLTRAIN initially sets the guide pane position at the top or at the bottom of the screen.

3. The information menu can be shown or hidden. If the information menu is hidden and the user hits the "INFO" key to access information, the information will be shown. The existence of the information menu pane may confuse novices on the system because they may not know what it is. According to the user's preference and proficiency level, MSLTRAIN initially sets this value on or off.

4. MSLTRAIN can show an error message or just beep when the user makes an error. Experts on MSLTRAIN may not like error messages when they accidentally make errors. According to the user's proficiency, MSLTRAIN initially sets this value on or off.

Adaptive Response: During the session, MSLTRAIN provides the user with information they may need for the decisions in each step even without their request. Furthermore, MSLTRAIN shows information in the scope (whole or part) and in the format (table or graph) MSLTRAIN judges the user will prefer.

In the example situation, the manager is doing Step 5. Now suppose the manager selected "Bill Muraham" in the menu (Figure 27 on page 263). The manager wants to schedule training for "Bill Muraham" and needs some information to decide the month for the sessions "Bill Muraham" should take. At this point, MSLTRAIN reasons first which information the manager needs. If the manager wants to schedule the person for the first time, he needs information showing which month is available for the person to get training, according to the training commitment needed in that month. MSLTRAIN decides to provide "Gap between Training Commitment (TC) and Training Schedule (TS)," which indicates how many more people the manager should schedule during each milestone period. Then MSLTRAIN reasons which information scope (whole or by package) is proper for the manager. If the manager is focusing on a specific package, he doesn't need information for all packages. MSLTRAIN reasons the manager is focusing on the first package in the decision window in Figure 28 on page 264 and decides to provide only the "MS Word" part of the information. Then MSLTRAIN reasons which format (table or graph) is appropriate for the manager. MSLTRAIN checks the manager's preference on the format and decides "Table" is appropriate for the manager. Finally MSLTRAIN displays the information, "Gap between TC and TS; By Package (Table)"[39] of "MS Word." There can be the following further examples:

---

[39] To see what this table looks like, see "Appendix A. Information Items in MSLTRAIN" on page 175.

- After the manager schedules all sessions on "MS Word," MSLTRAIN decides the manager will switch his focus from "MS Word" to the next package. MSLTRAIN displays the information on the next package, "Designer."

- If the manager moves the cursor to the row of "Designer," MSLTRAIN knows the manager is switching his focus from "MS Word" to "Designer." MSLTRAIN displays the information on "Designer."

- After the manager schedules all sessions for "Bill Muraham," MSLTRAIN decides the manager wants to see the schedule consequences, such as the person-days he should expect to be required for training during each month. MSLTRAIN displays "TS: Training Schedule; By Person (Table)" of "Bill Muraham."

- If the manager wants information, he hits the "INFO" key. Then the cursor moves to the information menu pane. At this point, MSLTRAIN decides what information the manager may want to see and places the cursor on that information so the manager doesn't have to hit extra keystrokes.

In the example, suppose the manager already scheduled "Bill Muraham" and selected "Bill Muraham" again. In this case, the manager wants to change the training schedule of "Bill Muraham." Suppose he wants to change because he scheduled too many person-days during some months. In this case, the manager may need information showing the whole training schedule and expected person-days required for training during each month. MSLTRAIN recognizes this situation and provides the information, "TS: Training Schedule; Whole (Table)." However, suppose the manager wants to change the schedule because he scheduled "Bill Muraham" during a milestone period when there are more people scheduled on "Designer" than needed based on desired organizational proficiency. In this case, the manager needs information showing the overage during the period on "Designer." MSLTRAIN displays "Gap between TC and TS; By Package (Table)" of

"Designer." Furthermore, MSLTRAIN places the cursor at the row of "Designer" in the decision window.

Basically MSLTRAIN recognizes the context of the manager's decisions, reasons the goal of the manager, and designs responses according to the manager's goal. In "5.2.2 Responsiveness Process", I explain what MSLTRAIN does internally.

5.2.2 Responsiveness Process

MSLTRAIN does the responsiveness process: observe, understand, interpret, and implement. In the example, MSLTRAIN goes through the following stages when the manager selects "Bill Muraham" for the first time:

1. Observe
    a. Recognize the manager selected "Bill Muraham" in the menu of Step 5.
    b. Infer the manager's input means to start scheduling decision on "Bill Muraham."
    c. Update the interaction history of the manager.
2. Understand
    a. Reset the decision context: the manager has scheduled none.
    b. Infer that the manager wants to schedule "Bill Muraham" on a specific package, "MS Word" because he hasn't schedule "Bill Muraham" and the first package in the decision window is "MS Word."
    c. Infer that the manager needs information showing which month is available for "Bill Muraham" to get training on "MS Word."
3. Interpret

a. Recognize that the scope of information should be *part*, that is, "By Package" from the manager's specific needs and determine that an information item and the portrayal format of the information should be decided further.

b. Decide the information item "Gap between TC and TS" is appropriate for the user by searching the goal-information table that maintains the appropriate information for the user's goal.

c. Decide the portrayal format, "Table," by recognizing the manager preferred "Table."

4. Implement

   a. Display the information, "Gap between TC and TS; By Package (Table)," of "MS Word."

For the other examples, MSLTRAIN goes through the same sequence of stages but different reasoning. These stages are performed by knowledge sources in the responsive layer of MSLTRAIN, which I explain in "5.3 Structure of MSLTRAIN" on page 109. The corresponding computer traces of the example in this section are shown "5.4 Validation of MSLTRAIN" on page 125.

5.2.3 Operational Characteristics

The responsive system identifies the user's needs implicitly, maintains individual user models, and does feedback control as explained in "4.2 Operational Specification of the Responsive System" on page 81. In this section, I explain MSLTRAIN in terms of these characteristics.

Implicit Needs Identification: MSLTRAIN implicitly identifies the user's needs by monitoring the user's input, maintaining the decision context, and reasoning the user's goal, which are explained in the last two sections. The examples in "5.2.1 Adaptive Features" on page 103 exemplify MSLTRAIN's capability to identify the user's needs implicitly; and the stages in "5.2.2 Responsiveness Process" on page 106 shows how MSLTRAIN does this.

Individual User Models: MSLTRAIN maintains individual user models. It recognizes the user by asking the user's name. Each user's characteristics and interaction history is maintained separately in an association list. If MSLTRAIN recognizes the user, MSLTRAIN retrieves information about the user, such as summary of previous interaction history and preferences from the long-term memory and places this information into the working memory. In the example, if the manager has used MSLTRAIN before, MSLTRAIN recognizes the manager by name and retrieves information about the manager, such as the manager's previous interaction summary and the manager's preference for tables and part scope. From the interaction history, MSLTRAIN can infer the manager's proficiency level. The preference information can be used in the design (interpret) stage of the responsiveness process.

Feedback System Control: If the manager doesn't like the information MSLTRAIN provided in the example, he can access information he wants to see. This means there were some mistakes in reasoning or that the manager's preference has changed. MSLTRAIN recognizes this situation and revises the manager's current goal or updates the manager's preference. In the example, when MSLTRAIN displays the information, "Gap between TC and TS; By Package (Table)," suppose the manager accesses and uses another information item, "Gap between TC and TS; By Package (Graph)." This implies the manager may prefer graphs to tables. In this situation, MSLTRAIN updates the manager's preference for portrayal format.

In this section, I explained responsiveness in MSLTRAIN using examples. In the next section, I explain how MSLTRAIN does all those things.

5.3 *Structure of MSLTRAIN*

In this section, I describe the detailed structure of MSLTRAIN: the overall working mechanism and the blackboard architecture.

5.3.1 Overall Working Mechanism

MSLTRAIN works within two processes[40]: the application process and the responsive layer process. The application process takes care of the application program and the user interface in the structure of the responsive system shown in Figure 12 on page 86. The responsive layer process takes care of the responsive layer in the structure of the responsive system. The responsive layer process works as a layer on the application process. The responsive layer process continuously observes the application process and designs or modifies system parameters. Figure 15 on page 111 shows how the two processes work together. If the user inputs something to the system (the up-arrows from the "User" line in the top part of Figure 15), the application process accepts the input (the screened box on the "Application Process" line in the top part of Figure 15), which invokes the responsive layer process (the up-arrows from the "Application Process" line of the top part of Figure 15). Then the responsive layer process analyzes the input, reasons the user's goal, designs an appropriate response (the screened box of the "Responsive Layer Process" line of the top part of Figure 15), and shows the response or sets the values of some system parameters the application process needs (the down-arrows from the "Responsive Layer Process" line of the top part of Figure 15). While the responsive layer process is working, the application process is waiting. The responsive layer process can be turned off. If we stop the responsive layer process, the re-

[40] Actually, TI Explorer doesn't have multiple processors. However, it can have multiple processes running simultaneously by timesharing. The user can define and develop their own processes in their applications in TI Explorer.

sponsive MSLTRAIN will be the same as the original MSLTRAIN that doesn't have responsive features (the bottom part of Figure 15 on page 111).

The connection between the two processes is realized using the the concept of a *reporter* and an *executor*. The reporter resides in the application process and reports to the responsive layer process what happens in the application process such as the user's input, end of processing of some function in the application process, etc. The reporter has knowledge of what and when to report. The executor also resides in the application process and executes what the responsive layer process orders. Through the executor, the responsive layer process can change responses generated by the application process. The responsive layer process can dispatch or recall reporters and executors.

If the reporter reports, the responsive layer process analyzes the report and solves the corresponding problems if needed. In MSLTRAIN, reporters are implemented as demons[41] to functions or methods in the application process. There are currently sixteen reporters and one executor in MSLTRAIN. "Appendix C. Program Source Defining Reporters and Executors" on page 206 shows all reporters and an executor in the form of the Lisp code.

5.3.2 Blackboard

MSLTRAIN maintains a blackboard consisting of eight hierarchical levels (Figure 16 on page 112): input, history, semantics, context, goal, user characteristics, needs, and design. The levels hold intermediate solution results of the responsiveness process as messages. I explain each level in more detail.

---

[41] A demon is a procedure, in the guise of programs, automatically invoked in response to a certain event having occurred on an object (Holsapple, 1987).

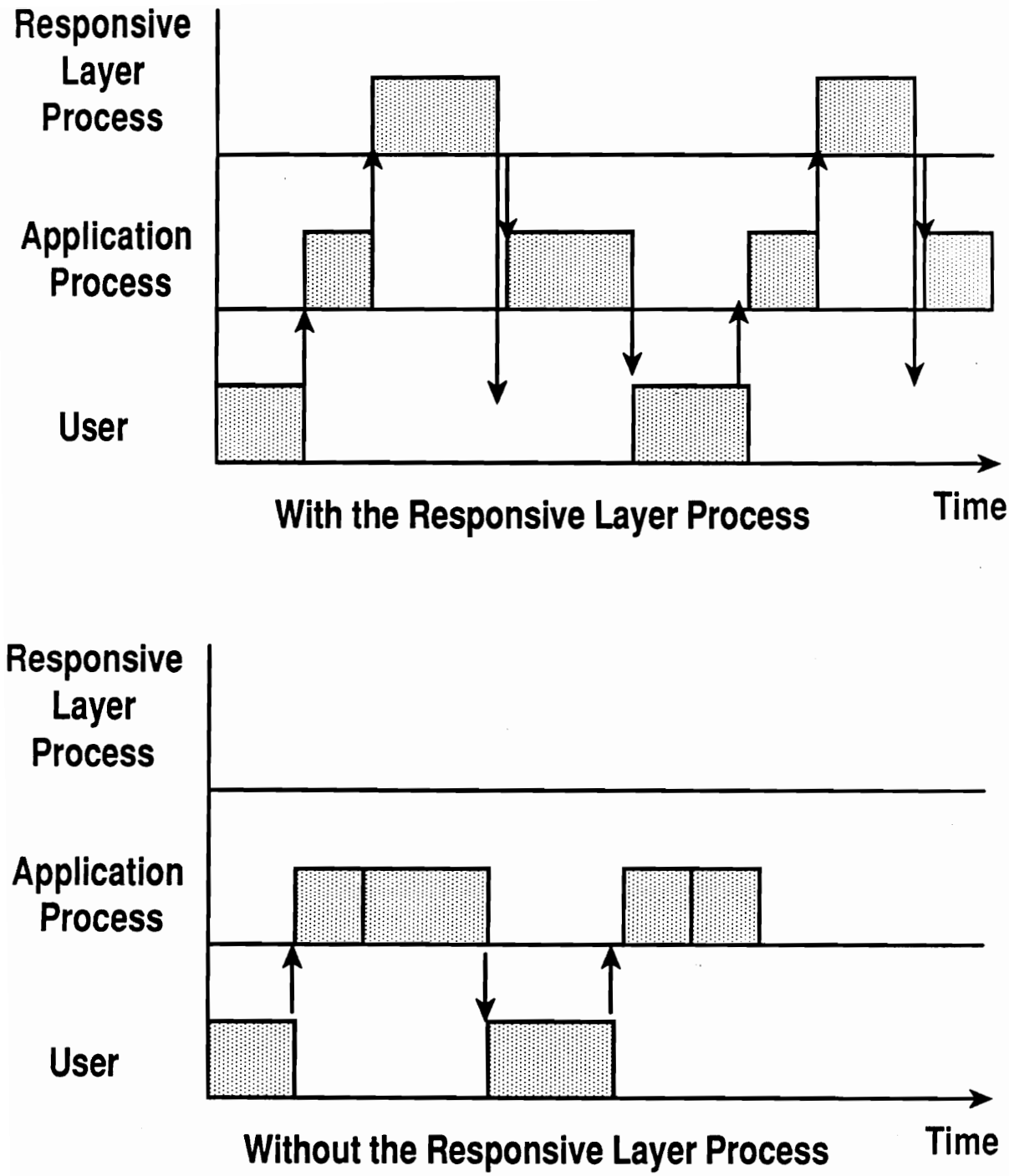


Figure 15. MSLTRAIN works with two processes.

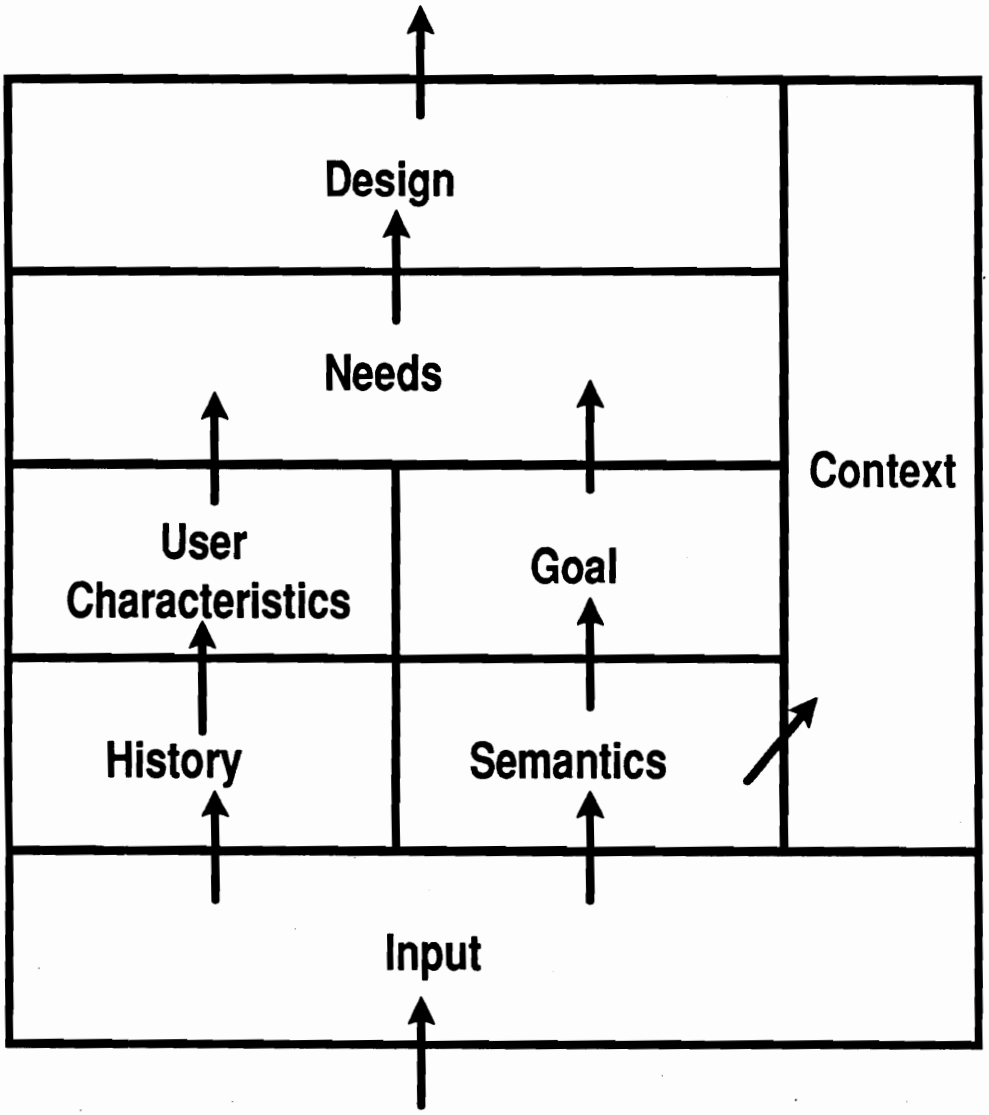


Figure 16. The blackboard in MSLTRAIN consists of eight hierarchical levels.

Input: This level keeps the primitive form of the user's input. Messages in this level contain the following information: type of the input (command or continue), time when the user made the input, content of input, input device (keyboard or mouse), input window (menus, decision windows), input reference (the name of the item selected, number of step, etc.). In the example in "5.2.2 Responsiveness Process" on page 106, the manager's input is posted at this level like

```
(COMMAND 14410 SELECT KEY "MENU 5" "Bill Muraham"),
```

which means the manager selected "Bill Muraham" in the menu in Step 5 with the keyboard at 14410 internal time.

History: This level keeps the history of the interaction with the user. Messages in this level contain the following information: the name of a variable and its value. For example, total number of commands, steps the user went through, total use, total number of errors, total number of inputs made with the mouse, ratio of mouse use, error ratio, etc. In the example in "5.2.2 Responsiveness Process" on page 106, the total number of commands was increased by one.

Semantics: This level keeps what the input means in terms of the application. Messages in this level contain the following information: type of action, object of the action, and reference of the action. In the example in "5.2.2 Responsiveness Process" on page 106, the following message was posted at this level:

```
(START DECISION "Bill Muraham"),
```

which means the input implies the manager wants to start making decisions on "Bill Muraham."

Context: This level keeps the context on decisions. Messages in this level contain the following information: current step, current person the user is working on, current package the user is working on, packages on which the user should schedule the person, proficiency levels on each package the user should schedule and their decision status, etc. In the example in "5.2.2 Responsiveness Process" on page 106, the following message was posted at this level:

```
(5 "Bill Muraham" NIL NIL NIL NIL ("MS Word" "Designer") (((1 0) (2 0)) ((2 0) (3
0))) (0 0) (0 0) (0 0 0) (NIL NIL NIL) 1 NIL 1),
```

which means the current step is 5, the current person the user is working on is "Bill Muraham," the user is not currently working on any package, the packages the user will work on are "MS Word" and "Designer," the user hasn't scheduled "Bill Muraham" yet, etc.

Goal: This level keeps the goal of the user. Messages in this level contain the following information: type of goal (decision or information access), focus of the goal (selection of items in the menu, decision on a specific package, or decision in general), and content of the goal (schedule, change schedule because of the organizational proficiency, or change schedule because of person-days in some month). In the example in "5.2.2 Responsiveness Process" on page 106, the following message was posted at this level:

```
(DECISION 1 0),
```

which means the user wants to schedule on a specific package.

User Characteristics: This level keeps the characteristics of the user. Messages in this level contain the following information: the name of a variable and its value. For example, portrayal format preference, information scope preference, input device preference, decision pattern, etc. In the example in "5.2.2 Responsiveness Process" on page 106 no message was posted at this level. Usually the values are updated with some feedback extracted from the user's response to the system's response.

Needs: This level keeps the user's needs. Messages in this level contain the following information: type of needs (information, cursor move, etc), required type of action, and specification of the needs. In the example in "5.2.2 Responsiveness Process" on page 106, the following message was posted at this level:

```
(INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0))),
```

which means the user needs some information focusing on a specific package at the information display pane.

Design: This level keeps the design of responses. Messages in this level contain the following information: type of the design, and values of elements of the design. Usually, at first, the elements don't have any values. Knowledge sources fill the message until all elements get values. In the example in "5.2.2 Responsiveness Process" on page 106, the following message was posted at this level:

```
(INFO-DESIGN (NIL BY-PKG NIL 0)),
```

which means the type of the design is information and the first element (information item) and the third element (portrayal format) of the design are not specified yet. The message is finally updated like:

```
(INFO-DESIGN (COMP-RT-TS BY-PKG 0 0)),
```

which means the design of the response is completed and the information item is COMP-RT-TS ("Gap between TC and TS"), the scope is BY-PKG ("By PKG"), the portrayal format is 0 ("Table"), and the display place is 0 ("Information Display Pane").

5.3.3 Knowledge Sources

In MSLTRAIN, there are fifty knowledge sources. Knowledge sources are the key of the responsiveness process. They have the knowledge of observing, understanding, interpreting, and implementing. In MSLTRAIN, each knowledge source maintains the following attributes:

- Name;
- Problem-Domain — Observe, Understand, Interpret, or Implement;
- Blackboard — the name of the blackboard the KS works on;
- Description;
- From-Level — the level in the blackboard whose changes trigger the KS;
- To-Level — the level in the blackboard the KS posts messages on;
- Trigger-Condition — conditions specifying which events the KS can be triggered on;

- Pre-Condition — conditions specifying under which solution state the KS can be executed; and
- Action — the executable part of the KS producing changes on the blackboard.

Knowledge sources can take the following actions to the blackboard:

- POST-MSG — post a message on a level in the blackboard,
- UPDATE-MSG — update a message already posted on a level in the blackboard, and
- FILL-MSG — fill in an attribute value to a blank message on a level in the blackboard.

Figure 17 on page 117 shows an example of a knowledge source, DECIDE-DISPLAY-FORM, explained in plain English terms instead of Lisp code. "Appendix E. Program Source Defining Knowledge Sources" on page 220 provides the full program source defining all knowledge sources in MSLTRAIN.

Each knowledge source corresponds to a stage in the responsiveness process: observe, understand, interpret, and implement. I explain each knowledge source by each group.

5.3.3.1 Observing Knowledge Sources

Knowledge sources in this group are triggered by the reporter's report or events generated by the change in the "INPUT" level in the blackboard. These KS's post their messages to the "SEMANTICS" level or update messages in the "HISTORY" level in the blackboard. The following list shows what each knowledge source does when executed.

- From the Reporters
  - ACCEPT-REPORT — accept and process the report sent by the reporters in the application process.

Name:	DECIDE-DISPLAY-FORM
Problem-Domain:	INTERPRET
Blackboard:	Main Blackboard
Description:	"Decide the portrayal format of information the user may prefer."
From-Level:	DESIGN
To-Level:	DESIGN
Trigger-Condition:	An event creates a new design D, whose type is "Information," "Information Menu," or "Information Item" and whose portrayal format value is not determined yet.
Pre-Condition:	The information item element of the design D has some value.
Action:	

1. If the information item supports only one portrayal format, set that format to be the value of the portrayal format element of the design D.
2. If the information item supports more than one portrayal format and there is any format the user prefers, set that format to be the value of the portrayal format element of the design D.
3. If the information item supports more than one portrayal format and the user's preference on format is not known yet, set "Table" to be the value of the portrayal format element of the design D.

Figure 17. Each knowledge maintains several attribute values.

- From the "Input" Level

  - INITIALIZE-NEW-SESSION — initialize the values of the responsive layer process parameters before starting a new session.

  - RECOGNIZE-USER — recognize the user by asking for name and retrieve information about the user.

  - UPDATE-INT-HISTORY — update interaction history.

  - INFER-TOP-SEMANTICS — infer what the user's input means at the top level commands.

  - INFER-STEP-SEMANTICS — infer what the user's input means at the step level commands.

  - INFER-ACCESS-SEMANTICS — infer what the user's input means when the input is related to information access.

  - HANDLE-FOLLOW-UP — handle when the application process ends updating the database about the user's decision.

  - INFER-CONFIRM-SEMANTICS — infer what the user's input means when the user responded to the "YES" "NO" question of the system.

  - SAVE-SESSION — save information about the session and the user's characteristics when the user ends the session.

- From the "HISTORY" Level

  - COMPUTE-INT-CHARS — update the user's interaction characteristics.

- UPDATE-DECISION-PATTERN-HIS — update the user's decision pattern history.

5.3.3.2 Understanding Knowledge Sources

Knowledge sources in this group are triggered by events generated by the change in the "SEMANTICS," "HISTORY," or "GOAL" level in the blackboard. These KS's post their messages to the "GOAL," the "USER-CHAR," or "NEEDS" level in the blackboard. The following list shows what each knowledge source does when executed.

- From the "SEMANTICS" Level

  - UPDATE-STEP-CONTEXT — update the context of the user's decisions when the user starts or continues a step.

  - UPDATE-DECISION-CONTEXT — infer the context of the user's decisions when the user started or continued a decision in a step.

  - INFER-STEP-GOAL — infer the user's goal when the user started or continued a step.

  - INFER-DEC-START-GOAL — infer the user's goal when the user started a decision in a step.

  - INFER-DEC-SET-GOAL — infer the user's goal when the user made a decision in a step.

  - INFER-DEC-CHANGE-GOAL — infer the user's goal when the user changed a decision in a step.

  - INFER-CURSOR-MOVE-GOAL — infer the user's goal when the user moved the cursor.

- INFER-WANT-INFO-GOAL — infer the user's goal when the user started to access information.

- INFER-ACCESS-GOAL — infer the user's goal when the user requested an information item.

- From the "HISTORY" Level

  - INFER-USER-PROFICIENCY — infer the user's proficiency level on MSLTRAIN from the user's interaction history.

  - INFER-DEVICE-PREFERENCE — infer which input device the user prefers from the user's interaction history.

  - INFER-DECISION-PATTERN — infer the user's decision pattern from the user's interaction history.

  - UPDATE-USER-UNDERSTANDING — update the understanding level of the user on MSLTRAIN steps.

  - UPDATE-FORM-PREFERENCE — update the user's preference on portrayal format.

  - UPDATE-VIEW-PREFERENCE — update the user's preference on information scope.

- From the "GOAL" Level

  - SUPPORT-SESSION-CONTINUE — infer the user's needs when the user wants to continue an unfinished previous session.

  - SUPPORT-DECISION-MAKING — infer the user's needs when their goal is to make decisions.

- SUPPORT-MENU-ACCESS — infer the user's need when their goal is to access information

- SUPPORT-ITEM-ACCESS — infer the user's need when their goal is to access information

- From the "USER-CHAR" Level

  - SUPPORT-PROFICIENCY-CHANGE — infer the user's needs when the user's proficiency has changed.

  - SUPPORT-DEVICE-PREFERENCE — infer the user's needs when the user has a certain preference on input devices.

  - SUPPORT-USER-EVOLUTION — infer the user's needs when the user seems to understand a certain step in MSLTRAIN.

5.3.3.3 Interpreting Knowledge Sources

Knowledge sources in this group are triggered by events generated by a change in the "NEEDS" or "DESIGN" level in the blackboard. These KS's post their messages to the "DESIGN" level or fill the messages in the "DESIGN" level in the blackboard. The following list shows what each knowledge source does when executed.

- From the "NEEDS" Level

  - DECIDE-INTERACTION-MODE — decide the interaction mode appropriate to the user.

- DECIDE-ERROR-MESSAGE-P — decide whether to show error messages or just to beep.

- DECIDE-SHOW-SUGG-MENU-P — decide whether to show the information menu pane or not.

- DECIDE-MENU-POSITION — decide the position of the guide pane.

- DESIGN-NETWORK-FLOW — decide which nodes in the full guide mode should be removed or added.

- DESIGN-INFORMATION — infer which elements in the information design should be determined.

- DESIGN-CURSOR-POSITION — decide where to put the cursor.

- From the "DESIGN" Level

  - REASON-INFO-ITEM — reason the information item the user may need.

  - DECIDE-DISPLAY-FORM — reason the information portrayal format the user may prefer.

  - DECIDE-POP-UP — reason where to display information.

  - DECIDE-VIEW-POINT — reason the information scope the user may prefer.

5.3.3.4 Implementing Knowledge Sources

Knowledge sources in this group are triggered by events generated by the change in the "DESIGN" level in the blackboard. These KS's display information to the user or set the values of system parameters. The following list shows what each knowledge source does when it is executed.

- ARRANGE-SYSTEM-PARAMETERS — set the values of system parameters.
- INFO-DISPLAY — display information designed.
- PLACE-CVV-CURSOR — order the executor, CVV-CURSOR-PLACER, to place the cursor at the designed location.
- ARRANGE-INFO-DISPLAY — set the values of information display parameters.
- ARRANGE-NETWORK — remove or add nodes in the full guide interaction mode.

5.3.4 Scheduler

In the responsive layer process, a scheduler controls the blackboard actions: triggering, choosing, and executing knowledge sources. The scheduler maintains the following information:

- Goal — the goal to accomplish, given according to the report from the application process;
- Events — a list of events generated by a change in the blackboard;
- Triggered KSAR's — a list of knowledge source activation records (KSAR) that were triggered but didn't satisfy the precondition;

- Invokable KSAR's — a list of knowledge source activation records that were triggered and satisfied the precondition; and
- Control Strategy — a strategy to choose among the invokable KSAR's.

The scheduler goes through the following steps iteratively.

1. Check whether the goal is satisfied. If so, stop. Otherwise, go to the next step.

2. Update the set of KSAR's. Inspect all knowledge sources to determine if they can be triggered by the event created. Add the knowledge sources triggered to the triggered-KSAR list.

3. Inspect all KSAR's in the triggered-KSAR list to determine if their precondition can be satisfied with the current solution state. If so, add them to the invokable-KSAR list and delete them from the triggered-KSAR list.

4. If the Invokable-KSAR list is empty, stop. Otherwise, go to the next step.

5. Select one KSAR from the invokable-KSAR list with the control strategy. The strategy is to choose one KSAR whose To-Level is the lowest level in the blackboard among the invokable KSAR's.

6. Execute the selected KSAR. Delete the KSAR from the invokable-KSAR list.

7. Go to Step 1.

If the scheduler finishes the steps, the responsive layer process stops and waits until there is new report from the application process. "Appendix D. Program Source Defining the Blackboard Architecture" on page 212 shows the program source defining the whole blackboard architecture: the scheduler, the blackboard, and the knowledge sources.

5.4 *Validation of MSLTRAIN*

Deutsch (1982) defines *validate* as "to ensure that a product or a program functions and contains the features as prescribed by its requirements and specifications." This validation is to check whether the responsive system does what it is supposed to do. For the validation of MSLTRAIN, I provide actual computer traces taken from a real session in MSLTRAIN. The traces show what the responsive layer process did internally. The purpose of providing the traces is to demonstrate MSLTRAIN does the responsiveness process. The traces include:

- Name of the knowledges source triggered and executed during the session, which appears right after `">> KS:."`
- Name of the level in the blackboard whose change caused the knowledge source to be triggered.
- Types of action the knowledge source took: POST-MSG, UPDATE-MSG, FILL-MSG, or SET-PARAMETER.
- Name of the level in the blackboard where the knowledge source posted a message.
- Content of the message, which appears right after `"MSG:."`

The following is the part of the traces corresponding the example in "5.2.2 Responsiveness Process" on page 106. The traces after ### OBSERVING, ### UNDERSTANDING, ### INTERPRETING, and ### IMPLEMENTING represent the observing, the understanding, the interpreting, and the implementing stages in the responsiveness process, respectively. The messages associated with the example are also explained in "5.3.2 Blackboard" on page 110.

```
### OBSERVING

>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 14410 SELECT KEY MENU 5 Bill Muraham)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (START DECISION Bill Muraham)

>> KS: UPDATE-INT-HISTORY
```

```
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

### UNDERSTANDING

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bill Muraham NIL NIL NIL NIL (MS Word Designer)
               (((1 0) (2 0)) ((2 0) (3 0))) (0 0) (0 0) (0 0 0) (NIL NIL NIL) 1 NIL 1
               NIL PART TABLE)

>> KS: INFER-DEC-START-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 1 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0)))
        POST-MSG on NEED Level, MSG: (CURSOR PLACE (ROW))

### INTERPRETING

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PKG NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM COMP-RT-TS)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 0)

>> KS: DESIGN-CURSOR-POSITION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (CURSOR-POSITION 0)

### IMPLEMENTING

>> KS: PLACE-CVV-CURSOR

>> KS: INFO-DISPLAY
```

The entire list of traces are shown in "Appendix F. Computer Traces of the Responsiveness Layer Process" on page 241.

5.5 Conclusion

In this chapter, I explained the prototype responsive management tool, MSLTRAIN, I developed to demonstrate we can really implement the concept of responsiveness into a management tool. The computer traces illustrate that MSLTRAIN does what it is supposed to do. However, this doesn't tell whether MSLTRAIN accomplishes what it is designed to accomplish: gain a better fit between the tool and the users using it. In the next chapter, I explain a laboratory experiment I've done to check whether MSLTRAIN accomplishes what it is designed to accomplish.

Chapter 6. Evaluation of MSLTRAIN

The purpose of this chapter is to describe the experiment I performed to evaluate the prototype responsive management tool and to discuss the result of the experiment. In this chapter, I investigate whether the prototype accomplishes what it was designed to accomplish.

6.1 Introduction

The prototype responsive management tool, which I now call R-MSLTRAIN, was developed to be the same as the original MSLTRAIN, which I call O-MSLTRAIN, in every way except R-MSLTRAIN has the responsive layer. R-MSLTRAIN has additional adaptive features[42] compared to O-MSLTRAIN. The purpose of adding adaptive features is to improve the fit between MSLTRAIN and the users using it; more specifically, to improve user performance and user satisfaction with MSLTRAIN. R-MSLTRAIN provides the user with information the user may need in the format the user may prefer. I believe this will reduce the burden of the user in accessing in-

[42] See "5.2 Responsiveness in MSLTRAIN" on page 99.

formation, in terms of mental load, time, and keystrokes and provide more effective information support to the user. As a result of the adaptive features, I suspect users will like the system better and will be more likely to use the system again.

However, the adaptive features also have some negative aspects[43]. For example,

- Because of additional reasoning, the system requires more response time.
- Because the response of the system is not directly related to the user's input, the user may feel the system is inconsistent and unpredictable.
- When reasoning is wrong, the response of the system may distract the user.

These aspects may inhibit the user's performance and satisfaction with the system. However, I suspect overall the benefit of the adaptive features overcomes the drawbacks of the adaptive features. This experiment intends to test these speculations.

6.2 *Dependent Variables of Interest*

I'm primarily interested in the fit between the management tool and the user using it. As explained in "2.2.3 Operational Measures of Fit" on page 30, it's hard to measure *fit* directly. Usually we measure *fit* indirectly in terms of user performance and user perception. In this study, I'm interested in both user performance and user perception measures.

[43] See "2.4.3 Arguments against Adaptive Systems" on page 51.

6.2.1 User Performance Measures

We use time, number of errors, number of keystrokes, performance scores, etc as performance measures as I reviewed in "2.2.3 Operational Measures of Fit" on page 30. Among them, I'm interested in the following four user performance measures:

Total completion time (TCT): duration from the time when the user starts the session to the time when the user completes the session.

Total information access time (TAT): total time the user spends in accessing information. It's the sum of time periods from the time the user hits the "INFO" key to access information to the time the user hits the "END" key to finish information access.

Total number of keystrokes needed to complete the task (TCK): total number of keystrokes the user hits from the time the user starts the session to the time the user completes the task.

Total number of keystrokes needed to access information (TAK): total number of keystrokes the user hits to access information. It's the sum of keystrokes the user hits during the time periods from the time the user hits the "INFO" key to access information to the time the user hits the "END" key to finish information access.

TCT and TCK includes TAT and TAK, respectively.

6.2.2 User Perception Measures

I'm interested in the following two user perception measures:

Overall Rating of the System: the user's judgment of the system's performance, considering all aspects of the system.

Satisfaction with the System: the user's overall satisfaction with the system's performance computed from the user's judgment of specific aspects of the system.

For these measures, I designed an instrument shown in "Appendix H. User Perception Questionnaire" on page 268.[44] The questions can be classified into four categories:

1. Overall rating of the system — Q18;
2. Satisfaction with general aspects of the system — Q2, Q16, and Q17;
3. Satisfaction with aspects on which responsive systems are expected to be better — Q1, Q3, Q4, Q5, Q6, Q10, and Q15; and
4. Satisfaction with aspects on which responsive systems are expected to be worse — Q7, Q8, Q9, Q11, Q12, Q13, and Q14.

I operationally define "Overall Rating of the System" as the score on Question 18 and "Satisfaction with the System" as the average of the scores on Question 1 through Question 17. Note that the number of questions in the third category is the same as that in the fourth.

---

[44] I adapted some questions from Hiltz & Johnson (1990) and Chin et al (1988).

6.3 *Experimental Hypotheses*

The experimental hypotheses[45] are:

1. The user will spend less time to complete a task when using R-MSLTRAIN than when using O-MSLTRAIN.

2. The user will spend less time to access information during a task when using R-MSLTRAIN than when using O-MSLTRAIN.

3. The user will hit fewer keystrokes to complete a task when using R-MSLTRAIN than when using O-MSLTRAIN.

4. The user will hit fewer keystrokes to access information during a task when using R-MSLTRAIN than when using O-MSLTRAIN.

5. The user will rate R-MSLTRAIN better than O-MSLTRAIN.

6. The user will be more satisfied with R-MSLTRAIN than with O-MSLTRAIN.

These conjectures are reflected in the experimental design.

---

[45] These are not null hypotheses actually tested in the experiment. An experimental hypothesis is a conjecture that gives direction to the experiment (Leedy, 1980). The null hypotheses of the experiment are shown in "6.4.2 Null Hypotheses" on page 135.

6.4 *Experimental Method*

6.4.1 Experimental Design

The experiment was a 2 X 2 mixed factorial design (Table 2): two levels of **responsiveness** (within-subject) and two levels of **order** (between-subject).

Responsiveness: Responsiveness is manipulated by incorporating two kinds of MSLTRAIN: one with responsive features (R-MSLTRAIN) and one without responsive features (O-MSLTRAIN). Major difference in the levels is adaptivity of the system explained in Figure 10 on page 75. In the experiment, only the following responsive features are employed in R-MSLTRAIN:[46]

- Providing information appropriate for the user, in terms of kind, format, and scope even without the user's request, and

- Locating the cursor at the appropriate place.

Simply, if we stop the responsive layer process of R-MSLTRAIN explained in "5.3 Structure of MSLTRAIN" on page 109, it will be the same as O-MSLTRAIN.

Order of Use: Subjects used both versions of MSLTRAIN. Therefore, there can be a learning effect from the use of the first system. To account for the effect, half of the subjects used R-MSLTRAIN first and then O-MSLTRAIN. The other half of the subjects used O-MSLTRAIN first and then R-MSLTRAIN.

---

[46] I didn't employ some features intentionally, such as initial tuning features because they have meaning only when the user uses R-MSLTRAIN several times. In the experiment, subjects used R-MSLTRAIN only twice: once in the training session and once in the experimental session.

Table 2. The experiment used a 2x2 mixed factorial design (order x responsiveness).

Subject Group	First Run	Second Run
O-MSLTRAIN first (S1 - S10)	O-MSLTRAIN	R-MSLTRAIN
R-MSLTRAIN first (S11 - S20)	R-MSLTRAIN	O-MSLTRAIN

6.4.2 Null Hypotheses

The null hypothesis (**H0**) and the alternative hypothesis (**H1**) for each variable are as follows:

- Total Completion Time (TCT)

  H0: There will be no difference in the amount of time to complete the task when using R-MSLTRAIN and when using O-MSLTRAIN.

  H1: There will be a difference in the amount of time to complete the task when using R-MSLTRAIN and when using O-MSLTRAIN.

- Total Information Access Time (TAT)

  H0: There will be no difference in the amount of time spent to access information when using R-MSLTRAIN and when using O-MSLTRAIN.

  H1: The user will spend less time accessing information when using R-MSLTRAIN than when using O-MSLTRAIN.

- Total Number of Keystrokes Needed to Completion the Task (TCK)

  H0: There will be no difference in the number of keystrokes needed to complete the task when using R-MSLTRAIN and when using O-MSLTRAIN.

  H1: There will be a difference in the number of keystrokes needed to complete the task when using R-MSLTRAIN and when using O-MSLTRAIN.

- Total Number of Keystrokes Needed to Access Information (TAK)

  H0: There will be no difference in the number of keystrokes needed to access information when using R-MSLTRAIN and when using O-MSLTRAIN.

H1: The user will hit fewer number of keystrokes to access information when using R-MSLTRAIN than when using O-MSLTRAIN.

- Overall Rating of the System

H0: The user will rate R-MSLTRAIN and O-MSLTRAIN as the same.

H1: The user will rate R-MSLTRAIN and O-MSLTRAIN differently.

- Satisfaction with the System

H0: There will be no difference in the level of satisfaction with the system when using R-MSLTRAIN and when using O-MSLTRAIN.

H1: There will be a difference in the level of satisfaction with the system when using R-MSLTRAIN and when using O-MSLTRAIN.

Note: Two alternative hypotheses for TCK and TAK have a direction.

6.4.3 Subjects

Subjects consisted of twenty paid volunteers (graduate students) enrolled at Virginia Tech. Table 3 on page 137 shows the demographics of the subjects. All were familiar with management decision environments and with the use of interactive computers in general, but none had previous experience with MSLTRAIN. Eight (40%) of the subjects replied they had heard the term *responsive system* before. All subjects were paid $25 after the experimental session.

Table 3. Twenty subjects participated in the experiment.

Gender	No. of Subjects	Percent
Male	15	75.0
Female	5	25.0

Age	No. of Subjects	Percent
Under 20	0	0.0
20-25	9	45.0
26-30	7	35.0
Over 30	4	20.0

Major	No. of Subjects	Percent
Human Factors	5	25.0
Manufacturing	6	30.0
MgtSE	5	25.0
MBA	3	15.0
OR	1	5.0

Computer Experience	No. of Subjects	Percent
No experience	0	0.0
Some experience	8	40.0
Experienced	8	40.0
Very experienced	4	20.0

6.4.4 Experimental Apparatus

MSLTRAIN runs on a Texas Instruments (TI) Explorer, which I explained in "5.1.3 Hardware and Software Environment" on page 98. In the experiment, only the keyboard was used as an input device to remove the effect of using different input devices, such as the mouse. The system was placed on a table with two stands: one for the monitor and one for the keyboard. The table was located at the one side of the room and the main body (CPU and mass storage device) was located at the other side of the room to reduce the noise for the subject from it.

6.4.5 Experimental Task

Each subject was asked to assume he or she was a manager of an organization comprising eight people in addition to the manager. The subject made scheduling decisions under a scenario shown in "Appendix J. Instruction Packet" on page 275. The subject did only Step 5 of MSLTRAIN. The previous steps were assumed to be completed before the experiment; selecting computer packages the organization will need, determining milestone periods, determining organizational proficiency, and determining desired individual proficiencies.[47] The data were provided in the scenario and installed into the database of MSLTRAIN before the experiment so that subjects could access them whenever they wanted.

The experimental task was to select the months each person in the organization needed to reserve for training sessions. To complete the task, the subject should satisfy the following constraints:

- Satisfy the organizational needs in terms of milestones.

---

[47] For more details, see "5.1.1 Application Domain" on page 93.

- Schedule all the training sessions eight people in the organization require to take based on the desired individual proficiencies.
- Schedule people only on the month when they are not expected to be too busy to get training.
- Schedule people on only one session during a month.
- Schedule no more than nine person-days during a month.

If the subject satisfied all the constraints, the system notified the subject he or she had completed the task.

6.4.6 Experimental Protocol

Subjects had two sessions occurring on separate days. The first was a training session and the second was an experimental session. The reason for splitting the sessions was to remove the effect of fatigue in the training session because the training session was expected to take more than an hour.

6.4.6.1 Training Session

Subjects were trained on the computer equipment (Texas Instruments Explorer Workstation) they were going to use and on the domain they were going to work on. The following describes the training session protocol.

1. The subject read the introduction packet shown in "Appendix I. Introduction Packet" on page 271 and signed the Participant Informed Consent form.
2. The subject read the scenario shown in "Appendix J. Instruction Packet" on page 275.
3. The experimenter demonstrated how to use both versions of MSLTRAIN.

4. The subject practiced how to use both versions of MSLTRAIN by following the instruction shown in "Appendix J. Instruction Packet" on page 275.
5. The experimenter explained information items the subject might need to make decisions during the experimental task.
6. The experimenter explained constraints the subject should satisfy to complete the experimental task.
7. The experimenter explained strategies the subject should follow during the experimental task.
8. The experimenter loaded one of the MSLTRAINs.
9. The subject completed the experimental task as a practice using the MSLTRAIN the experimenter loaded.
10. The subject had a five-minute break.
11. The experimenter loaded the other MSLTRAIN.
12. The subject completed the experimental task using the other MSLTRAIN.

In the training session, subjects could ask any question concerning how to use MSLTRAIN and the information items they needed to make decisions. The experimenter helped the subject when the subject seemed to have difficulties. No data were collected at this session.

6.4.6.2 Experimental Session

The second session was an experiment. The experimental session was held on the day after the training session. The following describes the experimental session protocol.

1. The experimenter reminded the subject of how to use the keys, the information items, constraints, and strategies.
2. The experimenter loaded one of the MSLTRAINs.
3. The subject completed the experimental task using the MSLTRAIN the experimenter loaded.

4. The subject filled out the questionnaire shown in "Appendix H. User Perception Questionnaire" on page 268.
5. The subject had a five-minute break.
6. The experimenter loaded the other MSLTRAIN.
7. The subject completed the experimental task using the other MSLTRAIN.
8. The subject filled out a questionnaire asking about his or her satisfaction with the other MSLTRAIN.
9. The subject filled out the subject information questionnaire shown in "Appendix K. Subject Information Questionnaire" on page 283.

During the experimental session, the subject wasn't allowed to ask questions. The system automatically recorded all user performance measures and saved those measures in a disk file. Time was measured in minutes. The user perception measures were taken with the questionnaire. The questionnaires subjects filled out were put in an envelope, which wasn't opened until the experiment was complete. The user performance and user perception data are shown in "Appendix M. Raw User Performance and User Perception Data" on page 287.

6.5 Data Analysis and Results

6.5.1 Analysis of User Performance Data

I used an analysis of variance (ANOVA) procedure to determine the effect of the level of responsiveness on the user performance measures: total completion time, total information access time, total number of keystrokes needed to complete the task, and total number of keystrokes needed to

Table 4. ANOVA Summary Table for User Performance Results

Dependent Variables	Source	Degrees of Freedom	Sum of Squares	Mean Squares	F-Ratio
Total Completion Time	Between Subjects				
	Order (O)	1	85.12	85.12	5.16
	Subject within groups	18	296.99	16.50	
	Within Subjects				
	Responsiveness (R)	1	71.51	71.51	12.06*
	R x O	1	82.80	82.80	13.97*
	R x Subject within groups	18	106.72	6.93	
Total Information Access Time	Between Subjects				
	Order (O)	1	7.97	7.97	2.20
	Subject within groups	18	65.16	3.62	
	Within Subjects				
	Responsiveness (R)	1	75.85	75.85	22.10#
	R x O	1	7.32	7.32	2.13
	R x Subject within groups	18	61.79	3.43	
Total Number of Keystrokes Needed to Complete the Task	Between Subjects				
	Order (O)	1	18020.03	18020.03	4.65
	Subject within groups	18	69745.85	3874.77	
	Within Subjects				
	Responsiveness (R)	1	113103.23	113103.23	40.37*
	R x O	1	11799.23	11799.23	4.21
	R x Subject within groups	18	50428.05	2801.56	
Total Number of Keystrokes Needed to Access Information	Between Subjects				
	Order (O)	1	883.60	883.60	0.80
	Subject within groups	18	19799.80	1099.99	
	Within Subjects				
	Responsiveness (R)	1	55950.40	55950.40	57.85#
	R x O	1	462.40	462.40	0.48
	R x Subject within groups	18	17410.20	967.23	

* $p < 0.01$, two-tailed. # $p < 0.01$, one-tailed.

access information.[48] Table 4 on page 142 shows an ANOVA summary table of user performance results. Table 5 shows the cell means of each user performance measure.

Total Completion Time (TCT): Table 4 on page 142 indicates there is a responsiveness effect on TCT, $\underline{F}(1,18) = 12.06$, $p < 0.003$ and also an interaction effect of responsiveness and order on TCT, $\underline{F}(1,18) = 13.97$, $p < 0.002$. Table 5 on page 144 indicates the mean value of TCT when using R-MSLTRAIN ($\underline{M} = 11.40$) is shorter than that when using O-MSLTRAIN ($\underline{M} = 14.08$). The interaction effect on TCT seems to be due to the subjects' learning on the system. Figure 18 on page 145 shows that for both the R-MSLTRAIN-first and the O-MSLTRAIN-first groups the subjects took less time for the second run than for the first run of the system. This implies there was a learning effect in the experiment. The O-MSLTRAIN-first group took much less time in the second run ($\underline{M} = 11.42$) than in the first run ($\underline{M} = 16.97$), whereas the R-MSLTRAIN-first group took almost same time in the second run ($\underline{M} = 11.17$) as in the first run ($\underline{M} = 11.38$). This implies that in the second run of the O-MSLTRAIN-first group the learning effect was added on the responsiveness effect (Mean Difference = 5.55). In the other hand, in the second run of the R-MSLTRAIN-first group the learning effect offset the responsiveness effect (Mean Difference = 0.21).

Total Information Access Time (TAT): Table 4 on page 142 indicates there is a responsiveness effect on TAT, $\underline{F}(1,18) = 22.10$, $p < 0.0001$. Table 5 on page 144 indicates the mean value of TAT when using R-MSLTRAIN ($\underline{M} = 0.21$) is shorter than that when using O-MSLTRAIN ($\underline{M} = 2.96$).

Total Number of Keystrokes Needed to Complete the Task (TCK): Table 4 on page 142 indicates there is responsiveness effect on TCK, $\underline{F}(1,18) = 40.37$, $p < 0.0001$. Table 5 on page 144 indicates

---

[48] I used the SAS package. The SAS program for the ANOVA procedure is shown in "Appendix L. SAS Program for the ANOVA Procedure" on page 285.

Table 5. Means and Standard Deviations in Each Cell of the User Performance Measures

Variable	Resp. / Order	O-MSLTRAIN		R-MSLTRAIN		Overall	
		Mean	S.D.	Mean	S.D.	Mean	S.D.
TCT	O first	16.97	4.76	11.42	1.82	14.19	4.52
	R first	11.17	3.35	11.38	2.76	11.28	2.99
	Overall	14.08	4.99	11.40	2.28	12.74	4.06
TAT	O first	3.83	3.43	0.22	0.29	2.03	3.01
	R first	2.09	1.46	0.19	0.34	1.14	1.42
	Overall	2.96	2.72	0.21	0.31	1.58	2.36
TCK	O first	416.70	76.94	276.00	43.62	346.35	94.42
	R first	339.90	65.86	267.90	34.54	303.90	63.12
	Overall	378.30	80.07	271.95	38.52	325.13	82.13
TAK	O first	86.70	46.53	5.10	7.55	45.90	52.96
	R first	70.50	43.38	2.50	4.67	36.50	46.07
	Overall	78.60	44.61	3.80	6.25	41.20	49.22

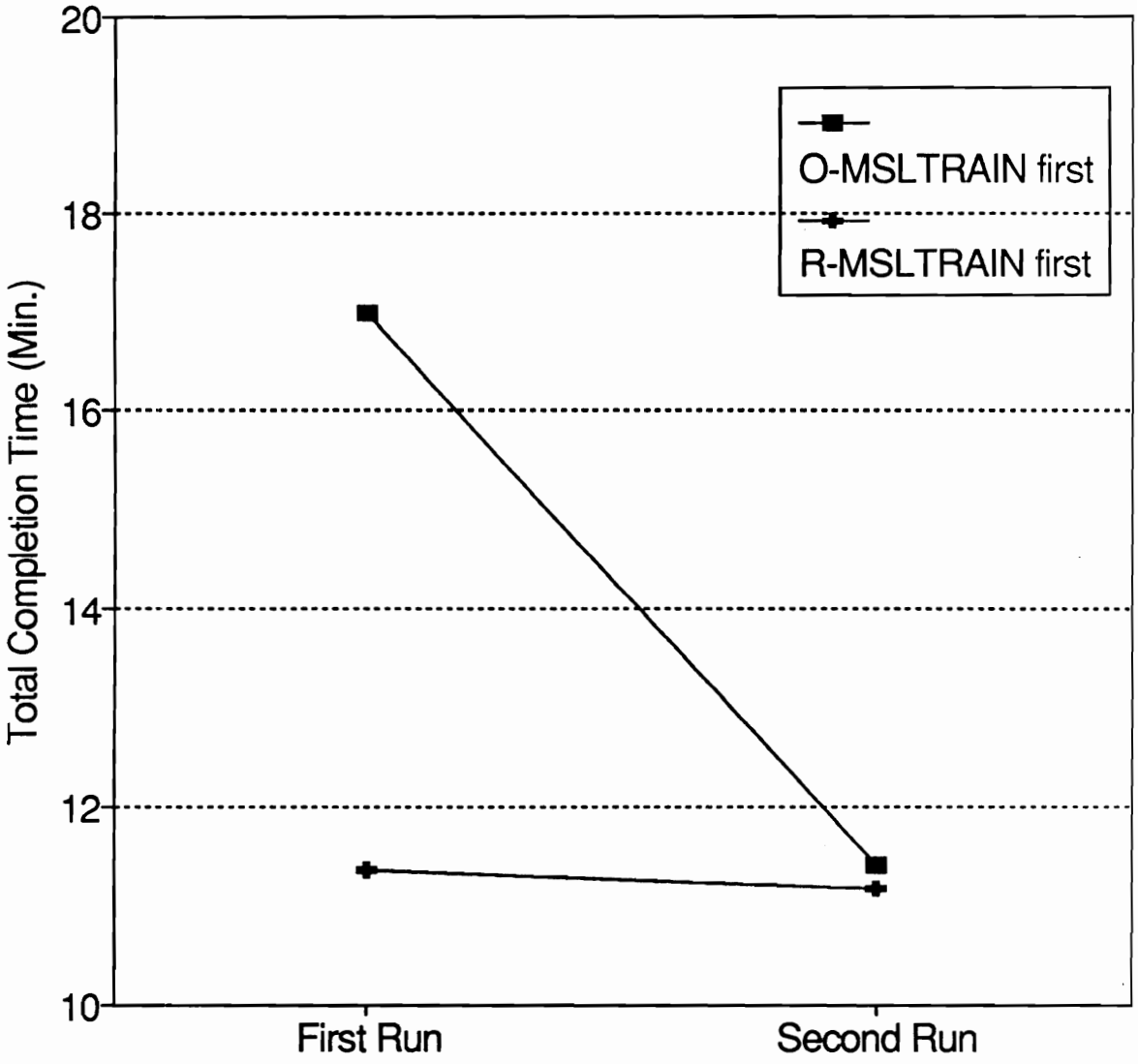


Figure 18. Cell Means of TCT for the First and the Second Run

the mean value of TCK when using R-MSLTRAIN ($\underline{M}$ = 271.95) is smaller than that when using O-MSLTRAIN ($\underline{M}$ = 378.30).

Total Number of Keystrokes Needed to Access Information (TAK): Table 4 on page 142 indicates there is responsiveness effect on TAK, $\underline{F}(1,18) = 57.85, p < 0.0001$. Table 5 on page 144 indicates the mean value of TAK when using R-MSLTRAIN ($\underline{M}$ = 3.80) is smaller than that when using O-MSLTRAIN ($\underline{M}$ = 78.60).

6.5.2 Analysis of User Perception Data

I analyzed user perception data in three phases:

1. Scale the measures,
2. Compute coefficient alphas for the "Satisfaction" measure to test the internal consistency of the questions on the measure, and
3. Use an ANOVA procedure for the measures.

Scaling: Generally, we treat an ordinal scale like a continuous variable if it can be assumed it has an equal distance between the scales (Dunn-Rankin, 1983). For Question 1 through Question 17, I gave the scale value of 1, 2, 3, 4, 5, 6, and 7 for the response items, "strongly disagree," "disagree," "tend to agree," "neither disagree or agree," "tend to agree," "agree," and "strongly agree," respectively. For Question 18, I gave the scale values of 5, 4, 3, 2, and 1 for the response items "excellent," "good," "fair," "poor," and "unsatisfactory," respectively. The scale values of Question 7, 8, 9, and 14 were reversed because the questions were stated negatively. The scores for each question item were used to indicate user perception variables. The "Overall Rating of the System" variable was indicated in responses to Question 18 and the "Satisfaction with the System" variable was indicated in responses to Question 1 through Question 17.

Reliability Coefficient Alpha: To check the internal consistency of questionnaire responses, I computed the *reliability coefficient alpha* for the "Satisfaction" measure, which is computed by the formula

$$\hat{\alpha} = \frac{k}{k-1}\left(1 - \frac{\sum_{i=1}^{k}\hat{\sigma}_i^2}{\hat{\sigma}_x^2}\right),$$

where k is the number of items in the test, $\hat{\sigma}_i^2$ is the variance of item i, and $\hat{\sigma}_x^2$ is the total test variance (Crocker & Algina, 1986). Coefficient alpha can be used as an index of internal consistency and as an estimate for the lower bound of the reliability coefficient.[49]

The value of the coefficient alpha for the "Satisfaction" measure is **.895**. This means at least 89% of the variance of the "Satisfaction" measure is due to true score variance of the question items. Generally, reliability coefficient alphas over .60 are acceptable and alphas over .70 are preferred (Nunnally, 1967). The value for this study exceeds the preferred range of consistency. Therefore, it is reasonable to use the sum of the scores for Question 1 through Question 17 as the measure of "Satisfaction with the System."

Analysis of Variance

I used an ANOVA procedure to determine the effect of the level of responsiveness for the user perception measures: overall rating of the system and satisfaction with the system. Table 6 on page 148 shows an ANOVA summary table for user perception results. Table 7 on page 149 shows the cell means of each variable.

Overall Rating of the System: Table 6 on page 148 indicates there is a responsiveness effect on the overall rating of the system, $\underline{F}(1,18) = 18.45$, $p < 0.0004$. Table 7 on page 149 indicates the mean

---

[49] For more detail, see Crocker & Algina (1986).

Table 6. ANOVA Summary Table for the User Perception Results

Dependent Variables	Source	Degrees of Freedom	Sum of Squares	Mean Squares	F-Ratio
Overall Rating of the System	Between Subjects				
	Order (O)	1	0.23	0.23	0.25
	Subject within groups	18	16.25	0.90	
	Within Subjects				
	Responsiveness (R)	1	7.23	7.23	18.45*
	R x O	1	0.23	0.23	0.57
	R x Subject within groups	18	7.05	0.39	
Satisfaction with the System	Between Subjects				
	Order (O)	1	1.27	1.27	1.75
	Subject within groups	18	13.05	0.73	
	Within Subjects				
	Responsiveness (R)	1	11.78	11.78	30.07*
	R x O	1	0.30	0.30	0.77
	R x Subject within groups	18	7.05	0.39	

* $p < 0.01$

Table 7. Means and Standard Deviations in Each Cell of the User Perception Measures

Variable	Resp. / Order	O-MSLTRAIN		R-MSLTRAIN		Overall	
		Mean	S.D.	Mean	S.D.	Mean	S.D.
Rating	O first	3.60	1.26	4.60	0.52	4.10	1.07
	R first	3.60	0.52	4.30	0.67	3.95	0.69
	Overall	3.60	0.94	4.45	0.60	4.03	0.89
Satisfaction	O first	4.93	1.16	6.19	0.50	5.56	1.08
	R first	4.74	0.54	5.66	0.59	5.20	0.72
	Overall	4.84	0.88	5.92	0.60	5.38	0.93

value of the rating score on R-MSLTRAIN ($\underline{M}$ = 4.45) is larger than that on O-MSLTRAIN ($\underline{M}$ = 3.60).

Satisfaction with the System: Table 6 on page 148 indicates there is a responsiveness effect on the satisfaction with the system, $\underline{F}(1,18) = 30.07, p < 0.0001$. Table 7 on page 149 indicates the mean value of the satisfaction score on R-MSLTRAIN ($\underline{M}$ = 5.92) is larger than that on O-MSLTRAIN ($\underline{M}$ = 4.84).

6.5.3 Informal Observations

In the experiment, I made informal observations. I say they are *informal* because these issues were not incorporated into the experimental design.

First, I found individual differences in terms of the subjects' preferences for information items, the portrayal format, and the information scope. Seven people preferred the graph format. All subjects but two preferred a simpler form of information, such as the "By Package" scope. Three subjects greatly preferred a simpler form of information. For example, in O-MSLTRAIN, the user should request information. To see information with the "By Package" scope, the user should request the information whenever he or she switches the package focused on. For this reason, most subjects request the information with the "Whole" scope because that contains all the packages. However, three subjects always requested information with the "By Package" scope in O-MSLTRAIN even if there is some inconvenience. This kind of individual difference was the motivation of this research.

In the experiment, nineteen of the twenty subjects preferred R-MSLTRAIN to O-MSLTRAIN. One subject preferred O-MSLTRAIN to R-MSLTRAIN. The subject experienced a response of the system that confused him or her; for example, a beep he or she didn't understand. This supports

the discussion of the negative arguments on adaptive systems in "2.4.3 Arguments against Adaptive Systems" on page 51. Sometimes one wrong response can totally confuse the user.

The subjects rated R-MSLTRAIN better than O-MSLTRAIN even for the questions on the negative aspects of R-MSLTRAIN. For example, the response time of R-MSLTRAIN is longer than that of O-MSLTRAIN because of additional reasonings. But the mean score values of Question 14 (response time) in Table 8 on page 152 shows the subjects felt more satisfaction with R-MSLTRAIN ($\underline{M}$ = 5.55) than with O-MSLTRAIN ($\underline{M}$ = 5.30) in terms of response time. Table 8 on page 152 shows the mean values on Question 7, 8, 9, 11, 12, 13, and 14. The mean values for R-MSLTRAIN are larger than those for O-MSLTRAIN, which means the subjects felt more satisfaction with these matters. There are two possible explanations. First, the positive aspects of R-MSLTRAIN cause the subjects to feel better about the negative aspects. Second, the subjects might have been biased to R-MSLTRAIN and didn't answer correctly.

Question 15 asks subjects how strongly they agree that the system seemed to anticipate what they intended to do. For this question, five subjects strongly agreed that R-MSLTRAIN seemed to anticipate what they intended to do, four subjects agreed, and three subjects disagreed. Overall, subjects seemed to agree that R-MSLTRAIN seemed to anticipate what they intended to do. This result supports R-MSLTRAIN accomplished its responsiveness, to a limited degree.

6.6 Discussion

This study compares two versions of MSLTRAIN that differ in responsiveness. From the results of the experiment, I can conclude:

- There was an interaction effect of responsiveness and learning on the total completion time.

Table 8. Mean Scores of Responses to Questions on Negative Aspects of R-MSLTRAIN

Question	O-MSLTRAIN	R-MSLTRAIN	Overall
Q7	4.30	5.10	4.70
Q8	4.45	5.45	4.95
Q9	5.20	5.95	5.58
Q11	5.65	5.80	5.73
Q12	6.10	6.20	6.15
Q13	5.70	5.75	5.73
Q14	5.30	5.55	5.43

- Subjects spent less time to access information when using R-MSLTRAIN than when using O-MSLTRAIN.

- Subjects hit fewer keystrokes to complete the task when using R-MSLTRAIN than when using O-MSLTRAIN.

- Subjects hit fewer keystrokes to access information when using R-MSLTRAIN than when using O-MSLTRAIN.

- Subjects rated R-MSLTRAIN better than O-MSLTRAIN.

- Subjects were more satisfied with R-MSLTRAIN than with O-MSLTRAIN.

The subjects performed better when using R-MSLTRAIN than using O-MSLTRAIN in terms of time and keystrokes. R-MSLTRAIN provides information the user may need even without the user's request. The user doesn't have to request information he or she needs. Therefore, time and keystrokes needed to access information decrease when using R-MSLTRAIN. As the information access time decreases, the total completion time decreases. Less time and fewer keystrokes mean reduction of the user's physical effort to interact with the system, which means higher productivity caused by reducing the amount of input (Sink, 1985). Better performance implies a better match in the interface screen explained in "3.2 A Conceptual Model of Matching" on page 62.

The learning effect on the experiment in terms of the total completion time implies there should have been more training before the experimental session. The mean value of the total completion time for the O-MSLTRAIN-first group's first run is far higher than other means as shown in Table 5 on page 144. This implies the user spent more time when first using O-MSLTRAIN. O-MSLTRAIN doesn't provide information automatically. Untrained users may have problems figuring out how to approach the task. I think there should have been one more practice session just before the experimental session.

The subjects perceived R-MSLTRAIN was better than O-MSLTRAIN in terms of their rating and satisfaction. This also seems to be due to R-MSLTRAIN's capability to provide information even without the user's request. The fact the user didn't have to access information seemed to make the subjects feel better about R-MSLTRAIN, even about the negative aspects of R-MSLTRAIN. Better satisfaction also implies a better match in the interface screen explained in "3.2 A Conceptual Model of Matching" on page 62.

However, the results of the experiment have some limits. First, the results represent a small sample: twenty graduate students in the Department of Industrial and Systems Engineering and the Department of Management. Second, the results are based on a specific task: Step 5 of MSLTRAIN. I can't generalize the results to other domains. Third, the user performance measures and the user perception measures in the experiment are only part of whole constructs. There are many other ways to operationalize user performance and user perception.

Overall, I can say R-MSLTRAIN fits users better than O-MSLTRAIN does in terms of both user performance and user perception. To a limited degree, I can say the results support my hypothesis that responsiveness in MSLTRAIN will improve the match between MSLTRAIN and the managers using it. R-MSLTRAIN demonstrates we can implement the concept of responsiveness into a management tool and the results of the experiment show R-MSLTRAIN accomplishes what it is designed to accomplish: better match in the interface screen. Finally, from the results, I can say the concept of responsiveness in management tools is valuable to improve the match between the tool and the managers using it.

Chapter 7. Conclusion

7.1 Review of the Goal

This research primarily contributes to the body of knowledge of management systems engineering in terms of the Management System Model (MSM). This research studies and operationalizes the information portrayal/information perception interface in the MSM with the concept of responsiveness. The match in the information portrayal/information perception interface is critical for improving the management tool's success and sharing. As outlined in "1.1 The Problem Statement" on page 1, this research intended to operationalize and implement the concept of responsiveness in management tools to provide a way for a management tool to match the varying and evolving needs of managers using it. To accomplish this goal, I

- Presented a conceptual model of matching that provides the definition of responsiveness in management tools. Furthermore, I explained the responsiveness process a management tool goes through to be responsive to the user and three constructs of responsiveness in management tools in terms of their capabilities. These provide the conceptual basis.

- Introduced the responsive system and specified it in terms of its operational characteristics and structure. I combined the concept of responsiveness with existing artificial intelligence technology. These provide the technological basis.

- Developed a prototype responsive management tool satisfying the operational characteristics of the responsive system and realizing the structure of the responsive system. The prototype provides an example of practical application.

- Conducted a laboratory experiment involving twenty human subjects to evaluate the prototype (R-MSLTRAIN) compared to a reference management tool (O-MSLTRAIN). The result of the experiment shows R-MSLTRAIN is superior to O-MSLTRAIN in terms of both user performance and user perception in the sample, which implies responsiveness in a management tool, to a limited degree, improves the matching between the management tool and the users using it.

Overall, I can say this research has accomplished what it intended to do. The operationalization of the concept of responsiveness, the specification of the responsive system, the prototype, and the evaluation of the prototype demonstrates we can implement the concept of responsiveness into a management tool and shows how we can implement the concept.

7.2 Generalizations and Limits

This research has shown the concept of responsiveness and the structure of the responsive system is implemented into MSLTRAIN successfully. However, it is arguable whether the concept and structure can be generalized to other substantive domains. A research is valuable when it is

generalizable and its results can be used by future researchers (Brinberg & McGrath, 1985). I think this research is easily generalizable to other domains based on the following points.

First, I drew the concept of responsiveness in management tools from the human world such as secretaries, nurses, salespeople, etc. This implies the concept of responsiveness I defined is general and robust. We can explain the behavior and the capabilities of responsive secretaries and nurses with the conceptual model of matching, three constructs of responsiveness, and the responsiveness process. I believe a more general concept can be generalized easily.

Second, this research maintained consistency in applying the concept. For example, the responsive system consists of three parts, corresponding to the three constructs of responsiveness. MSLTRAIN works with two processes: one taking care of the responsive layer and one taking care of the application domain and the user interface. I believe a structure designed consistently can be generalized easily.

Third, the blackboard architecture has been used in several areas successfully as reviewed in "4.3.2.1 Blackboard Architecture" on page 88. The use of generally known and frequently used technology enhances the generalizability of this research.

Fourth, the major responsive feature of MSLTRAIN is to provide information appropriate to the user. Information support is critical in most management tools in terms of content, format, and time (Rouse, 1984; Hollnagel, 1986). I believe my selection of a popular feature in management tools helps results be generalized easily.

But there are some limits based on the following points. First, MSLTRAIN was developed on a TI Explorer workstation. All specific features of MSLTRAIN such as using two processes, the concept of reporters and executors, and object-oriented schemes depend on the machine's capability. If I had used another machine, the features would have been different.

Second, the knowledge employed in MSLTRAIN was very specific, in terms of the user's input, semantics, goals, needs, user characteristics etc. If MSLTRAIN had employed more general knowledge, such as knowledge of the relationship between the user's personality and the information portrayal format, the result could be generalized more easily.

Third, the evaluation result is based on a small number of subjects — a small part of the population. Also the result is limited to Step 5 of MSLTRAIN. The limits of the experiment, in terms of sample size and application, inhibits the generalizability of the result of this research.

7.3 Topics for Further Research

The research question I asked was how we could implement the concept of responsiveness into a management tool. Now that I have answered the question, we can proceed to the next step. In this section, I provide some ways to extend this research in terms of the conceptual domain, the technological domain, and the substantive domain.

Conceptual Domain: In the conceptual domain, further research is needed in the following areas:

- Extension of the concept of responsiveness

  This research defined responsiveness only from the individual's perspective. We can extend the concept to the organizational perspective. Markus & Robey (1983) identify four kinds of fit: 1) User - System fit, 2) Organizational Structure - System fit, 3) Power Distribution - System fit, and 4) Environment - System fit. This research focused on the first kind of fit. It will be useful to put the other perspectives of fit into the concept of responsiveness.

In this research, I was interested primarily in the information portrayal/information perception interface in the MSM. The ultimate concept of responsiveness should reflect the "what is managed" component in the MSM. We need more research considering the measurement/data interface in the MSM.

This research didn't concern the issue of response time. Timeliness is critical for providing information to managers. Reasoning required in the responsiveness process takes additional time. There should be some kind of trade-off between better reasoning and quicker response. Further research is needed on taking the issue of response time into the conceptual model of responsiveness and figuring out how we know and implement the trade-off.

In "2.2.1 Decision Support Perspective" on page 22, I discussed the difference between the user's wishes and needs. Management tools can confirm the user's wishes or complement the user's needs. However, often there can be some contradiction. The user may need something he or she doesn't want. In this case, it is difficult to determine whether or not to provide what the user needs (Carroll & McKendree, 1987). If we clarify the difference between the user's needs and wishes and take that into the concept of responsiveness, that will be useful for extending the concept of responsiveness. I suspect deeper study of the projection in the interface screen (P_{U-T}) in Figure 7 on page 64 will help clarify the difference.

- Study of other related concepts

We can think of other concepts requiring higher levels of intelligence explained in Figure 11 on page 82, such as anticipatory systems and consensus-gathering systems. The anticipatory system differs from the responsive system in terms of system control type. The anticipatory system anticipates the user's future state. The consensus-gathering system differs from the responsive system in terms of user model type. The consensus-gathering system maintains interrelated user models.

Technological Domain: We can combine more sophisticated artificial intelligence technology with the concept of responsiveness such as learning and planning techniques. Learning in the responsiveness process is vital. This research didn't employ any sophisticated learning technique because the application didn't involve much learning. If the application allows natural language inputs, there will be more chance to learn about the user. In this kind of application, some artificial intelligence techniques will improve the adaptivity of the system. Michalski et al (1983) classify learning systems based on the underlying learning strategies used: rote learning and direct implanting of new knowledge, learning from instruction, learning by analogy, learning from examples, and learning from observation and discovery. If we combine learning from observation into the responsive system, we will have a higher level of adaptivity in the responsive system.

Planning techniques used in Peachery & McCalla (1986) can be applied both for inferring the user's intention (plans) and for designing appropriate responses (response plans). Implicit acquisition of user models suggested by Kass & Finin (1989) can be also applied for inferring the user's goals if there is some development in that area.

Substantive Domain: There are two alternatives for further research in this domain: refining MSLTRAIN more and extending MSLTRAIN to other application domains.

Responsiveness has not been implemented in all steps of MSLTRAIN, such as Step 1. Implementation of responsiveness in Step 1 can be a challenge because, in Step 1, the user goes through some unstructured decision making process. In that case, it's hard to reason the user's needs. We can implement other responsive features considering the user's evolution in his or her needs over a relatively long period. This type of feature will require a longitudinal evaluation of MSLTRAIN.

We need further research implementing the concept of responsiveness to other application domains, especially more unstructured, perplexity-oriented, and strategic domains. In these domains, managers' needs change frequently depending on the situation. I think the only way to support managers in these domains is to make management tools responsive. Otherwise, designers should

redesign the system whenever the managers' needs change. The concept of responsiveness will have greater effect on these domains.

7.4 Summary - Major Points of the Research

I can summarize major points of this research as follows:

- This research, for the first time, defined and operationalized the concept of responsiveness in management tools.

- This research provided a way to implement the concept of responsiveness into a management tool.

- This research demonstrated we can really implement the concept of responsiveness into an existing management tool.

- This research confirmed responsiveness in a management tool really improves the match between the tool and managers using it, to a limited degree.

Bibliography

Ackoff, Russell L. (1967). Management Misinformation Systems. *Management Science*, 14(4), 147-158.

Ahituv, Niv (1980). A Systematic Approach Toward Assessing the Value of an Information System. *MIS Quarterly*, 61-75.

Alavi, Maryam, & Henderson, John C. (1981). An Evolutionary Strategy for Implementing a Decision Support System. *Management Science* , 27(11), 1309-1323.

Allen, James F., & Perrault, C. Raymond (1980). Analyzing Intention in Utterances. *Artificial Intelligence*, 15, 143-179.

Alty, J. L. (1984). The Application of Path Algebras to Interactive Dialogue Design. *Behavior and Information Technology*, 3(2), 119-132.

Alty, J. L., & McKell, P. (1986). Application Modeling in a User Interface Management System. In: Harrison, M. D., & Monk, A. F. (eds.), *People and Computers: Designing for Usability: Proceedings of the Second Conference of the British Computer Society Human Computer Interaction Specialist Group*, Cambridge: Cambridge Uinv. Press.

Ambardar, A. Kak (1988). User-Computer Interaction: Analysis of Individual Differences and Information Retrieval. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 1246-1249.

Anthony, Robert N. (1965). *Planning and Control Systems: A Framework for Analysis*, Division of Research, Graduate School of Business Administration, Harvard University.

Bailey, James E., & Pearson, Sammy W. (1983). Development of a Tool for Measuring and Analyzing Computer User Satisfaction. *Management Science*, 29(5), 530-544.

Bair, James H. (1984). Designing the VDT Interface for Human-Computer Productivity. In: Bennett, John, Case, Donald, Sandelin, Jon, & Smith, Michael (eds.), *Visual Display Terminals: Usability Issues and Health Concerns*, Englewood Cliffs, NJ: Prentice-Hall.

Balzert, Hemut (1987). A Blackboard Architecture for the Realization of Software-Ergonomic Demands. In: Bullinger, H. J, & Shackel, B. (eds.), *Human-Computer Interaction: INTERACT '87*, New York: North-Holland.

Balzert, Hemut, & Lutze, Rainer (1987). Information and Consultation Systems - A New Dimension of User Support. In: Bullinger, H. J, & Shackel, B. (eds.),

Human-Computer Interaction: INTERACT '87, New York: North-Holland.

Bariff, M. L., & Lusk, E. J. (1977). Cognitive and Personality Tests of Management Information Systems. *Management Science*, 23(8), 820-829.

Barkin, Stephen R., & Dickson, Gary W. (1977). An Investigation of Information System Utilization. *Information & Management*, 1, 35-45.

Baroudi, Jack J., & Orlikowski, Wanda J. (1988). A Short-Form Measure of User Information Satisfaction: A Psychometric Evaluation and Notes on Use. *Journal of Management Information Systems*, 4(1), 44-59.

Barr, Avron, & Feigenbaum, Edward A. (eds.) (1981). *The Handbook of Artificial Intelligence*, Vol. I, II, III, Willian Kaufmann, Inc.

Baum, L., Kaiser, K, Blevins, D., Miller, B., Jagannathan, V., & Anderson, M. (1988). Adapting the Blackboard Model for Cockpit Information Management. Boeing Technical Report.

Bechtel, William (1985). Realism, Instrumentalism, and the Intentional Stance. *Cognitive Science*, 9, 473-497.

Belker, Loren B. (1981). *The Successful Secretary: You, Your Boss, and the Job*, New York: AMACOM.

Belkin, N. J., Hennings, R. D., & Seeger, T. (1984). Simulation of a Distributed Expert-based Information Provision Mechanism. *Information Technology*, 3(3), 122-141.

Benbasat, Izak, & Taylor, Ronald N. (1978). The Impact of Cognitive Styles on Information System Design. *MIS Quarterly*, June, 43-54.

Benbasat, Izak, Dexter, A. S., & Masulis, P. S. (1981). An Experimental Study of the Human/Computer Interface. *Communications of the ACM*, 24(11), 752-762.

Benbasat, Izak, & Taylor, Ronald N. (1982). Behavioral Aspects of Information Processing for the Design of Management Information Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-12(4), 439-450.

Benbasat, Izak, & Wand, Yair (1984). A Structured Approach to Designing Human-Computer Dialogues. *International Journal of Man-Machine Studies*, 21, 105-126.

Bennett, J. L. (1979). The Commercial Impact of Usability in Interactive Systems. In: Infotech State of the Art Report, *Man/Computer Communication*, Vol. 2, Englnad: Infotech.

Bennett, John L. (1984). Managing to Meet Usability Requirements: Establishing and Meeting Software Development Goals. In: Bennett, John, Case, Donald, Sandelin, Jon, & Smith, Michael (eds.), *Visual Display Terminals: Usability Issues and Health Concerns*, Englewood Cliffs, NJ: Prentice-Hall.

Benyon, David (1984). MONITOR. A Self-Adaptive User Interface. In: Shackel, B. (eds.), *Human-Computer Interaction - INTERACT '84*, Amsterdam, North-Holland.

Benyon, David, Innocent, Peter, & Murray, Dianne (1987). System Adaptivity and the Modeling of Stereotypes. In: Bullinger, H. J, & Shackel, B. (eds.), *Human-Computer Interaction: INTERACT '87*, New York: North-Holland.

Blaylock, Bruce K., & Rees, Loren P. (1984). Cognitive Style and the Usefulness of Information. *Decision Sciences*, 15, 74-90.

Blumenthal, Sherman C. (1969). *Management Information Systems - A Framework for Planning and Development*, New Jersey: Prentice Hall.

Boguraev, Branimir (1985). User Modelling in Cooperative Natural Language Front Ends. In: Gilbert, G. Nigel, & Heath, Christian, *Social Action and Artificial Intelligence*, England: Gower.

Booher, Dianna (1985). *The New Secretary: How to Handle People As Well As You Handle Paper*, New York: Facts On File Pub.

Branscomb, L. M., & Thomas, J. C. (1984). Ease of Use: A System Design Challenge. *IBM Systems Journal*, 23(3), 224-235.

Brecht, Barbara, & Jones, Marlene (1988). Student Models: the Genetic Graph Approach. *International Journal of Man-Machine Studies*, 28, 483-504.

Briggs, Pamela (1988). What We Know and What We Need to Know: the User Model versus the User's Model in Human-Computer Interaction. *Behaviour and Information Technology*, 7(4), 431-442.

Brinberg, David, & McGrath, Joseph E. (1985). *Validity and the Research Process*, Newbury Park: SAGE.

Brooks, H. M., & Belkin, N. J. (1983). Using Discourse Analysis for the Design of Information Retrieval Mechanisms. *Proceedings of the Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 31-47.

Bumbaca, Federico (1988). Intelligent Computer-Assisted Instruction: A Theoretical Framework. *International Journal of Man-Machine Studies*, 29, 227-255.

Bundy, Alan (1984). Intelligent Front Ends. In: Bramer, M. A. (eds.) *Research and Development in Expert Systems: Proceedings of the Fourth Technical Conference of the British Computer Society Specialist Group on Expert Systems*, Cambridge: Cambridge Univ. Press.

Buxton, W., Lamb, M. R., Sherman, D., & Smith, K. C. (1983). Towards a Comprehensive User Interface Management System. *Computer Graphics*, 17(3), 35-42.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*, Hillside, NJ: L. Erlbaum Associates.

Carroll, John M., & McKendree, Jean (1987). Interface Design Issues for Advice-Giving Expert Systems. *Communications of the ACM*, 30(1), 14-31.

Cats-Baril,William L., & Huber, George P. (1987). Decision Support Systems for Ill-Structured Problems: An Empirical Study. *Decision Sciences*, 18(3), 350-372.

Charniak, Eugene, & McDermott, Drew (1985). *Introduction to Artificial Intelligence*, Reading, MA : Addison-Wesley Pub. Company.

Chignell, M. H., & Hancock, P. A. (1989). An Introduction to Intelligent Interfaces. In: Hancock, P. A., & Chignell, M. H. (eds.), *Intelligent Interfaces: Theory, Research and Design*, New York: North Holland.

Chin, John P., Diehl, Virginia A., & Norman, Kent L. (1988). Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface. *Human Factors in Computing Systems: Proceedings of CHI '88 Conference*, 213-218.

Clarke, A. A. (1986). A Three-Level Human-Computer Interface Model. *International Journal of Man-Machine Studies*, 24, 503-517.

Clowes, I., Cole, I., Arshad, F., Hopkins, C., & Hockley, A. (1985). User Modelling Techniques for Interactive Systems. In: Johnson, Peter, & Cook, Stephen (eds.), *People and Computers: Designing the Interface - Proceedings of the Conference of the British Computer Society Human Computer Interaction Specialist Group*, Cambridge, Cambridge Univ. Press.

Conrad, Michael (1983). *Adaptability: The Significance of Variability from Molecule to Ecosystem*, New York: Plenum Press.

Cox, Brad J. (1986). *Object-Oriented Programming*, Reading, MA: Addison-Wesley Pub. Company.

Crocker, Linda, & Algina, James (1986). *Introduction to Classical & Modern Test Theory*, New York: Holt, Rinehart and Winston.

Croft, W. Bruce (1984). The Role of Context and Adaptation in User Interfaces. *International Journal of Man-Machine Studies*, 21, 283-292.

Daniels, Penny J. (1986). The User Modelling of an Intelligent Interface for Document Retrieval Systems. In: Brooks, B. C. (eds.), *Intelligent Information Systems for the Information Society*, New York: North-Holland.

Davenport, Colin, & Weir, George (1986). Plan Recognition for Intelligent Advice and Monitoring. In: Harrison, M. D., & Monk, A. F. (eds.), *People and Computers: Designing for Usability: Proceedings of the Second Conference of the British Computer Society Human Computer Interaction Specialist Group*, Cambridge: Cambridge Uinv. Press.

De, Suranjan (1986). Providing Effective Decision Support: Modeling Users and Their Requirements. *Decision Support Systems*, 2, 309-319.

De Greene, Kenyon B. (1982). *The Adaptive Organization: Anticipation and Management of Crisis*, new York: John Wiley & Sons.

Dehning, Wadtrand, Essig, Heidurn, & Maass, Susanne (1981). *The Adaptation of Virtual Man-Computer Interfaces to User Requirements in Dialogues*, New York: Springer-Verlag.

Del Galdo, Elisa M., Williges, Robert C., & Williges, Beverley H. (1986). An Evaluation of Critical Incidents for Software Documentation Design. *Proceedings of the Human Factors Society*, 30th, 19-23.

Delgrande, James P. (1987). A Formal Approach to Learning from Examples. *International Journal of Man-Machine Studies*, 26, 123-141.

Deming, W. Edwards (1986). *Out of Crisis*, Cambridge: MIT, Center for Advanced Engineering Study.

Desmarais, Michel C., Larochelle, Serge, & Giroux, Luc (1987). The Diagnosis of User Strategies. In: Bullinger, H. J, & Shackel, B. (eds.), *Human-Computer Interaction: INTERACT '87*, New York: North-Holland.

Deutsch, Michael S. (1982). *Software Verification and Validation: Realistic Project Approaches*, Englewood Cliffs, NJ: Prentice-Hall.

De Waele, Martin (1978). Managerial Style and the Design of Decision Aids. *Omega*, 6(1), 5-13.

Dickson, Gary W., Senn, James A., & Chervany, Norman L. (1977). Research in Management Information Systems: The Minnesota Experiments. *Management Science*, 23(9), 913-923.

Dix, Alan, & Runciman, Colin (1985). Abstract Models of Interactive Systems. In: Johnson, Peter, & Cook, Stephen (eds.), *People and Computers: Designing the Interface - Proceedings of the Conference of the British Computer Society Human Computer Interaction Specialist Group*, Cambridge, Cambridge Univ. Press.

Driver, Michael J., & Mock, Theodore J. (1975). Human Information Processing, Decision Style Theory, and Accounting Information Systems. *Accounting Review*, July, 490-508.

Drucker, Peter F. (1967). *The Effective Executive*, Pan Books Ltd.

Dunn-Rankin, Peter (1983), *Scaling Methods*, Hillsdale, NJ: Lawrence Erlbaum Associates.

Dzida, W., Herda, S., & Itzfeldt, W. D. (1978). User-Perceived Quality of Interactive Systems. *IEEE Transactions on Software Engineering*, SE-4(4), 270-276.

Edmonds, Ernest A. (1981). Adaptive Man-Computer Interfaces. in Coombs, M. J., & Alty, J. L. (eds.), *Computing Skills and the User Interface*, New York: Academic Press.

Ein-Dor, Philip, & Segev, Eli (1981). *A Paradigm for Management Information Systems*, New York: Praeger.

Erman, L. D., Hayes-Roth, F., Lesser, V. R., & Reddy, D. R. (1980). The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Computing Surveys*, 12(12), 213-253.

Fischer, Gerhard, & Stevens, Curt (1987). Volunteering Information - Enhancing the Communication Capabilities of Knowledge-Based Systems. In: Bullinger, H. J, & Shackel, B. (eds.), *Human-Computer Interaction: INTERACT '87*, New York: North-Holland.

Fitter, Mike, & Sime, Max (1980). Creating Responsive Computers: Responsibility and Shared Decision-Making. In: Smith, H. T., & Green, T. R. G. (eds.), *Human Interaction with Computers*, New York: Academic Press.

Foley, James, Gibbs, Christina, Kim, Won Chul, & Kovacevic, Srdjan (1988). A Knowledge-Based User Interface Management System. *Human Factors in Computing Systems: Proceedings of CHI '88 Conference*, 67-72.

Forrester, Jay W. (1961). *Industrial Dynamics*, Cambridge, MA: MIT Press.

Fowler, C. J. H., & Murray, D. (1987). Gender and Cognitive Style Differences at the Human Computer Interface. In: Bullinger, H. J, & Shackel, B. (eds.), *Human-Computer Interaction: INTERACT '87*, New York: North-Holland.

Gaines, B. R. (1979). Man/Computer Communication: An Overview. In: Infotech State of the Art Report, *Man/Computer Communication*, Vol. 2, England: Infotech.

Gargan, Robert A., Sullivan, Joseph W., & Tyler, Sherman W. (1988). Multimodal Response Planning: An Adaptive Rule Based Approach. *Human Factors in Computing Systems: Proceedings of CHI '88 Conference*, 229-234.

Gerring, Phillip E., & Shortliffe, Edward H. (1982). The Interviewer/ Reasoner Model: An Approach to Improving System Responsiveness in Interactive AI Systems. *AI Magazine*, Fall, 24-27.

Gershman, Anatole (1981). Figuring Out What the User Wants - Steps toward an Automatic Yellow Page Assistant. *Proceedings of IJCAI '81*, 423-425.

Gilbert, G. Nigel (1987). Cognitive and Social Models of the User. In: Bullinger, H. J, & Shackel, B. (eds.), *Human-Computer Interaction: INTERACT '87*, New York: North-Holland.

Glaser, Barney G., & Strauss, Anselm L. (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*, New York: Aldine Pub. Company.

Glorioso, Robert M., & Osorio, Fernando C. C. (1980). *Engineering Intelligent Systems*, U.S.A., Digital Press.

Goldberg, Adele, & Robson, David (1983). *Smalltalk-80: The Language and its Implementation*, Reading, MA: Addison-Wesley Pub. Company.

Goldstein, I. P. (1982). The Genetic Graph: A Representation for the Evolution of Procedural Knowledge. In: Sleeman, D., & Brown, J. S. (eds.), *Intelligent Tutoring Systems*, New York: Academic Press.

Good, M.D., Whiteside, J. A., Wixon, D. R., & Jones, S. J. (1984). Building a User-Derived Interface. *Communications of the ACM*, 27(10), 1032-1043.

Goodwin, Nancy C. (1987).Functionality and Usability. *Communications of the ACM*, 30(3), 229-233.

Gorry, G. Anthony, & Scott Morton, M. S. (1971). A Framework for Management Information Systems. *Sloan Management Review*, Fall, 55-70.

Gould, John D., & Lewis, Clayton (1985). Designing for Usability: Key Principles and What Designers Think. *Communications of the ACM*, 28(3), 300-311.

Greenberg, Saul, & Witten, Ian H. (1985). Adaptive Personalized Interfaces - A Question of Viability. *Behavior and Information Technology*, 4(1), 31-45.

Grikscheit, Gary M., Cash, Harold C., & Crissy, W. J. E. (1981). *Handbook of Selling: Psychological, Managerial, and Marketing Bases*, New York: John Wiley & Sons.

Harmond, Paul, & King, David (1985). *Expert Systems: Artificial Intelligence in Business*, New York: John Wiely & Sons.

Hayes, Eugene B., & Reddy, Raj (1981). Breaking the Man-Machine Communication Barrier. *Computer*, March, 19-30.

Hayes-Roth, Barbara, & Hayes-Roth, Frederick (1979). A Cognitive Model of Planning. *Cognitive Science*, 3, 275-310.

Hayes-Roth, Barbara (1985). A Blackboard Architecture for Control. *Artificial Intelligence*, 26, 251-321.

Hayes-Roth, Frederick, Waterman, Donald A., & Lenat, Douglas B. (eds.) (1983). *Building Expert Systems*, Reading, MA: Addison-Wesley Pub. Company.

Henderson, John C., & Nutt, Paul C. (1980). The Influence of Decision Style on Decision Making Behavior. *Management Science*, 26(4), 371-386.

Hiltz, Starr Roxanne, & Johnson, Kenneth (1990). User Satisfaction with Computer-Mediated Communication Systems. *Management Science*, 36(6), 739-764.

Holland, John H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, Ann Arbor: The University of Michigan Press.

Hollnagel, Erik (1986). Information and Reasoning in Intelligent Decision Support Systems. *International Journal of Man-Machine Studies*, 25, 33-51.

Holsapple, C. W. (1987). Adapting Demons to Knowledge Management Environments. *Decision Support Systems*, 3, 289-298.

Huber, George P. (1983). Cognitive Style as a Basis for MIS and DSS Designs: Much Ado about Nothing ? *Management Science*, 29(5), 567-579.

Innocent, P. R. (1982). Towards Self-Adaptive Interface Systems. *International Journal of Man-Machine Studies*, 16, 287-299.

Ives, Blake, Olson, Margrethe H., & Baroudi, Jack J. (1983). The Measurement of User Information Satisfaction. *Communications of the ACM*, 26(10), 785-793.

Jagannathan, V., Dodhiawala, R. T., & Baum, L. S. (1988). Boeing Blackboard System: The Erasmus Version. *International Journal of Intelligent Systems*, 3, 281-293.

Johnston, William F. (1988). *Responsiveness in American Schools Overseas: Discrepancies Between Parental Expectations and School Performance*, Unpublished Ph.D. Dissertation, Virginia Tech.

Kammersgaard, John (1988). Four Different perspectives on Human-Computer Interaction. *International Journal of Man-Machine Studies*, 28, 343-362.

Kantorowitz, Eliezer, & Sudarsky, Oded (1989). The Adaptable User Interface. *Communications of the ACM*, 32(11), 1352-1358.

Kass, Robert, & Finin, Tim (1989). The Role of User Models in Cooperative Interactive Systems. *International Journal of Intelligent Systems*, 4, 81-112.

Katzan, Harry (1984). *Management Support Systems*, New York: Van Nostrand Reinhold Com.

Kaufmann, Roger, & English, Fenwick W. (1979). *Needs Assessment: Concept and Application*, Englewood Cliffs, NJ: Educational Technology Pub.

Keen, Peter G. W., & Scott Morton, M. S. (1978). *Decision Support Systems: An Organizational Perspective*, Reading, MA: Addison-Wesley Pub. Company.

Keen, Peter G. W. (1980). Adaptive Design for Decision Support Systems. *Data Base*, Fall, 15-25.

Kidd, A. L. (1985). What Do Users Ask? - Some Thoughts on Diagnostic Advice. In:

Merry, M. (eds.), *Expert Systems 85: Proceedings of the Fourth Technical Conference of the British Computer Society Specialist Group on Expert Systems*, 9-19, Cambridge: Cambridge Univ. Press.

Kilmann, Ralph L., & Mitroff, Ian I. (1976). Qualitative versus Quantitative Analysis for Management Science: Different Forms for Different Psychological Types. *Interfaces*, 6(2), 17-27.

Klensin, John C. (1982). Short-term Friendly and Long-term Hostile. *SIGSOC Bulletin*, Proceedings of '81 Conference, 105-110.

Kurstedt, H. A. (1985a). Series of working articles describing the Management System Model. Article #1: The industrial engineer's systematic approach to management. Article #2: " The Management System Model helps your tools work for you. Article #3: "Managing a management system reduces to concentrating in the interfaces between components. Management Systems Laboratories, Virginia Tech, Blacksburg, VA 24061.

Kurstedt, Harold A., Jr. (1985b). Responsive Decision Support Systems: A Broad View Illustrates When to Include Expert Systems. *Paper presented at the 2nd Euro Mini Conference on Expert Systems and Artificial Intelligence in Decision Support Systems*, Luntern, The Netherlands, November.

Kurstedt, Harold A. (1986). *So, You Have a Lot of Hardware and Software Your People Don't Know How to Use*, Worksheet, Management Systems Laboratories, Virgnia Tech.

Kurstedt, H. A., Lee, K. S., Mendes, P. M., & Berube, D. S. (1988a). The Responsive System: A New Challenge for AI. *Proceedings of the First International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, (Tullahoma, Tennessee; June 2-3), 177-184.

Kurstedt, H. A., Lee, K. S., Mendes, P. M., & Jones, M. R. (1988b). The Responsive System Operationalizes the Management System Model Concept using Distributed AI Techniques. *Proceedings of the Conference on the Impact of Artificial Intelligence on Business and Industry*, (Denton, Texas; Oct. 9-11), 63-73.

Lee, Kwang S., Kurstedt, Harold A., & Middleman, Rouis I. (1989). A Framework Indicates Direction in Future Intelligent System Development. *Proceedings of the 25th Southeastern TIMS Annual Meeting*, (Myrtle Beach, Oct. 5-6), 303-306.

Leedy, Paul D. (1980). *The Practical Research: Planning and Design*, New York: Macmillan.

Lesser, Victor R., & Corkill, Daniel D. (1981). Functionally Accurate, Cooperative Distributed Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1), 81-96.

Liang, Ting-Peng (1987). Design of a Self-evolving Decision Support System. *Journal of Management Information System*, 4(1), 59-82.

Lucas, Henry C., Clowes, Kenneth W., & Kaplan, Robert B. (1974). Frameworks for Information Systems. *INFOR*, 12(3), 245-260.

Lucas, Henry C. (1978). The Evolution of an Information System: From Key-Man to Every Person. *Sloan Management Review*, Winter, 39-52.

Lucas, Henry C., & Nielson, Norman R. (1980). The Impact of the Mode of Information Presentation on Learning and Performance. *Management Science*, 26(10), 982-993.

Luconi, Fred L., Malone, Thomas W., & Scott Morton, M. S. (1986). Expert Systems: The Next Challenge for Managers. *Sloan Management Review*, Summer, 3-14.

Lusk, Edward J., & Kersnick, Michael (1979). The Effect of Cognitive Style and Report Format on Task Performance: The MIS Design Consequences. *Management Science*, 25(8), 787-798.

Macintosh, Norman B. (1981). A Contextual Model of Information Systems. *Accounting, Organization, and Society*, 6(1), 39-53.

Macmillan, Stuart A., & Sleeman, Derek H. (1987). An Architecture for a Self-improving Instructional Planner for Intelligent Tutoring Systems. *Computational Intelligence*, 3, 17-27.

Mallak, Larry A. (1986). *Applying the Management System Model to a Federal Government Organization*, M.S. Thesis, Virginia Tech.

Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A., & Cohen, M. D. (1987). Intelligent Information-Sharing Systems. *Communications of the ACM*, 30(5), 390-402.

Management Systems Laboratories (1986). *Research and Development of Models and Instrument to Define, Measure, and Improve Shared Information Processing Within Government Oversight Agencies*, Proposal to DOE.

Manheim, Marvin L. (1988). An Architecture for Active DSS. *Proceedings of the 21st Annual Hawaii International Conference on System Sciences*, Vol. III, 356-365.

Manheim, Marvin L. (1989). Issues in Design of Symbiotic DSS. *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, Vol. III, 14-23.

Mann, Robert I., Hugh J. Watson, and Paul H. Cheney (1986). Accommodating Cognitive Style through DSS Hardware and Software. *Proceedings of Annual Hawaii International Conference on System Science*, 19th, 627-636.

Markus, M. Lynne, & Robey, Daniel (1983). The Organizational Validity of Management Information Systems. *Human Relations*, 36(3), 203-226.

Martin, Charles F. (1988). *User-Centered Requirements Analysis*, Englewood Cliffs, NJ: Prentice-Hall.

Martin, Merle P. (1986). The Human Connection in Systems Design. *Journal of Systems Management*, 18(10), 6-29.

Maskery, H. S. (1984). Adaptive Interfaces for Naive Users - An Experimental Study. In: Shackel, B. (eds.), *Human-Computer Interaction - INTERACT '84*, Amsterdam, North-Holland.

McCalla, Gordon I., Bunt, Richard B., & Harms, Janelle J. (1986). The Design of the SCENT Automated Advisor. *Computational Intelligence*, 2, 76-92.

McCauley, Clark, Stitt, Christpher L., & Segal, Mary (1980). Stereotyping: From Prejudice to Prediction. *Psychological Bulletin*, 87(1), 195-208.

McKenney, James L., & Keen, Peter G. W. (1974). How Managers' Mind Work. *Harvard Business Review*, 52(3), 79-90.

McKeown, Hathleen R., Wish, Myron, & Matthews, Kevin (1985). Tailoring Explanations for the User. *Proceedings of IJCAI '85* , 794-798.

Michalski, Ryzard S., Carbonell, Jaime G., & Mitchell, Tom M. (eds.) (1983). *Machine Learning: An Artificial Intelligence Approach*, Palo Alto, CA: Tioga Pub. Company.

Miller, James R., Hill, William C., McKendree, Jean, & Masson, Michael E. J. (1987). The Role of the System Image in Intelligent User Assistance. In: Bullinger, H. J, & Shackel, B. (eds.), *Human-Computer Interaction: INTERACT '87*, New York: North-Holland.

Miller, Lance A., & Thomas, John C. (1977). Behavioral Issues in the Use of Interactive Systems. *International Journal of Man-Machine Studies*, 9, 509-536.

Mintzberg, Henry, Raisinghani, Duru, & Theoret, Andre (1976). The Structure of "Unstructured" Decision Processes. *Administrative Science Quarterly*, 21, Sep., 246-275.

Mintzberg, Henry (1980). Structure in 5's: A Synthesis of the Research on Organization

Design. *Management Science*, 26(3), 322-341.

Moran, Thomas P. (1981). The Command Language Grammar: A Representation for the User Interface of Interactive Computer Systems. *International Journal of Man-Machine Studies*, 15, 3-50.

Murdick, Robert G., & Munson, John C. (1986). *MIS Concepts and Design*, Englewood Cliffs, NJ: Prentice-Hall.

Murray, Dianne M. (1987). Embedded User Models. In: Bullinger, H. J, & Shackel, B. (eds.), *Human-Computer Interaction: INTERACT '87*, New York: North-Holland.

Murray, Malinda (1980). *Fundamentals of Nursing*, Englewood Cliffs, NJ: Prentice-Hall.

Myers, Isabel Briggs (1980). *Gifts Differing*, Palo Alto, CA: Consulting Psychologists Press.

Naumann, Justus D., & Jenkins, A. Milton (1982). Prototyping: The New Paradigm for Systems Development. *MIS Quarterly*, 6(3), 29-44.

Nickerson, Raymond S. (1981). Why Interactive Computer Systems Are Sometime Not Used by People Who Might Benefit from Them. *International Journal of Man-Machine Studies*, 15, 469-483.

Nielsen, Jakob (1986). A Virtual Protocol Model for Computer-Human Interaction. *International Journal of Man-Machine Studies*, 24, 301-312.

Nii, H. Penny (1986). 1) Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architecture. *AI Magazine*, 7(2), 38-53. 2) Blackboard Systems: Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective. *AI Magazine*, 7(3), 82-106.

Norman, Donald A. (1983). Some Observations on Mental Models. In: Gentner, Dedre, & Stevens, Albert L. (eds), *Mental Models*, Hillside, NJ: Lawrence Erlbaum Asso.

Norman, Donald A. (1984). Stages and Levels in Human-Machine Interaction. *International Journal of Man-Machine Studies*, 21, 365-375.

Norman, Donald A., & Draper, Stephen W.(eds.) (1986). *User Centered System Design*, Hillside, NJ: Lawrence Erlbaum Assoc.

Nosek, John T. (1984). DSS and Evolutionary Development. *Proceedings of Annual Hawaii International Conference on System Science*, 17th, 526-537.

Nunnally, Jum C. (1967). *Psychometric Theory*, New York: McGraw-Hill.

O'Leary, Daniel E. (1987). Validation of Expert Systems - With Applications to Auditing and Accounting Expert Systems. *Decision Sciences*, 18(3), 468-486.

Olsen, D. R., Buxton, W., Ehrich, R., Kasik, D. J., Rhyne, J. R., & Sibert, J. (1984). A Context for User Interface Management. *IEEE Computer Graphics and Application*, December, 33-42.

Peachery, Darwyn R., & McCalla, Gordon I. (1986). Using Planning Techniques in Intelligent Tutoring Systems. *International Journal of Man-Machine Studies*, 24, 77-98.

Penniman, W. D., & Dominick, W. D. (1980). Monitoring and Evaluation of On-line Information System Usage. *Information Processing & Management*, 16, 17-35.

Peterson, Richard E. (1977). The Components of Information. *Interfaces*, 7(4), 87-91.

Pliskin, Nava, & Shoval, Peretz (1987). End-user Prototyping: Sophisticated Users Supporting System Development. *Data Base*, Summer, 7-17.

Potosnak, K. M. (1984). Choice of Interface Mode by Empirical Grouping of Computer Users. In: Shackel, B. (eds.), *Human-Computer Interaction -. INTERACT '84*, Amsterdam, North-Holland.

Quinn, Lisa, & Russell, Daniel M. (1986). Intelligent Interfaces: User Models and Planners. *Human Factors in Computing Systems: Proceedings of CHI '86 Conference*, 314-320.

Raghavan, Sridhar A., & Chand, Donald R. (1989). Exploring Active Decision Support: The JANUS Project. *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, Vol. III, 33-45.

Ralston, Anthony, & Reilly, Edwin D. (eds.) (1983). *Encyclopedia of Computer Science and Engineering*, New York: Van Nostrand Reinhold Co.

Ramaprasad, Arkalgud, & Mitroff, Ian I. (1984). On Formulating Strategic Problems. *Academy of Management Review*, 9(4), 597-605.

Ramaprasad, Arkagud (1987). Cognitive Process as a Basis for MIS and DSS Design. *Management Science*, 33(2,), 139-148.

Reimann, Bernard C., & Waren, Allan D. (1985). User-oriented Criteria for the Selection of DSS Software. *Communications of the ACM*, 28(2), 166-179.

Remus, William (1984). An Empirical Investigation of the Impact of Graphical and Tabular Data presentations on Decision Making. *Management Science*, 30(5), 533-542.

Remus, William E., & Kottemann, Jeffrey E. (1986). Toward Intelligent Decision Support Systems: An Artificially Intelligent Statistician. *MIS Quarterly*, 10(4), 403-418.

Rich, Elaine (1979). *Building and Exploiting User Models*, Ph.D. Dissertation, Carnegie-Mellon University.

Rich, Elaine (1983). Users are Individuals: Individualizing User Models. *International Journal of Man-Machine Studies*, 18, 199-214.

Rissland, Edwina L. (1984). Ingredients of Intelligent User Interfaces. *International Journal of Man-Machine Studies*, 21, 377-388.

Roach, J., & Wilding, M. (1985). Adapting to Individual Users: The User Trainable Interface. *Proceedings of the International Conference on Cybernetics and Society*, 228-235.

Roberts, Franklin C. (1984). An Overview of Intelligent CAI Systems. *Peabody Journal of Education*, 62, 40-51.

Robey, Daniel, & Taggart, William (1981). Measuring Managers' Mind: The Assessment of Style in Human Information Processing. *Academy of Management Review*, 6(3), 375-383.

Robey, Daniel (1983). Cognitive Style and DSS Design: A Comment on Huber's Paper. *Management Science*, 29(5), 580-582.

Rohr, Gabriele, & Tauber, Michael J. (1984). Representational Frameworks and Models for Human-Computer Interfaces. In: Van Der Veer, G. C., Tauber, M. J., Green, T. R. G., & Gorny, P. *Readings on Cognitive Ergonomics - Mind and Computers: Proceedings of the 2nd European Conference*, New York: Springer-Verlag.

Rosen, Robert (1985). *Anticipatory Systems: Philosophical Mathematical and Methodological Foundations*, Oxford: Pergamon Press.

Rouse, William B. (1984). Design and Evaluation of Computer-Based Decision Support Systems. In: Salvendy, G. (eds.), *Human-Computer Interaction: Proceedings of the First U.S.A.-Japan Conference on Human-Computer Interaction*, New York: Elsevier, 229-246.

Rouse, William B., Geddes, Norman D., & Curry, Rendwick E. (1987). An Architecture for Intelligent Interfaces: Outline of an Approach to Supporting Operators of Complex Systems. *Human Computer Interaction*, 3, 87-122.

Rubin, Kenneth S., Mitchell, Christine M., & Jones, Patricia (1987). Using a Black-

board Architecture for Dynamic Intent Inferencing. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 1150-1153.

Rubinstein, Richard, & Hersh, Harry (1984). *The Human Factor: Designing Computer Systems for People*, U.S.A.: Digital Press.

Rushinek, Avi, & Rushinek, Sara F. (1986). What Makes Users Happy ? *Communications of the ACM*, 29(7), 594-598.

Sage, Andrew P. (1981). Behavioral and Organizational Considerations in the Design of Information Systems and Processes for Planning and Decision Support. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(9), 640-678.

Saja, Allan D. (1985). The Cognitive Model: An Approach to Designing the Human-Computer Interface. *SIGCHI Bulletin*, 16(3), 36-40.

Sauter, Vicki L., & Schofer, Joseph L. (1988). Evolutionary Development of Decision Support Systems: Important Issues for Early Phases of Design. *Journal of Management Information Systems*, 4(4), 77-92.

Schmidt, C. F., Sridharan, N. S., & Goodson, J. L. (1978). The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence. *Artificial Intelligence*, 11, 45-83.

Schneider, Michael L., & Thomas, John C. (1983). The Humanization of Computer Interfaces. *Communications of the ACM*, 26(4), 252-253.

Schneiderman, Ben (1987). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Reading, MA: Addison-Wesley Pub. Company.

Scriven, Michael (1967). The Methodology of Evaluation. In: Tyler, Ralph W., Gagne, Robert M., & Scriven, Michael, *Perspectives of Curriculum Evaluation*, Chicago: Rand McNally.

Scriven, Michael (1981). *Evaluation Thesaurus*, CA: Edgepress.

Shackel, Brian (1984). The Concept of Usability. In: Bennett, John, Case, Donald, Sandelin, Jon, & Smith, Michael (eds.), *Visual Display Terminals: Usability Issues and Health Concerns*, Englewood Cliffs, NJ: Prentice-Hall.

Shulok, Thomas A. (1988). *CHARTMAKER: A "True Consultant" Expert System for Designing Charts*, Unpublished M.S. Thesis, Virginia Tech.

Simon, Herbert A. (1960). *The New Science of Management Decision*, Englewood Cliffs, NJ: Prentice Hall.

Sink, D. Scott (1985). *Productivity Management: Planning, Measurement and Evaluation, Control and Improvement*, New York: John Wiley & Sons.

Sleeman, D., & Brown, J. S. (eds.) (1982). *Intelligent Tutoring Systems*, New York: Academic Press.

Sleeman, D. (1985). UMFE: A User Modelling Front-End Subsystem. *International Journal of Man-Machine studies*, 23, 71-88.

Sleeman, D., Appelt, Doug, Konolige, Kurt, Rich, Elaine, Sridharan, NS, & Swartout, Bill (1985). User Modeling Panel. *Proceedings of IJCAI '85*, 1298-1302.

Smith, J. Jerrame (1985). SUSI - A Smart User-System Interface. In: Johnson, Peter, & Cook, Stephen (eds.), *People and Computers: Designing the Interface - Proceedings of the Conference of the British Computer Society Human Computer Interaction Specialist Group*, Cambridge, Cambridge Univ. Press.

Smith, Reid G., & Davis, Randall (1981). Frameworks for Cooperation in Distributed Problem Solving. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1), 61-70.

Spark Jones, Karen (1986). Issues in User Modeling for Expert Systems. In: Cohn, A. G., & Thomas, J. R. (eds.), *Artificial Intelligence and its Applications*, Chichester: John Wiley & Sons.

Sprague, Ralph H. Jr., & Carlson, Eric D. (1982). *Building Effective Decision Support Systems*, Englewood Cliffs, NJ: Prentice-Hall Inc.

Staddon, J. E. R. (1983). *Adaptive Behavior and Learning*, Cambridge: Cambridge Univ. Press.

Starker, India, & Bolt, Richard A. (1990). A Gaze-Responsive Self-Disclosing Display. *Human Factors in Computing Systems: Proceedings of CHI '90 Conference*, 3-9.

Steele, Guy L. (1984). *Common LISP*, Digital Press.

Stefik, Mark, & Bobrow, Daniel G. (1986). Object-Oriented Programming: Themes and Variations. *AI Magazine*, Jan., 40-62.

Stewart, Tom (1986). Task Fit, Ease-of-Use, and Computer Facilities. In: Niels, Bjorn-Andersen, *Managing Computer Impact*, Norwood, NJ: Alex Pub. Corp.

Stocker, P. M., & Dearnley, P. A. (1974). A Self-Organizing Data Base Management System. In: Klimbie, J. W., & Kofferman, K. L. (eds.), *Data Base Management*, North-Holland.

Stroustrup, Bjarne (1988). What is Object-Oriented Programming? *IEEE Software*, May, 10-20.

Teitelman, Warren, & Masinter, Larry (1981). The Interlisp Programming Environment. *Computer*, 14(4), 25-33.

Teubner, Alan L., & Vaske, Jerry J. (1988). Monitoring Computer User's Behaviour in Office Environments. *Behaviour and Information Technology*, 7(1), 67-78.

Thomas, Christoph, Kellerman, Gert M., & Hein, Hans-Werner (1987). X-AID: An Adaptive and Knowledge-Based Human Computer Interface. *Proceedings of INTERACT '87*, 1075-1080.

Totterdell, Peter, & Cooper, Paul (1986). Design and Evaluation of the AID Adaptive Front-End to Telecom Gold. In: Harrison, M. D., & Monk, A. F. (eds.), *People and Computers: Designing for Usability* , Cambridge: Cambridge Univ. Press.

Totterdell, P. A., Norman, M. A., & Browne, D. P. (1987). Levels of Adaptivity in Interface Design. In: Bullinger, H. J, & Shackel, B. (eds.), *Human-Computer Interaction: INTERACT '87*, New York: North-Holland.

Tracey, William R. (1984). *Designing Training and Development Systems*, New York: AMACOM.

Trevellyan, Robert, & Browne, Dermot P. (1987). A Self-Regulating Adaptive System. *Proceedings of CHI + GI '87 Conference on Human Factors in Computing Systems and Graphics Interface*, 103-107.

Tyler, Sherman W. (1986). *SAUCI: Self-Adaptive User-Computer Interface*, Ph.D. Dissertation, Univ. of Pittsburgh.

Vasarhelyi, Miklos A. (1977). Man-Machine Planning Systems: A Cognitive Style Examination of Interactive Decision Making. *Journal of Accounting Research*, Spring, 138-153.

Veer, G. C. Van Der, Tauber, Michael J., Waern, Yvonne, & Muylwijk, Bert Van (1985). On the Interaction between System and User Characteristics. *Behavior and Information Technology*, 4(4), 289-308.

Vroom, V. H., & Yetton, P. W. (1973). *Leadership and Decision Making*, Pittsburgh, Univ. of Pittsburgh Press.

Watson, Richard T., DeSantis, Gerardine, & Poole, Marshall Scott (1988). Using a GDSS to Facilitate Group Consensus: Some Intended and Unintended Consequenses. *MIS Quarterly*, 12(3), 463-477.

Wedley, William, & Field, Richard H. G. (1984). A Predecision Support System. *Academy of Management Review*, 9(4), 696-703.

Whiteside, John, Bennett, John, & Holtzblatt, Karen (1988). Usability Engineering: Our Experience and Evolution. In: Helander,

Martin (eds.), *Handbook of Human-Computer Interaction*, Amsterdam: North-Holland.

Williges, R. C., Williges, B. H., & Elkerton, J. (1987). Software Interface Design. In: Salvendy, G. (eds.), *Handbook of Human Factors*, New York: Wiley, 1416-1449.

Yaverbaum, Gayle J. (1988). Critical Factors in the User Environment: An Experimental Study of Users, Organizations and Tasks. *MIS Quarterly*, 12(1), 75-88.

Zachary, Wayne (1985). Beyond User-Friendly: Building Decision-Aid Interfaces for Expert End Users. *Proceedings of the International Conference on Cybernetics and Society*, 641-647.

Zeigler, L. Harmon, & Tucker, Harvey J. (1978). *The Quest for Responsive Government: An Introduction to State and Local Politics*, North Scituate, MA: Duxbury Press.

Zissos, Adrian Y., & Witten, Ian H. (1985). User Modelling for a Computer Coach: A Case Study. *International Journal of Man-Machine studies*, 23, 729-750.

Zmud, Robert W. (1979). Individual Differences and MIS Success: A Review of the Empirical Literature. *Management Science*, 25(10), 966-979.

Appendix A. Information Items in MSLTRAIN

This appendix shows examples of information items listed in the "Information Menu" of MSLTRAIN and explains them. The information items are:

- General
  - Busy Months - Whole
  - Package Description
- OP: Organizational Proficiency
  - Whole (Table)
  - By PKG[50] (Table)
  - By PKG (Graph)
- IP: Individual Proficiency
  - Whole (Table)
  - By Person (Table)
- TC: Training Commitment
  - Total (Table)
  - By PKG (Table)

---

[50] "By PKG" means "By Package."

- TS: Training schedule
  - Whole (Table)
  - By Person (Table)
  - By Person (Graph)
- Gap between OP and IP
  - Whole (Table)
  - Whole (Graph)
  - By PKG (Table)
  - By PKG (Graph)
- Gap between TC and TS
  - Whole (Table)
  - Whole (Graph)
  - By PKG (Table)
  - By PKG (Graph)

Users can access an information item by clicking the mouse in the item or by hitting the "INFO" key and then selecting the item with the "RETURN" key.

General; Busy Months - Whole

Example

Individual Busy Months

Name	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
Bill Muraham		X	X				X				X	
Bob Hamilton		X				X					X	
Chris Dalton			X	X			X	X				X
Dana Keating			X					X				X
Fred Kennedy	X	X			X	X				X		
Jane Newmann	X	X	X			X						
Nick Johnson					X			X	X	X		
Paul Moriarty	X			X	X				X			

Explanation

This table shows the months during which people are expected to be too busy to get training. "X" in the table indicates the busy months. It is recommended not to schedule a person during a busy month.

General; Package Description

Example

Description of Designer

Name	Designer
Purpose	Graphics
Required Training Days	Novice: 1 Regular: 3 Expert: 6
Description	Graphics draw/paint program, comes with clipart, use PANTONE color-matching palette.

Explanation

This table provides brief information about a package. "Required Training Days" means days required for a novice, a regular, or an expert session. In the example, an expert session for "Designer" takes 3 days. When you schedule a person for a training session, you should consider the person will be out of the office for the number of training days.

OP: Organizational Proficiency; Whole (Table)

Example

Desired Organizational Proficiency

(Total Number of People = 8)

Package	By Feb			By Jun			By Oct		
	N	R	E	N	R	E	N	R	E
MS Word	3	2	0	3	1	1	4	2	1
Lotus 123	2	2	0	3	3	0	4	2	1
Designer	3	1	0	4	0	1	4	2	1

N: novice R: regular E: expert

Explanation

This table shows milestones in terms of the number of people proficient in given packages. The milestones provide steps to get the final organizational proficiency. In the example, the organization needs three novices, three regulars, and no experts on "Lotus 123" by the end of June 1991.

OP: Organizational Proficiency; By PKG (Table)

Example

Desired Organizational Proficiency of MS Word

(Total Number of People = 8)

Proficiency Level	Current	By Feb	By Jun	By Oct
Novice	1	3	3	4
Regular	1	2	1	2
Expert	0	0	1	1

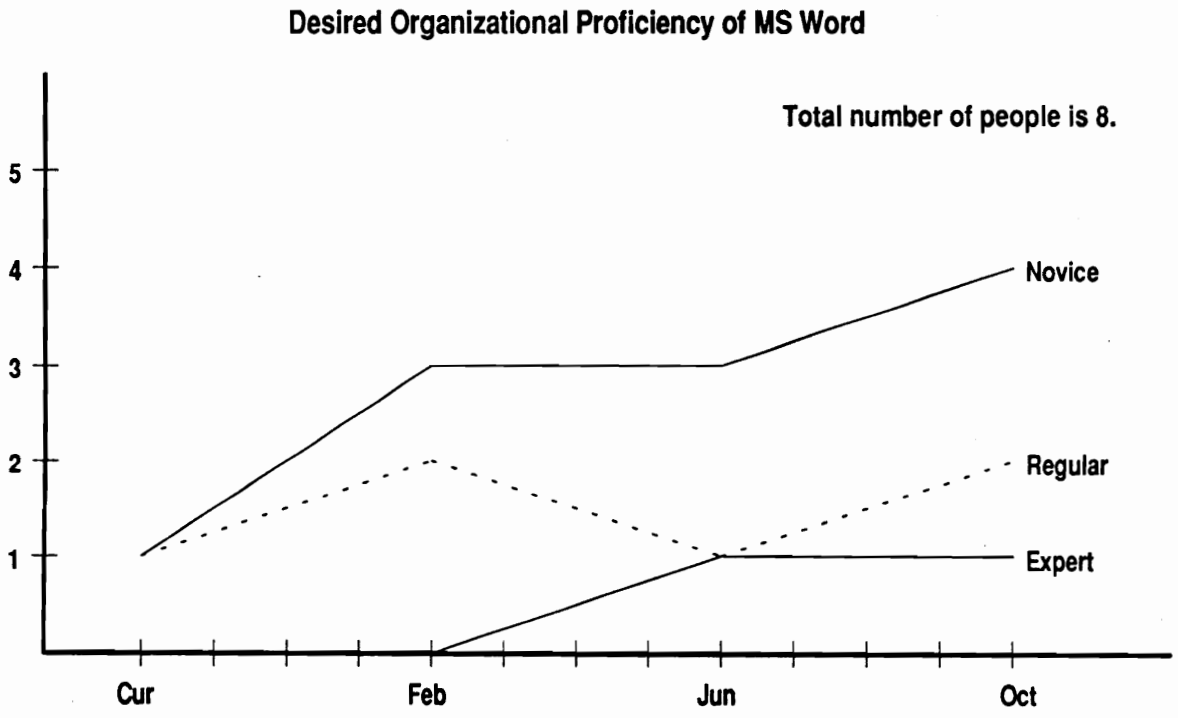
The numbers represent the number of people.

Explanation

This table shows milestones in terms of the number of people proficient in a specific package. The milestones provide steps to get the final organizational proficiency. In the example, the organization needs three novices, two regulars, and no experts on "MS Word" by the end of February 1991.

OP: Organizational Proficiency; By Person (Graph)

Example



Explanation

This graph shows milestones in terms of the number of people proficient in a specific package. The graph indicates the trend of organizational proficiency during the planning horizon. In the example, the organization needs three novices, two regulars, and no expert on "MS Word" by the end of February 1991.

TC: Training Commitment; No. of People (Table)

Example

Required Number of People to Be Trained

(Total Number of People = 8)

Package	Nov - Feb			Mar - Jun			Jul - Oct		
	N	R	E	N	R	E	N	R	E
MS Word	3	1	0	0	0	1	2	1	0
Lotus 123	2	2	0	2	1	0	1	0	1
Designer	3	1	0	1	0	1	2	2	1

N: novice R: regular E: expert

Explanation

This table shows how many people are required to be trained on packages during a period. As the result of the training, the organizational proficiency goal is satisfied. In the example, the organization needs to get two novices and one regular trained during the March to June period on "Lotus 123".

TC: Training Commitment; Total Days (Table)

Example

Expected Training Commitment in Days

Package	Nov-Feb (656)	%	Mar-Jun (672)	%	Jul-Oct (672)	%	Total (2000)	%
MS Word	6	0.9	6	0.9	5	0.7	17	0.8
Lotus 123	8	1.3	5	0.7	7	1.0	20	1.0
Designer	6	0.9	7	1.0	8	1.2	21	1.0
Total	20	3.1	18	2.6	20	2.9	58	2.0

Percentage (%) means the ratio between person-days required for training over total working person-days during a period.

Explanation

This table shows how many person-days are required to satisfy the organizational proficiency. Percentage means the ratio between person-days required for training on a package over total working person-days during a period. In the example, the organization needs to spend 18 person-days, 2.6% of total 672 working person-days (8 people x 84 working days in 4 months), during the March to June period.

TC: Training Commitment; By PKG (Table)

Example

Expected Training Commitment in Days on MS Word

Proficiency Level	Nov-Feb (656)	%	Mar-Jun (672)	%	Jul-Oct (672)	%	Total (2000)	%
Novice	3	0.5	0	0.0	2	0.3	5	0.2
Regular	3	0.5	0	0.0	3	0.4	6	0.3
Expert	0	0.0	6	0.9	0	0.0	6	0.3
Total	6	1.0	6	0.9	5	0.7	17	0.8

Percentage (%) means the ratio between person-days required for training over total working person-days during a period.

Explanation

This table shows how many person-days are required to satisfy the organizational proficiency on a specific package. Percentage means the ratio between person-days required for training on a proficiency level over total working person-days during a period. In the example, the organization needs to spend 22 person-days on MS Word, 3.2% of total 672 working person-days, during the March to June period.

IP: Individual Proficiency; Whole (Table)

Example

Desired Individual Proficiencies

Name	MS Word	Lotus 123	Designer
Bill Muraham	* → R	*	N → E
Bob Hamilton	* → N	* → N	* → N
Chris Dalton	* → N	* → N	* → R
Dana Keating	*	* → N	* → N
Fred Kennedy	* → N	N → E	* → N
Jane Newmann	N → R	* → R	*
Nick Johnson	* → N	N → R	* → N
Paul Moriarty	R → E	* → N	* → R

*: none N: novice R: regular E: expert.

The arrows (→) indicate upgrade.

Explanation

This table shows for each person in your organization the desired proficiency level on each package. The letter where the arrow starts indicates the current proficiency level of the person and the letter where the arrow ends indicates the desired proficiency level of the person. In the example, "Bill Muraham" is desired to be a regular on "MS Word" and an expert on "Designer."

IP: Individual Proficiency; By Person (Table)

Example

Desired Proficiencies of Bill Muraham

Package	Current		Desired
MS Word	None	→	Regular
Lotus 123	None		None
Designer	Novice	→	Expert

The arrows (→) indicate upgrade.

Explanation

This table shows for a specific person in the organization the desired proficiency level on the packages. In the example, "Bill Muraham" is desired to be a regular on "MS Word" and an expert on "Designer."

TS: Training Schedules; Whole (Table)

Example

Training Schedules

Name	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Total
Bill Muraham	W/N	x	x	.	D/R	.	x	.	.	W/R	x	D/E	13
Bob Hamilton	W/N	x		D/N*	.	x	.	.	.	.	x	.	3
Chris Dalton	W/N	D/N	x	x	L/N	D/R	x	x	.	.	.	x	6
Dana Keating	L/N	D/N	x	.	.	.	.	x	.	.	.	x	2
Fred Kennedy	x	x	W/N	D/N	x	x	L/R	.	L/E	x	.	.	11
Jane Newmann	x	x	x	L/N	.	x	L/R	.	.	W/R	.	.	7
Nick Johnson	.	D/N	W/N	.	x	.	.	x	x	x	L/R	.	5
Paul Moriarty	x	L/N	D/N	x	x	D/R	.	W/E	.	.	.	.	11
Total Days	4	4	3	4	4	6	6	6	6	6	3	6	56

Packages; W: MS Word L: Lotus 123 D: Designer Levels; N: novice R: regular E: expert
"x" indicates a busy month for the person to get training.
"*" indicates you scheduled more than one session on that month.

Explanation

This table shows the whole schedule for each person for all months of the year. The letter before "/" in the able indicates a package as the footnote says. The letter after "/" indicates a proficiency level. "Total Days" at the bottom indicates total scheduled training person-days during a month. "Total" at the right side indicates otal scheduled training days for a person. In the example, "Nick Johnson" is scheduled to be trained to the "Novice" level on "Lotus 123" in December, the "Novice" level on "Designer" in January, the "Regular" level on "Designer" in April, and the "Expert" level on "MS Word" in June. The total scheduled days for "Nick ohnson" is 11. The total scheduled person-days during November is 4.

TS: Training Schedules; By Person (Table)

Example

Training Schedule for Bill Muraham

Name	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Total
MS Word	N/1	x	x	.	.	.	x	.	.	R/3	x	.	4
Lotus 123	.	x	x	.	.	.	x	.	.	.	x	.	0
Designer	.	x	x	.	R/3	.	x	.	.	.	x	E/6	9
Total Days	1	0	0	0	3	0	0	0	0	3	0	6	13
Grand Total	4	4	4	3	4	6	6	6	6	6	3	6	56

Levels; N: novice R: regular E: expert

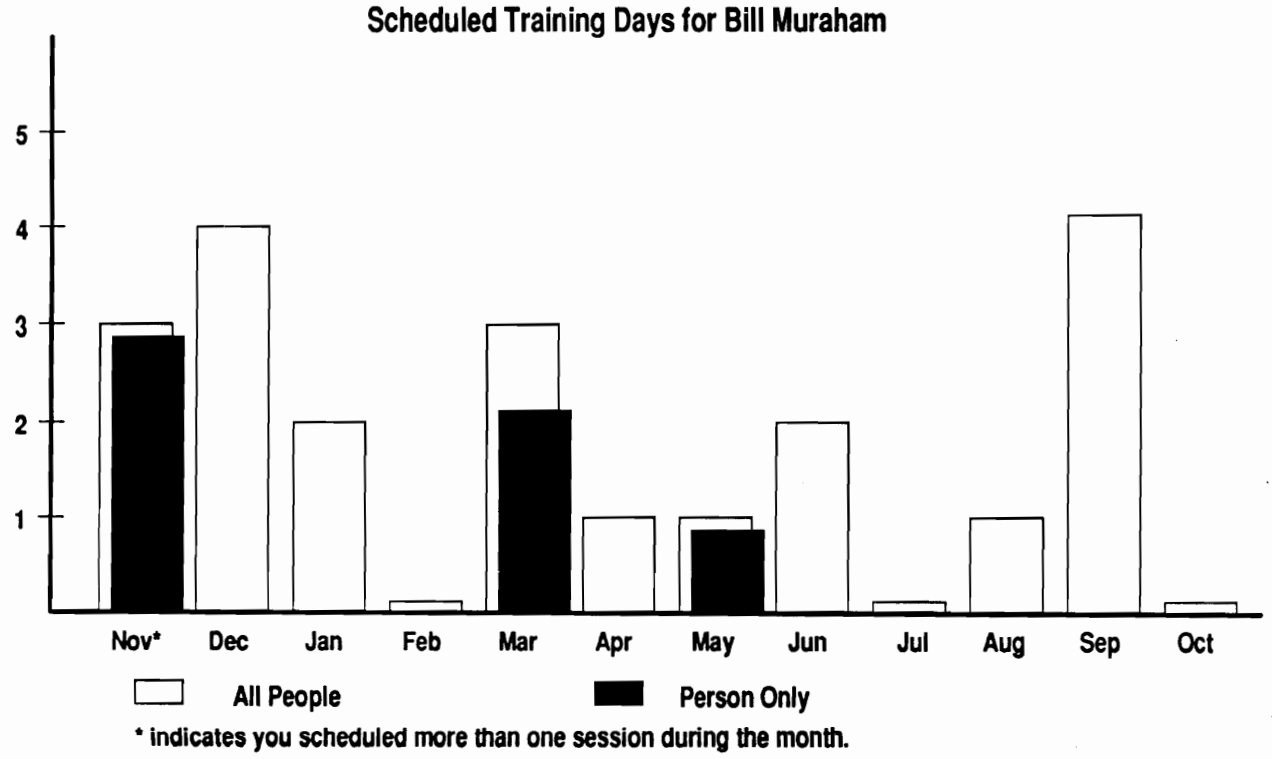
"x" indicates a busy month for the person to get training.

Explanation

This table shows the training schedule of a person. The number after "/" indicates required days for the raining session. "Grand Total" indicates total scheduled training days for all eight people. The schedules for even of the people are not shown in the table.

TS: Training Schedules; By Person (Graph)

Example



Explanation

This graph shows scheduled days for a person and for all people during each month. Blank bars indicate total scheduled person-days for all eight people during a month and black bars indicate scheduled days for a person during a month.

Gap between OP and IP; Whole (Table)

Example

Additional Number of People You Need to Assign

(Total Number of People = 8)

Package	Novice	regular	Expert
MS Word	0	1	-1
Lotus 123	0	0	0
Designer	1	0	0

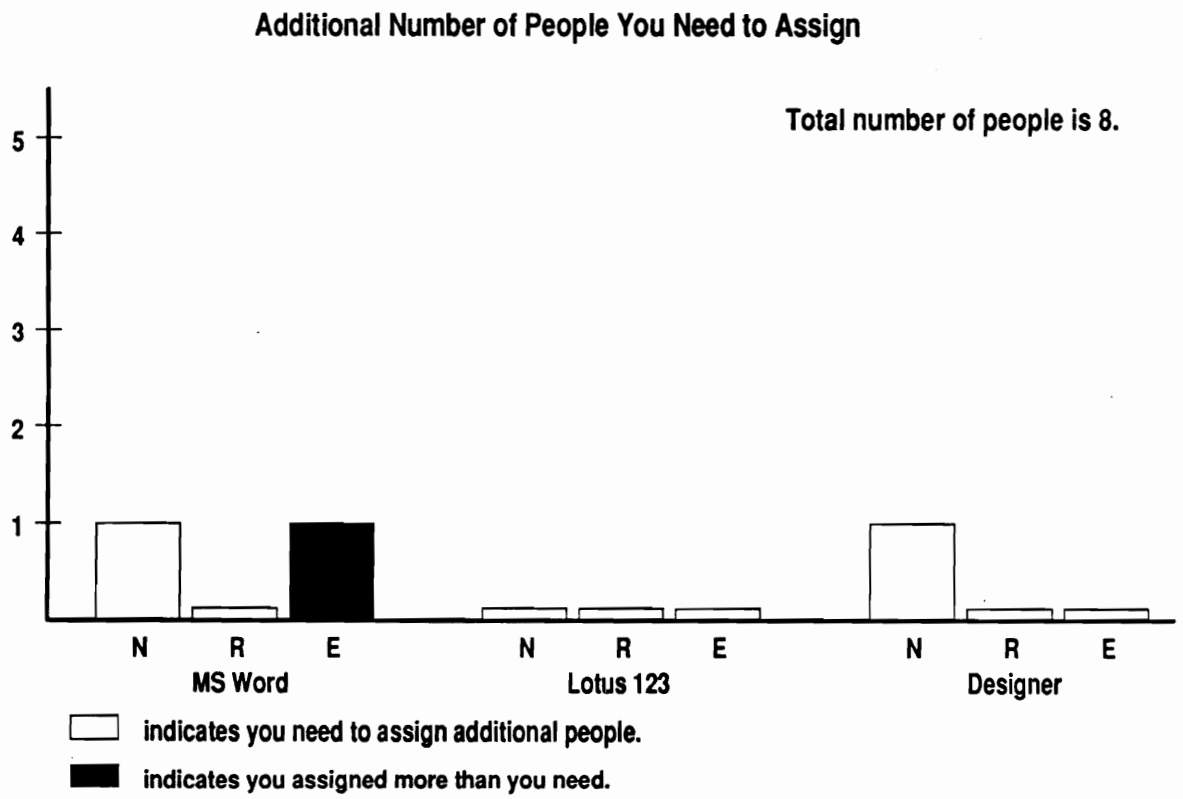
The numbers represent the difference (Desired - Expected).
Minus numbers mean you assigned more than you need.

Explanation

This table shows how more people you should assign on a proficiency level of a package. If the numbers are all zero, it means you assigned all you need. Minus numbers mean you assigned more than your organization needs. In the example, you assigned one more expert than you need and needs to assign one more regular on "MS Word."

Gap between OP and IP; Whole (Graph)

Example



Explanation

This graph shows how many more people you should assign on a proficiency level of a package. If the there is no bar, it means you assigned all you need. If there are black bars, it means you assigned more people than your organization needs.

Gap between OP and IP; By PKG (Table)

Example

Desired OP versus Expected OP of MS Word

(Total Number of People = 8)

Proficiency	Current	By Feb	By Jun	By Oct	Expected
Novice	1	3	1	2	4
Regular	1	1	0	2	1
Expert	0	0	1	0	2

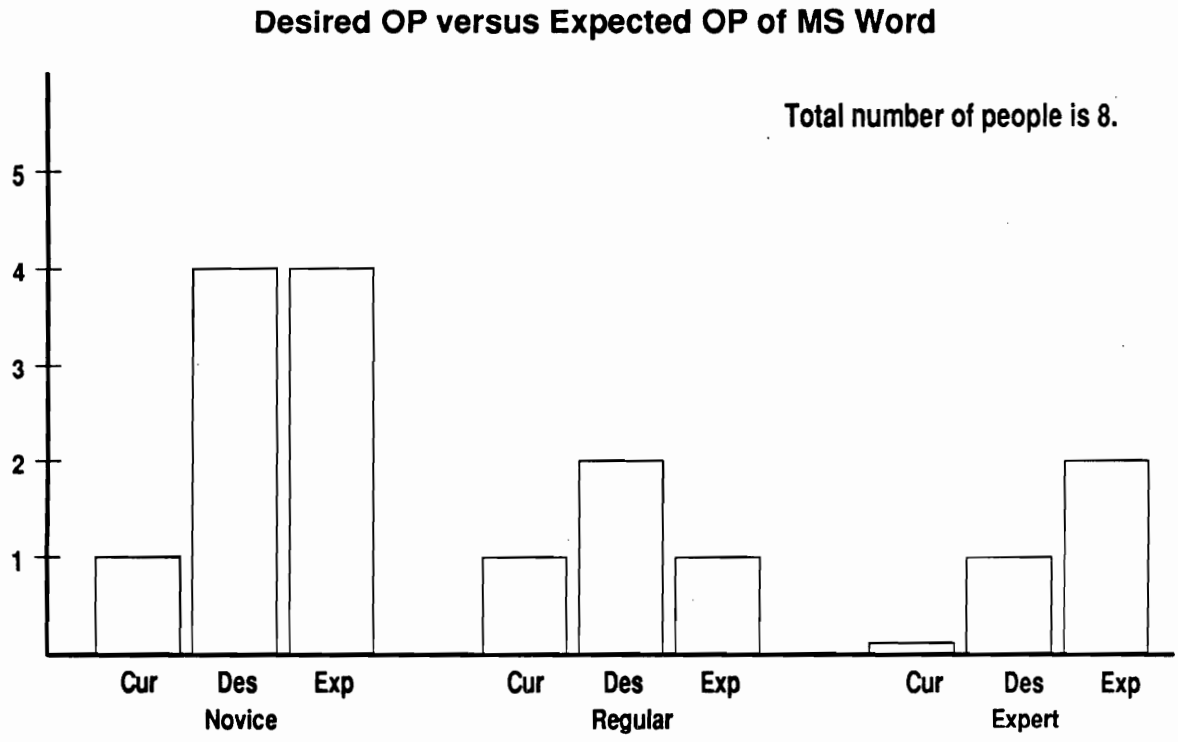
The numbers represent the number of people.

Explanation

This table shows the current number of people proficient on a package, the desired number of people by each milestone, and the number of people you assigned on the package.

Gap between OP and IP; By PKG (Graph)

Example



Explanation

This graph shows the current number of people proficient on a package, the desired number of people, and the number of people you assigned on the package.

Gap between TC and TS; Whole (Table)

Example

Additional Number of People You Need to Schedule

(Total Number of People = 8)

Package	Nov - Feb			Mar - Jun			Jul - Oct		
	N	R	E	N	R	E	N	R	E
MS Word	3	1	0	0	0	1	2	1	-1
Lotus 123	2	2	0	2	1	0	1	0	1
Designer	3	1	0	1	0	1	2	2	0

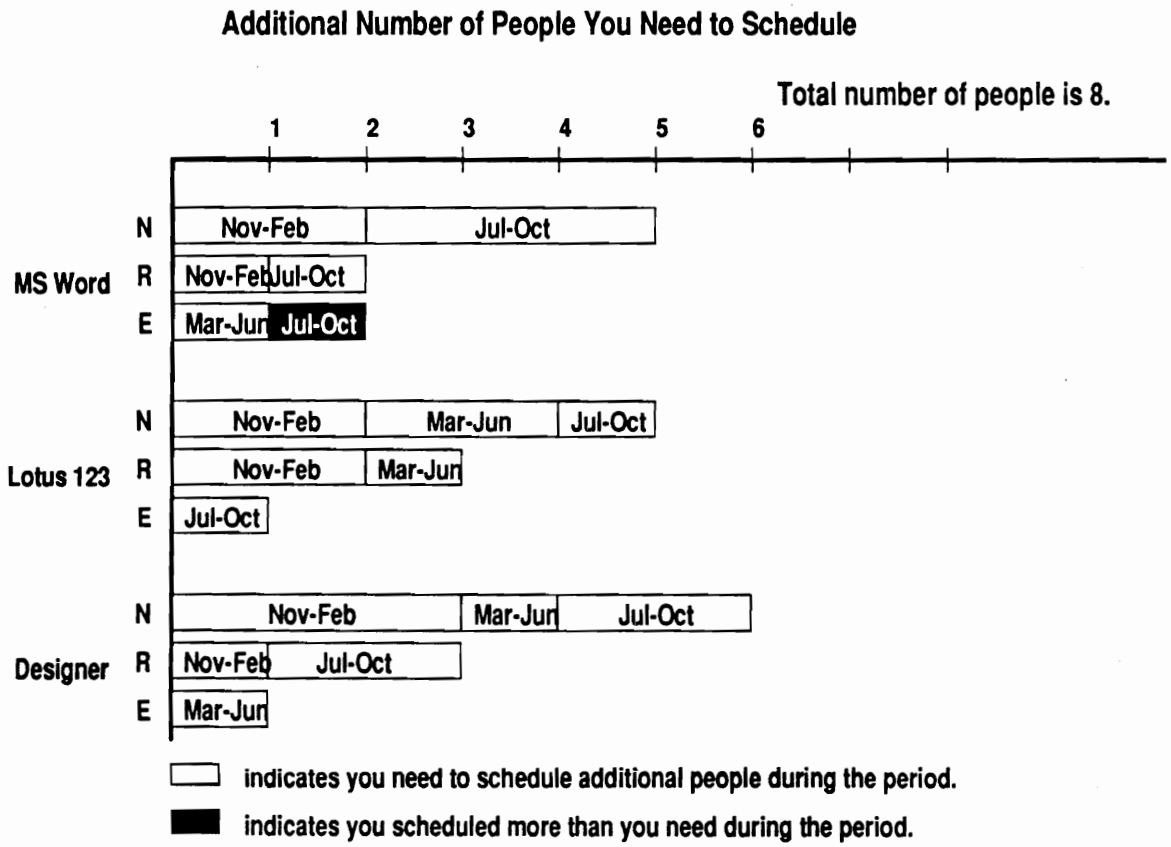
The numbers represent the additional number of people you need to schedule.
Minus numbers mean you scheduled more than you need during that period.

Explanation

This table shows how many more people you should schedule on a proficiency level of a package during a period. If the numbers are all zero, it means you scheduled all you need and the schedule satisfies your milestones for organizational proficiency. If there are minus numbers, it means you scheduled more people than you need during the period.

Gap between TC and TS; Whole (Graph)

Example



Explanation

This graph shows how more people you should schedule on a proficiency level of a package during a period. If there is no bar, it means you scheduled all you need and the schedule satisfies your milestones for organizational proficiency.

Gap between TC and TS; By PKG (Table)

Example

Additional Number of People You Need to Schedule on Designer

(Total Number of People = 8)

Proficiency	Nov-Feb	Mar-Jun	Jul-Oct
Novice	3	1	2
Regular	1	0	2
Expert	0	1	0

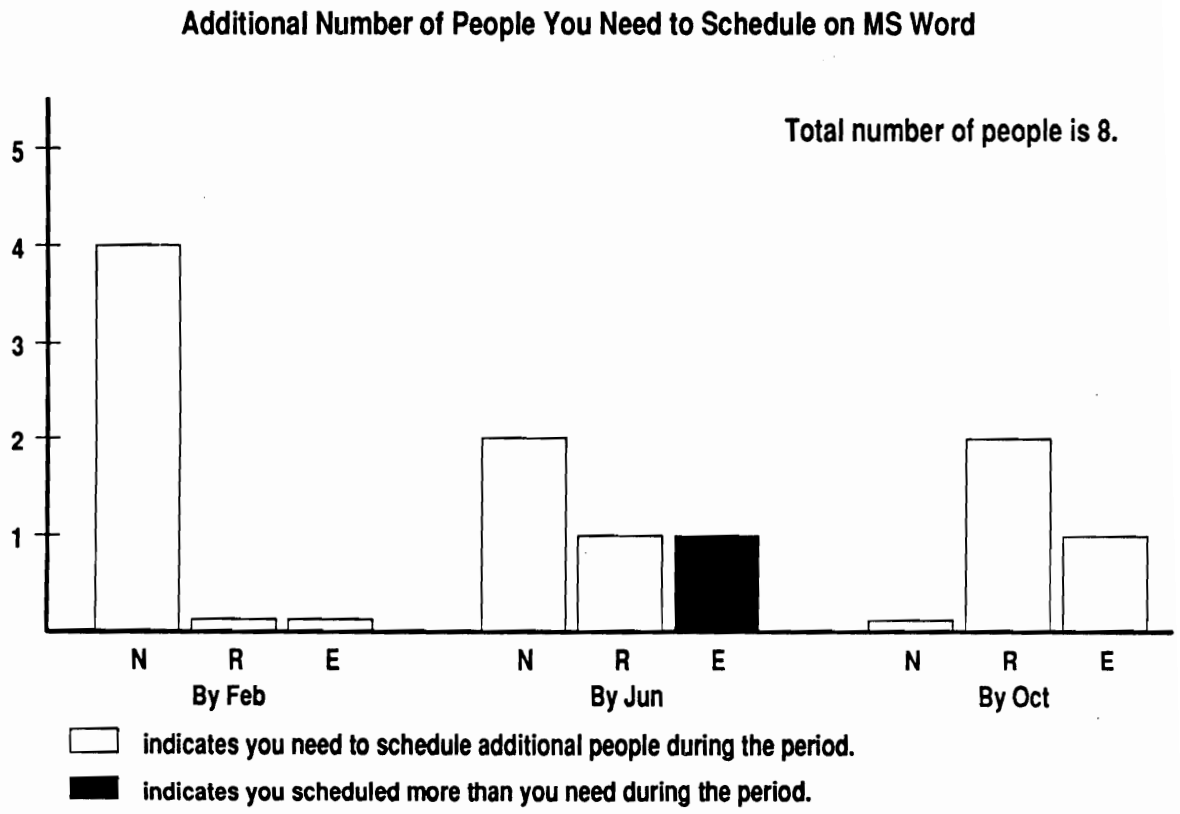
The numbers represent the additional number of people you need to schedule.
Minus numbers mean you scheduled more than you need during that period.

Explanation

This table shows the additional number of people you should schedule on each proficiency level on a package. If all numbers are zero, it means you scheduled all you need on the package and the schedule satisfies the organizational proficiency on the package. If there are minus numbers, it means you scheduled more than you need during the period.

Gap between TC and TS; By PKG (Graph)

Example



Explanation

This graph shows the number of people you should schedule more on each proficiency level on a package. If there is no bar, it means you scheduled all you need on the package and the schedule satisfies the organizational proficiency for the package. If there are black bars, it means you scheduled more than you need during the period.

Appendix B. MSLTRAIN Screens in Each Step

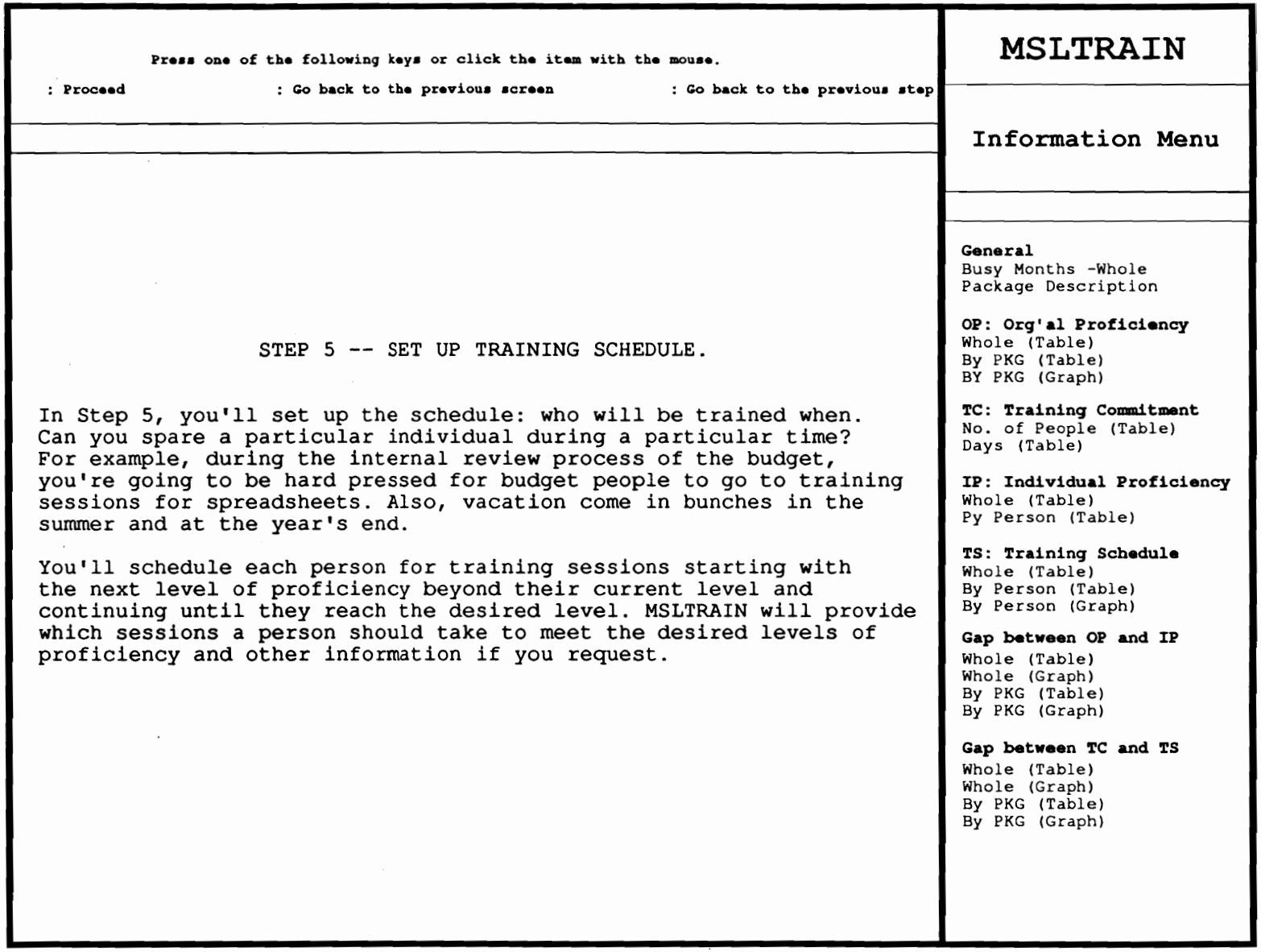


Figure 19. In the full guide interaction mode, MSLTRAIN guides the user through the steps.

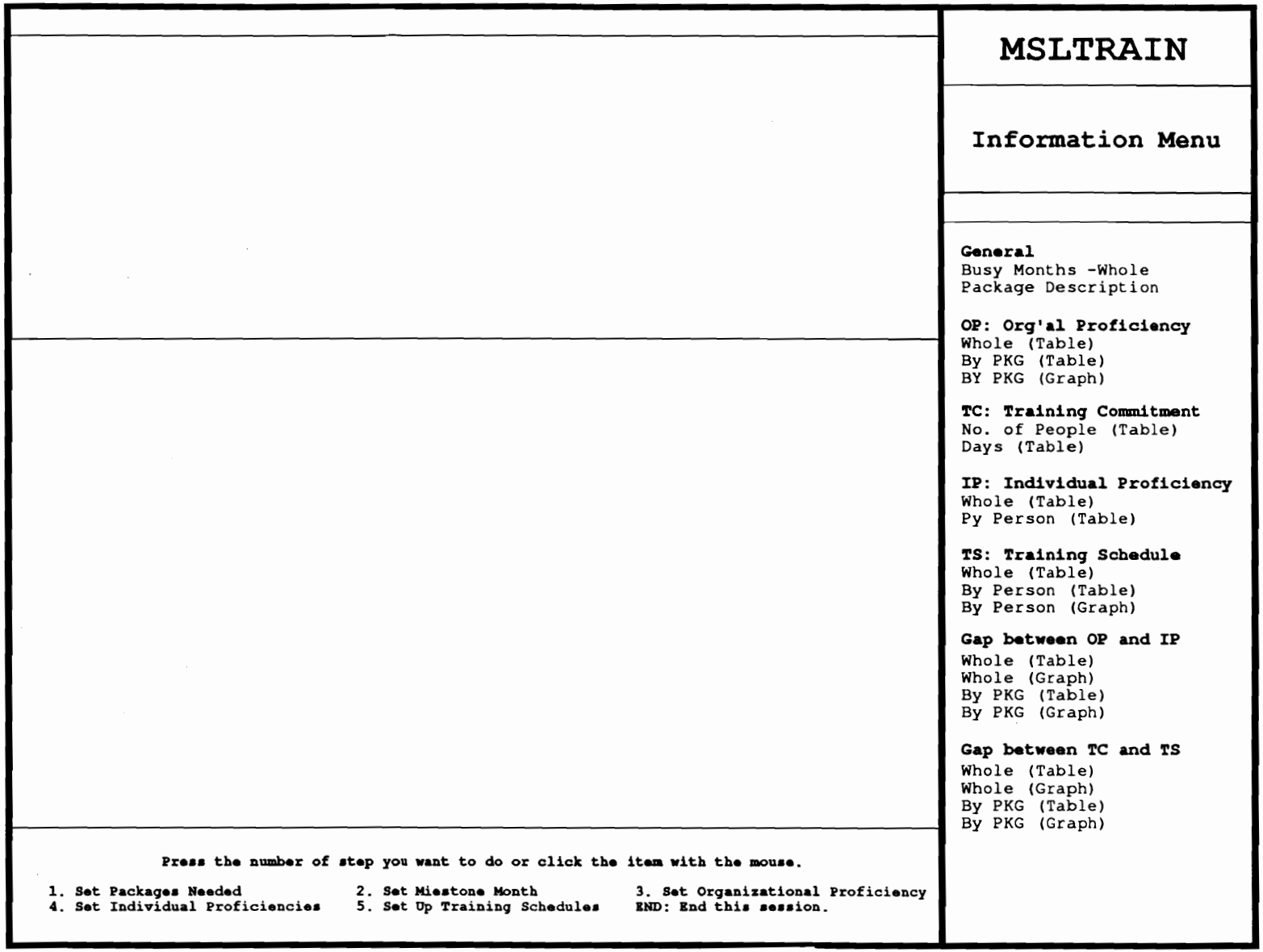


Figure 20. The guide pane can be placed at the bottom of the screen.

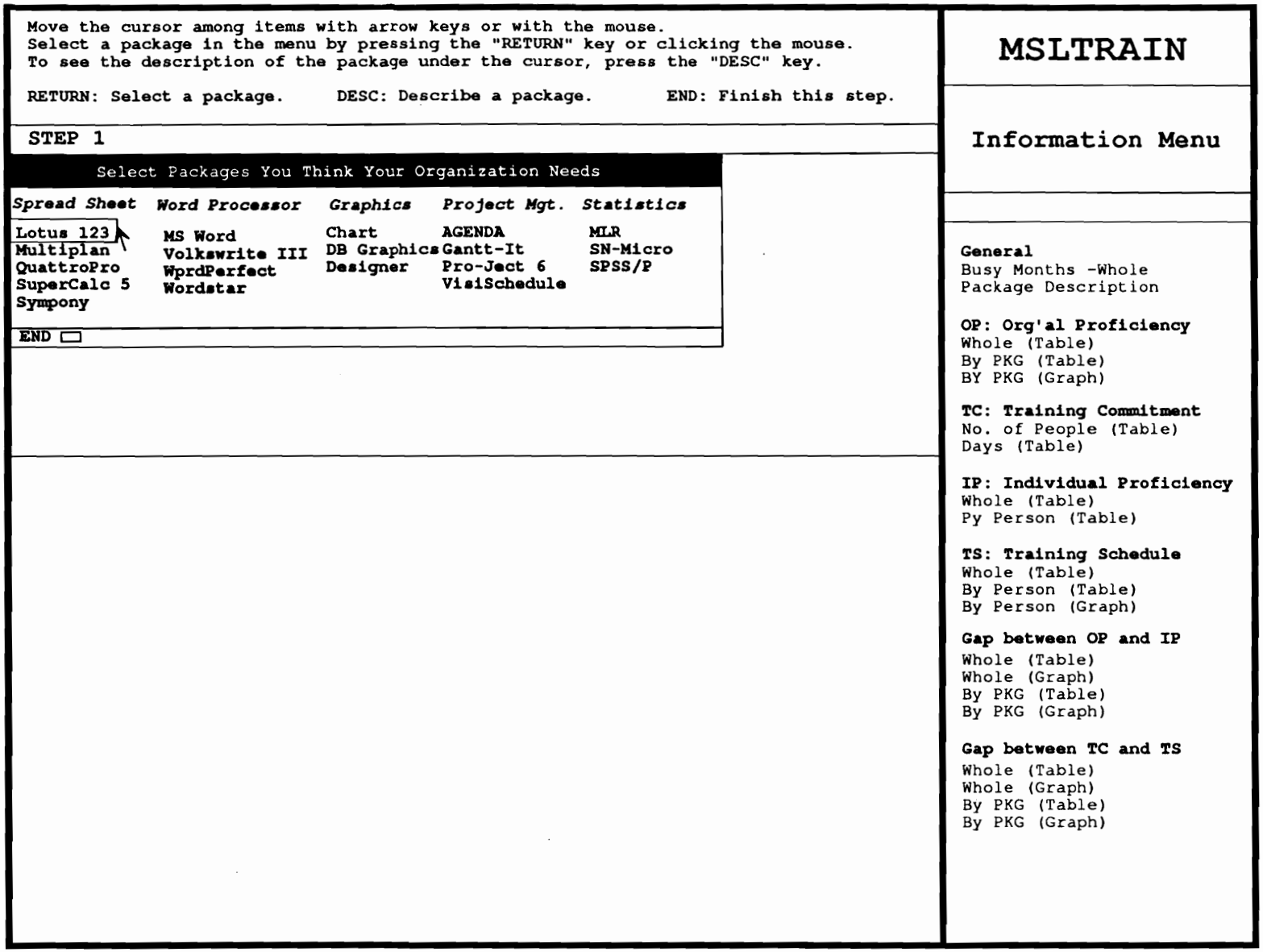


Figure 21. In Step 1, the manager determines the packages they will need.

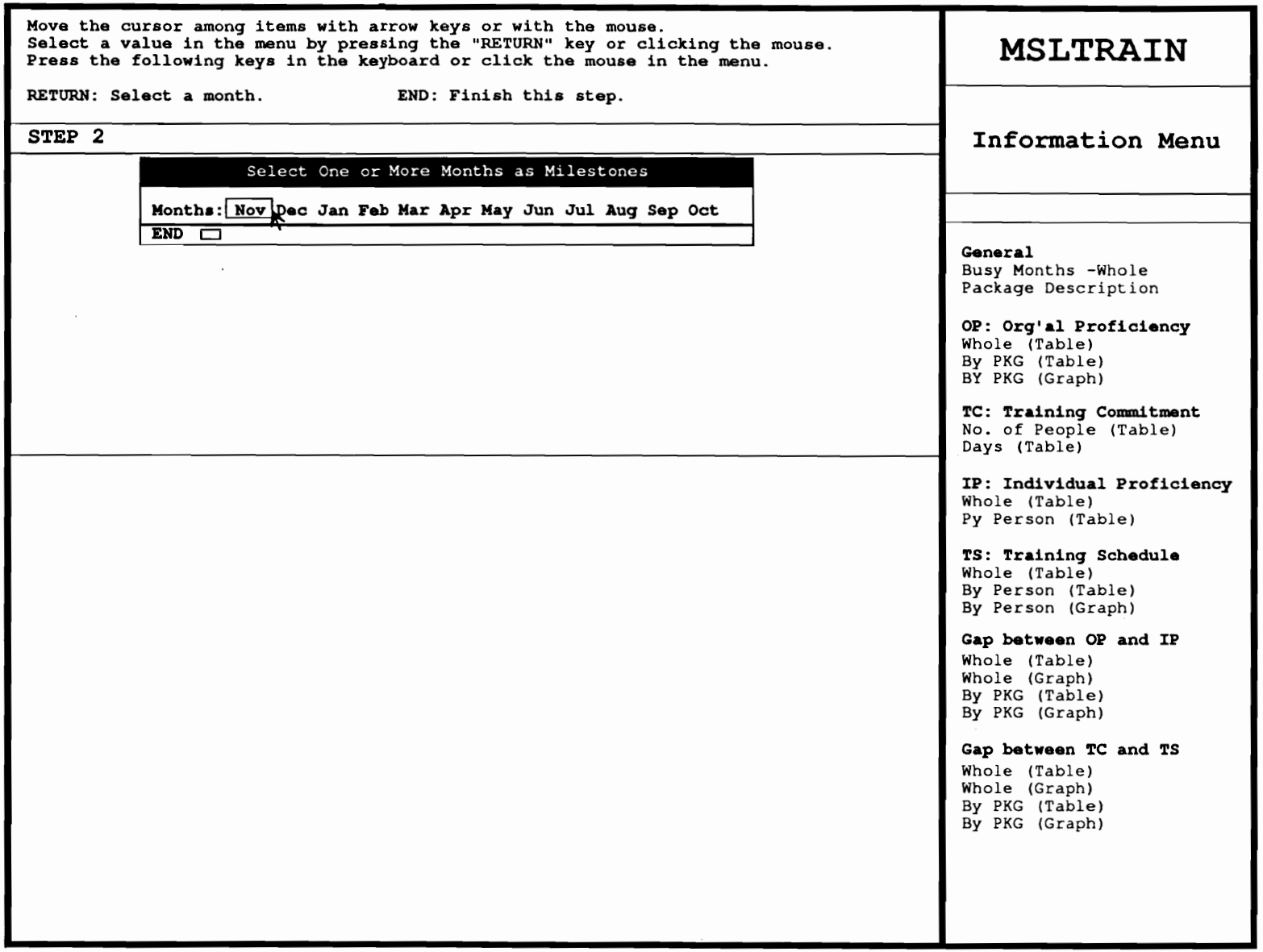


Figure 22. In Step 2, the manager sets milestone dates.

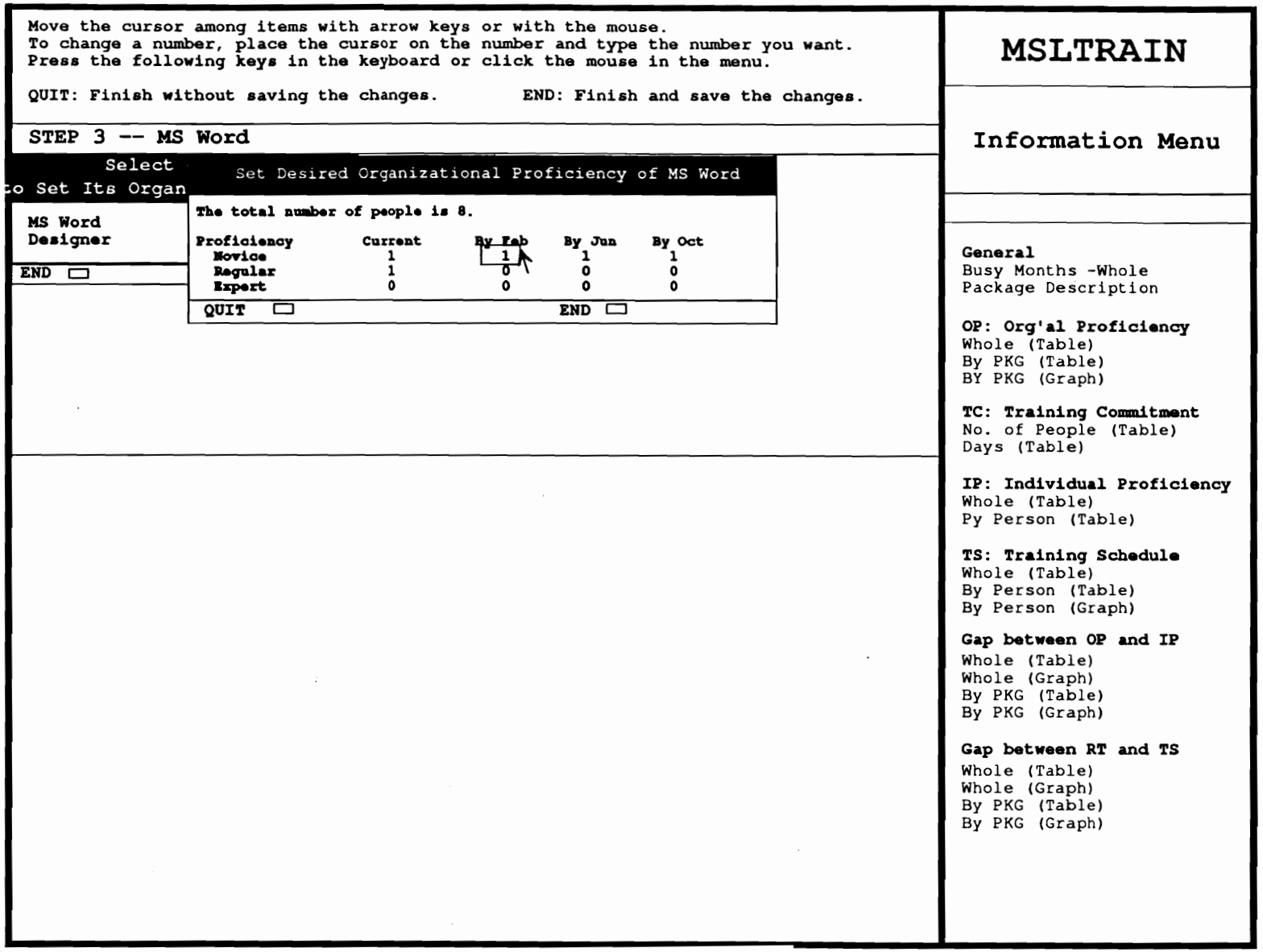


Figure 23. In Step 3, the manager sets desired organizational proficiency.

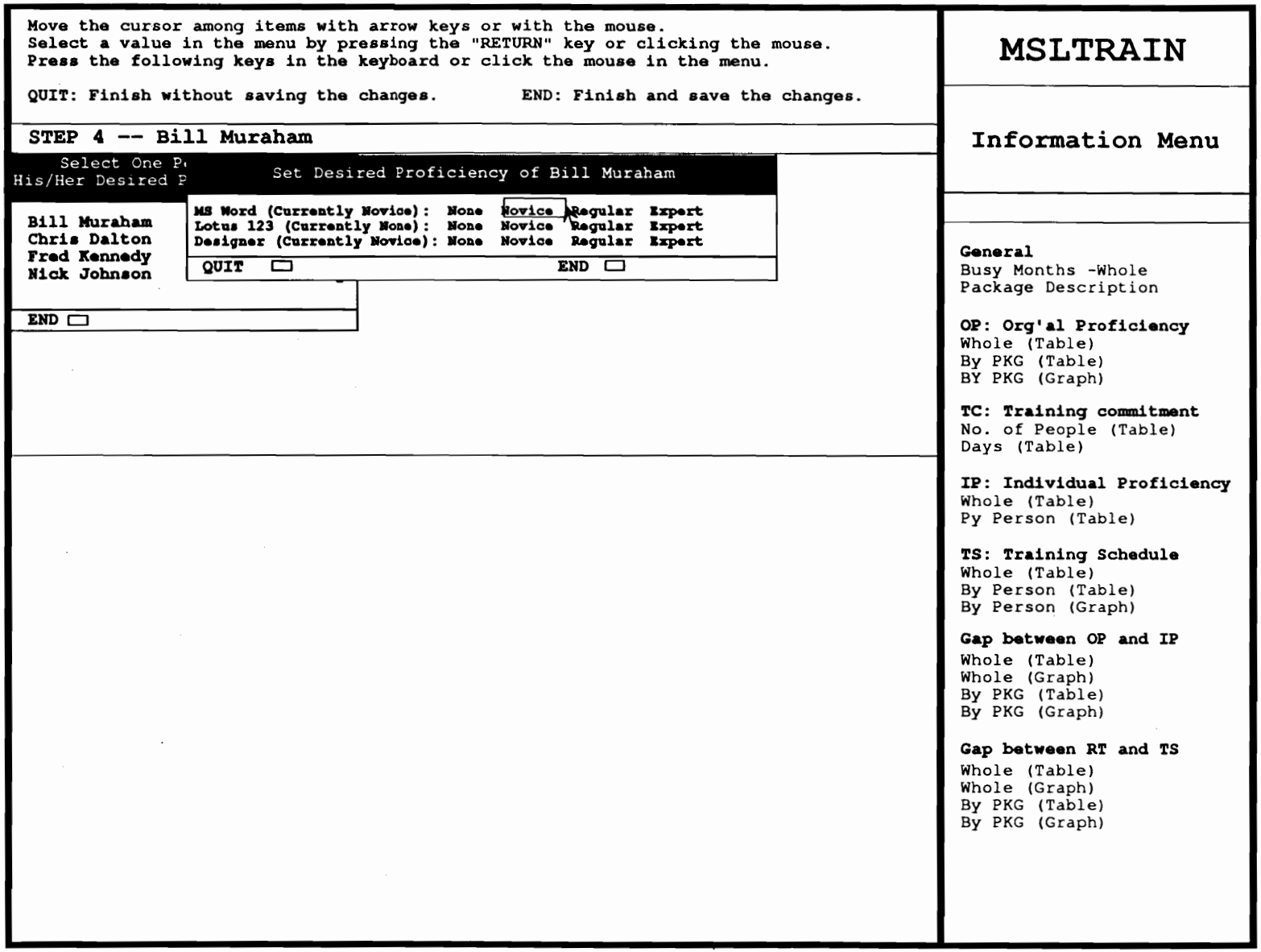


Figure 24. In Step 4, the manager sets desired individual proficiencies.

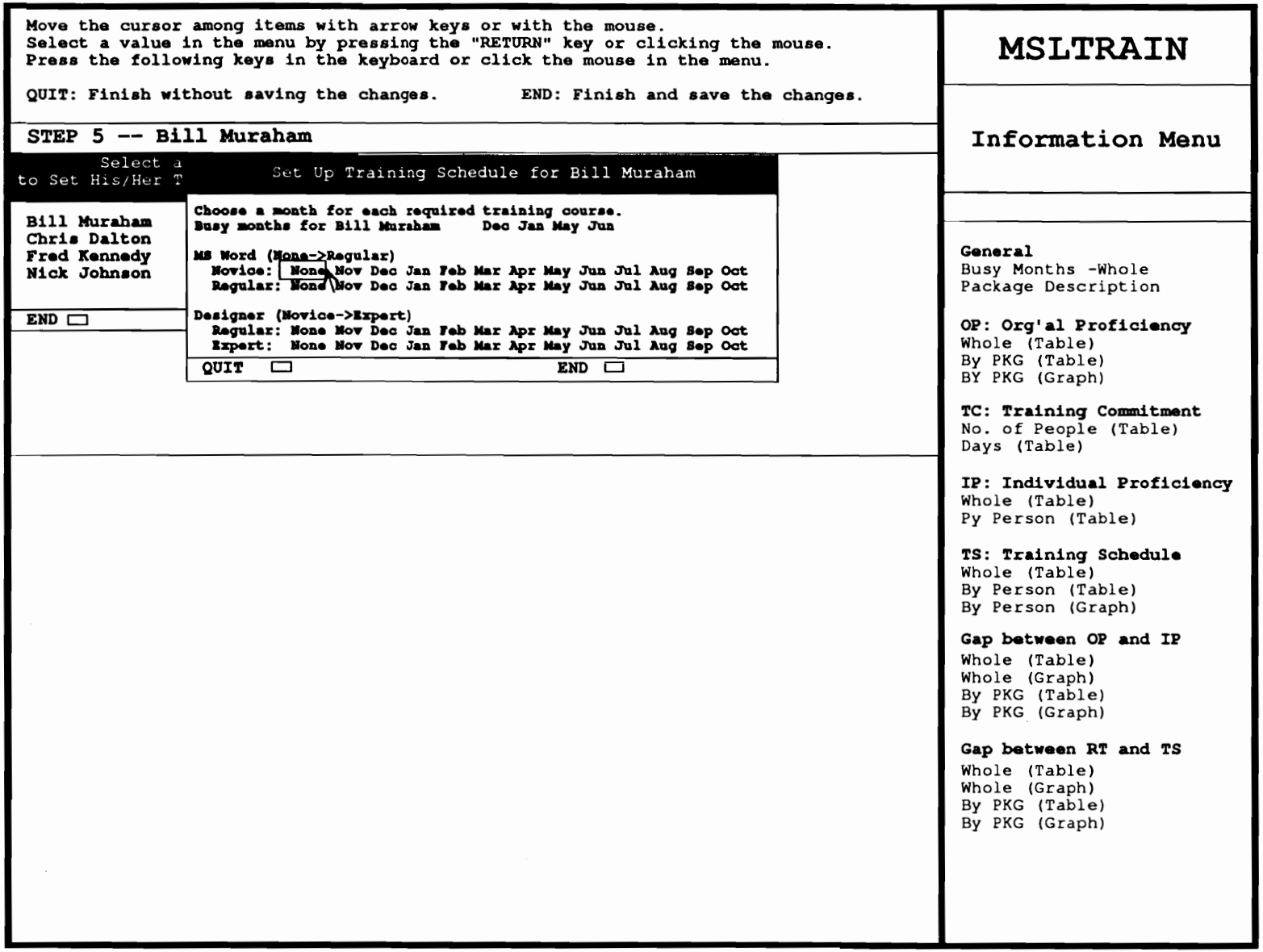


Figure 25. In Step 5, the manager schedules their people on training sessions.

Appendix C. Program Source Defining Reporters and Executors

```
;;; Flavor for basic MSLTRAIN demons
(DEFFLAVOR DEMON
   ((NAME NIL)
    (ACTIVE? NIL)
    (TYPE NIL)
    (TARGET-FUNCTION NIL)
    (ARGUMENTS NIL)
    (POSITION NIL)
    (WHEN-TO-ACT NIL)
    (CODE NIL))
   ()
   :GETTABLE-INSTANCE-VARIABLES
   :INITTABLE-INSTANCE-VARIABLES)

;;; Dispatch a demon to work.
;;; Actually define (:BEFORE or :AFTER) methods or Advise for functions.
(DEFMETHOD (DEMON :DISPATCH) ()
   (SETQ ACTIVE? T)
   (COMPILER:COMPILE-FORM
     (IF (CONSP TARGET-FUNCTION)
         `(DEFMETHOD ,(LIST (CAR TARGET-FUNCTION) POSITION (CADR TARGET-FUNCTION)) ,ARGUMENTS
            ,CODE)
         `(ADVISE ,TARGET-FUNCTION ,POSITION REPORT-TO-BB NIL ,CODE))))

;;; Recall a demon from work.
;;; Undefine a method or Unadvise a function.
(DEFMETHOD (DEMON :RECALL) ()
   (SETQ ACTIVE? NIL)
   (COMPILER:COMPILE-FORM
     (IF (CONSP TARGET-FUNCTION)
         `(UNDEFMETHOD ,(LIST (CAR TARGET-FUNCTION) POSITION (CADR TARGET-FUNCTION)))
         `(UNADVISE ,TARGET-FUNCTION ,POSITION))))

;;; Flavor for reporters that report what is happening between the user and the system.
(DEFFLAVOR REPORTER
   ((MSG-TITLE NIL)
    (MSG-GENERATOR NIL))
   (DEMON)
   :INITTABLE-INSTANCE-VARIABLES)

(DEFMETHOD (REPORTER :AFTER :INIT) (&REST IGNORE)
   (SETQ CODE `(SEND ,SELF :REPORT ,MSG-GENERATOR)))

(DEFMETHOD (REPORTER :REPORT) (MSG)
   (WHEN (AND ACTIVE?
              (SEND BB-PROCESS :ACTIVE-P)
              (OR (NULL MSG-GENERATOR) MSG)
              (OR (NULL WHEN-TO-ACT)
                  (IF (FUNCTIONP WHEN-TO-ACT)
                      (FUNCALL WHEN-TO-ACT MSG)
                      (*EVAL WHEN-TO-ACT))))
     (SETQ *REPORT* (CONS MSG-TITLE MSG))
     (PROCESS-WAIT " " #'(LAMBDA () (NULL *REPORT*)))))

;;; Macro to define a reporter
(DEFMACRO DEFREPORTER (NAME &REST INITIAL-VALUES)
   `(PROGN (DEFPARAMETER ,NAME (MAKE-INSTANCE 'REPORTER ,@INITIAL-VALUES :NAME ',NAME))
           (PUSHNEW ',NAME *ALL-REPORTERS*)
           ',NAME))

(DEFREPORTER NEW-SESSION-REPORTER
   :MSG-TITLE        :NEW-SESSION
   :TARGET-FUNCTION 'INITIALIZE-PARAMETERS
   :POSITION         :AFTER
   :MSG-GENERATOR    NIL)

(DEFREPORTER SESSION-START-REPORTER
```

```
   :MSG-TITLE        :START-SESSION
   :TARGET-FUNCTION '(MSLTRAIN :ASK-START-SESSION)
   :POSITION         :AFTER
   :MSG-GENERATOR    NIL)

(DEFREPORTER SESSION-END-REPORTER
   :MSG-TITLE        :END-SESSION
   :TARGET-FUNCTION 'HANDLE-END-SESSION
   :POSITION         :BEFORE
   :MSG-GENERATOR    NIL)

(DEFREPORTER YES-NO-REPORTER
   :MSG-TITLE        :CONTINUE
   :TARGET-FUNCTION 'MSL-MOUSE-CONFIRM
   :POSITION         :AFTER
   :MSG-GENERATOR
   '(LIST (TIME) (CAR VALUES) "KEY" (SEND W:SELECTED-WINDOW :NAME) NIL))

(DEFREPORTER MESSAGE-ERASE-REPORTER
   :MSG-TITLE        :CONTINUE
   :TARGET-FUNCTION '(RETURN-MESSAGE-MIXIN :DISPLAY-RETURN-MESSAGE)
   :POSITION         :AFTER
   :MSG-GENERATOR    '(LIST (TIME) 'SPACE 'KEY "POP-UP" SELF))

(DEFREPORTER DB-UPDATE-REPORTER
   :MSG-TITLE        :FOLLOW-UP
   :TARGET-FUNCTION 'UPDATE-BASE
   :POSITION         :AFTER
   :MSG-GENERATOR
   '(UNLESS (EQ (CAR VALUES) :BACK)
      '(CONFIRM)))

(DEFREPORTER ERROR-REPORTER
   :MSG-TITLE        :ERROR
   :TARGET-FUNCTION 'HANDLE-UNKNOWN-INPUT
   :POSITION         :BEFORE
   :MSG-GENERATOR
   '(LET ((SOURCE (GETF (CDR ARGLIST) :SOURCE "UNKNOWN")))
      (LIST (TIME) (CAR ARGLIST) SOURCE)))

(DEFREPORTER TOP-COMMANDS-REPORTER
   :MSG-TITLE        :COMMAND
   :TARGET-FUNCTION '(MSLTRAIN :EXECUTE-COMMAND)
   :ARGUMENTS        '(&REST IGNORE)
   :POSITION         :BEFORE
   :MSG-GENERATOR
   '(WHEN UCL:COMMAND-ENTRY
      (LET* ((ACTION
               (MSL-CHAR-NAME
                 (CAAR (SEND (IF (CONSP UCL:COMMAND-ENTRY)
                                 (CAR UCL:COMMAND-ENTRY) UCL:COMMAND-ENTRY)
                             :KEYS)))))
        (PROG1
          (AND ACTION (LIST (TIME) ACTION INPUT-DEVICE "TOP" *CURRENT-NODE*))
          (SETQ INPUT-DEVICE NIL)))))

(DEFREPORTER MENU-KEY-REPORTER
   :MSG-TITLE        :COMMAND
   :TARGET-FUNCTION '(MMENU :PROCESS-CHARACTER)
   :WHEN-TO-ACT      'DEFAULT-WHEN-TO-ACT
   :ARGUMENTS        '(CH)
   :POSITION         :BEFORE
   :MSG-GENERATOR
   '(LET (ACTION ITEM)
      (SETQ CH (MSL-CHAR-NAME CH))
      (COND ((ASSOC CH W:COMMAND-CHARACTERS)
             (SETQ ACTION
```

```
                    (COND ((OR (EQ CH 'QUIT) (EQ CH 'END)) 'QUIT)
                          ((AND (SETQ SELECT-TYPE (SELECT-CHAR? CH))
                                (NOT (EQUAL W:NAME "MENU 1")))
                           (SETQ ITEM (GETF (CDR W:CURRENT-ITEM) :VALUE))
                           (IF INFO-MENU-P `(SELECT ,SELECT-TYPE) 'SELECT)))))
             ((ASSOC CH *COMMON-COMMAND-CHARACTERS*)
              (OR (AND INFO-MENU-P (EQ CH 'INFO))
                  (SETQ ACTION CH))))
       (AND ACTION (LIST (TIME) ACTION 'KEY W:NAME ITEM))))

(DEFREPORTER CVV-KEY-REPORTER
   :MSG-TITLE        :COMMAND
   :TARGET-FUNCTION '(MSL-CVV-WINDOW :PROCESS-CHARACTER)
   :WHEN-TO-ACT      'DEFAULT-WHEN-TO-ACT
   :ARGUMENTS        '(IGNORE)
   :POSITION         :AFTER
   :MSG-GENERATOR
   '(PROGN
      (COND ((EQ COMMAND-TYPE :WINDOW)
             (COND ((OR (EQ MSL-CH 'END) (EQ MSL-CH 'QUIT))
                    (LIST (TIME) MSL-CH 'KEY W:NAME *WORKING-ITEM*))
                   ((OR (EQ MSL-CH 'UP) (EQ MSL-CH 'DOWN))
                    (MULTIPLE-VALUE-BIND (NIL Y-OFF)
                        (TV:SHEET-CALCULATE-OFFSETS SELF TV:MOUSE-SHEET)
                      (LIST (TIME) MSL-CH 'KEY W:NAME (SHEET-LINE-NO NIL (- SYS:MOUSE-Y Y-OFF)))))))
            ((EQ COMMAND-TYPE :SELECT)
             (WHEN (EQUAL W:NAME "CVV 5")
               (LIST (TIME) SET-VALUE-TYPE 'KEY W:NAME
                     (LIST (CADR CURRENT-VALUE) (CADAR CURRENT-VALUE)))))
            ((EQ COMMAND-TYPE :COMMON)
             (LIST (TIME) MSL-CH 'KEY W:NAME *WORKING-ITEM*)))))

(DEFREPORTER SUGG-KEY-REPORTER
   :MSG-TITLE        :COMMAND
   :TARGET-FUNCTION '(SUGG:NICE-SCROLLING-SUGGESTIONS-WINDOW :PROCESS-CHARACTER)
   :WHEN-TO-ACT      'DEFAULT-WHEN-TO-ACT
   :ARGUMENTS        '(CH)
   :POSITION         :BEFORE
   :MSG-GENERATOR
   '(LET (ACTION ITEM SELECT-TYPE)
      (WHEN (ASSOC (SETQ CH (MSL-CHAR-NAME CH)) W:COMMAND-CHARACTERS)
        (AND (SETQ ACTION
                   (COND ((MEMBER CH '(END QUIT INFO)) 'QUIT)
                         ((SETQ SELECT-TYPE (SELECT-CHAR? CH))
                          (SETQ ITEM (SECOND (THIRD W:CURRENT-ITEM)))
                          `(SELECT ,SELECT-TYPE))))
             (LIST (TIME) ACTION 'KEY "INFO MENU" ITEM)))))

(DEFREPORTER MENU-MOUSE-REPORTER
   :MSG-TITLE        :COMMAND
   :TARGET-FUNCTION '(MMENU :MOUSE-BUTTONS-ON-ITEM)
   :WHEN-TO-ACT      'DEFAULT-WHEN-TO-ACT
   :ARGUMENTS        '(IGNORE)
   :POSITION         :AFTER
   :MSG-GENERATOR
   '(WHEN W:CHOSEN-ITEM
      (LET ((ITEM (GETF (CDR W:CHOSEN-ITEM) :VALUE)) ACTION)
        (SETQ ACTION (IF INFO-MENU-P `(SELECT ,SELECT-TYPE) 'SELECT))
        (LIST (TIME) ACTION 'MOUSE W:NAME ITEM))))

(DEFREPORTER CVV-MOUSE-REPORTER
   :MSG-TITLE        :COMMAND
   :TARGET-FUNCTION 'MSL-CVV-PROCESS-MESSAGE
   :WHEN-TO-ACT      'DEFAULT-WHEN-TO-ACT
   :POSITION         :AFTER
   :MSG-GENERATOR
   '(LET ((WINDOW (CAR ARGLIST)))
```

```
      (WHEN (AND (EQUAL (SEND WINDOW :NAME) "CVV 5")
                 (CONSP (SECOND ARGLIST))
                 (EQ (CAR (SECOND ARGLIST)) :VARIABLE-CHOICE))
      (LIST (TIME) (SEND WINDOW :SET-VALUE-TYPE) 'MOUSE "CVV 5"
            (LIST (CADR (SEND WINDOW :CURRENT-VALUE)) (CADAR (SEND WINDOW :CURRENT-VALUE)))))))

(DEFREPORTER MARGIN-MOUSE-REPORTER
   :MSG-TITLE        :COMMAND
   :TARGET-FUNCTION 'TV:HANDLE-CHOICE-BUTTON
   :WHEN-TO-ACT      'DEFAULT-WHEN-TO-ACT
   :POSITION         :AFTER
   :MSG-GENERATOR
   '(LET ((CHOICE-BOX (CADR VALUES)))
      (WHEN CHOICE-BOX
        (LIST (TIME) (INTERN (STRING-UPCASE (CAR CHOICE-BOX)) 'MSL)
              'MOUSE (CAR VALUES) *WORKING-ITEM*))))

(DEFREPORTER SUGG-MOUSE-REPORTER
   :MSG-TITLE        :COMMAND
   :TARGET-FUNCTION '(SUGG:SUGGESTIONS-COMMAND-MENU-MIXIN :AFTER-LEFT-MOUSE-BUTTON)
   :POSITION         :BEFORE
   :ARGUMENTS        '(SELECT-TYPE)
   :MSG-GENERATOR
   '(WHEN (AND (EQUAL SYS:*CURRENTLY-SELECTED-APPLICATION* *MSL*)
               (CONSP W:LAST-ITEM) (CONSP (NTH 2 W:LAST-ITEM))
               (EQ (CAR (NTH 2 W:LAST-ITEM)) 'SUGG-DISPLAY-INFO-ITEM))
      (LIST (TIME) `(SELECT ,SELECT-TYPE) 'MOUSE "INFO MENU" (NTH 1 (NTH 2 W:LAST-ITEM)))))

(DEFUN DEFAULT-WHEN-TO-ACT (MSG)
   (OR (SUBSTRING? "MENU" (NTH 3 MSG))
       (SUBSTRING? "CVV"  (NTH 3 MSG))))

(DEFUN SET-REPORTERS (&OPTIONAL (REPORTERS *ALL-REPORTERS*))
   (DOLIST (X REPORTERS)
     (SEND (SYMBOL-VALUE X) :DISPATCH)))

(DEFFLAVOR EXECUTOR
   ((ARGS NIL)
    (ACTION NIL))
   (DEMON)
   :INITTABLE-INSTANCE-VARIABLES
   :SETTABLE-INSTANCE-VARIABLES
   (:SPECIAL-INSTANCE-VARIABLES ARGS))

(DEFMETHOD (EXECUTOR :AFTER :INIT) (&REST IGNORE)
   (OR CODE (SETQ CODE `(SEND ,SELF :EXECUTE))))

(DEFMETHOD (EXECUTOR :EXECUTE) ()
   (WHEN ACTIVE?
     (EVAL ACTION)))

(DEFMACRO DEFEXECUTOR (NAME &REST INITIAL-VALUES)
   `(PROGN (DEFPARAMETER ,NAME (MAKE-INSTANCE 'EXECUTOR ,@INITIAL-VALUES :NAME ',NAME))
           (PUSHNEW ',NAME *ALL-EXECUTORS*)
           ',NAME))

(DEFEXECUTOR CVV-CURSOR-PLACER
   :TARGET-FUNCTION '(MSL-CVV-WINDOW :SELECT)
   :POSITION         :AFTER
   :ARGUMENTS        '(&REST IGNORE)
   :CODE
   '(WHEN (AND CURSOR-LINE (NUMBERP CURSOR-LINE))
      (UNWIND-PROTECT
          (DOTIMES (N CURSOR-LINE)
            (SEND SELF :MOVE-CURSOR 'TV:DOWN))
        (SETQ CURSOR-LINE NIL))))
```

```
(DEFUN SET-EXECUTORS (&OPTIONAL (EXECUTORS *ALL-EXECUTORS*))
   (DOLIST (X EXECUTORS)
     (SEND (SYMBOL-VALUE X) :DISPATCH)))
```

Appendix D. Program Source Defining the Blackboard Architecture

1. Scheduler

```
;;;
;;; Main controller for the blackboard architecture.
;;; It contains scheduling information.
;;;
(DEFFLAVOR SCHEDULER
    ((NAME NIL)
     (DESCRIPTION NIL)
     (CYCLE NIL)
     (KSAR-NO NIL)
     (CURRENT-PROBLEM NIL)
     (TRIGGERED-KSARS NIL)
     (INVOKABLE-KSARS NIL)
     (EVENTS NIL)
     (EVENT-NO 0)
     (CONTROL NIL)
     (GOAL NIL)
     (SELECTED-KSAR NIL))
    ()
    :GETTABLE-INSTANCE-VARIABLES
    (:SETTABLE-INSTANCE-VARIABLES CURRENT-PROBLEM)
    (:INITTABLE-INSTANCE-VARIABLES NAME DESCRIPTION CONTROL))

;;;
;;; Main control of problem solving in the BB process
;;; 1. Inspect the goal. If the goal has been satisfied, stop.
;;; 2. Update KSAR lists. If there is no invokable KSAR, stop.
;;; 3. Choose one KSAR among invokable KSARs.
;;; 4. Execute the selected KSAR. Go to 1.
;;;
(DEFMETHOD (SCHEDULER :SOLVE) ()
   (WHEN CURRENT-PROBLEM
     (SETQ GOAL    (SEND CURRENT-PROBLEM :GOAL))
     (SETQ CONTROL (SEND CURRENT-PROBLEM :CONTROL)))
   (SETQ CYCLE 0 KSAR-NO 0)
   (SETQ TRIGGERED-KSARS NIL
         INVOKABLE-KSARS NIL
         SELECTED-KSAR   NIL)
   (SETQ EVENTS '(NIL))
   (UNWIND-PROTECT
       (LOOP (AND GOAL (EVAL GOAL) (RETURN))
             (SEND SELF :UPDATE-KSARS)
             (OR INVOKABLE-KSARS (RETURN))
             (SEND SELF :CHOOSE-KSAR)
             (SEND SELF :INTERPRET-KSAR))
     (SETQ EVENTS NIL)))

;;;
;;; Update KSAR lists (TRIGGERED-KSARs and INVOKABLE-KSARs) with new events.
;;; Triggered KSAR: Triggered but not ready to be executed
;;; Invokable KSAR: Triggered and ready to be executed
;;;
(DEFMETHOD (SCHEDULER :UPDATE-KSARS) ()
   (WHEN (OR (NULL SELECTED-KSAR) (EQUAL (SEND SELECTED-KSAR :STATUS) :EXECUTED))
     (WHEN SELECTED-KSAR
       (DELETE SELECTED-KSAR INVOKABLE-KSARS)
       (SETQ SELECTED-KSAR NIL))
     (DOLIST (EVENT EVENTS)
       (DOLIST (X (IF EVENT
                      (CDR (ASSOC (EVENT-LEVEL EVENT) *EVENT-KSS-TABLE*))
                      '(KS$ACCEPT-REPORT)))
         (LET* ((KS (SYMBOL-VALUE X))
                (COND-VALUES (SEND KS :TRIGGERED? EVENT)))
           (UNLESS (EQ COND-VALUES 'NIX)
```

```
            (PUSH (MAKE-INSTANCE 'KSAR :NAME (INCREMENT KSAR-NO) :KS KS :TRIGGER-EVENT EVENT
                                       :TRIGGER-CYCLE (INCREMENT CYCLE) :CONDITION-VALUES COND-VALUES)
                  TRIGGERED-KSARS)))))
    (SETQ EVENTS NIL)
    (WHEN TRIGGERED-KSARS
      (DOLIST (X TRIGGERED-KSARS)
        (OR (EQ (SEND X :PRECONDITION? :BIND NIL) 'NIX)
            (PUSH X INVOKABLE-KSARS)))
      (DOLIST (X INVOKABLE-KSARS)
        (DELETE X TRIGGERED-KSARS)))))

;;;
;;; Choose a KSAR among invokable KSAR's.
;;; Current Rule: 1. Choose a KSAR whose to-level is lower than those of others.
;;;               2. Choose a KSAR whose cycle number is larger than those of others.
;;;                  But, if the problem's control rule is FIFO (First In, First Out),
;;;                       vice versa.
;;; ***** More sophisticated, domain-dependent rules might be desired. ********
;;;
(DEFMETHOD (SCHEDULER :CHOOSE-KSAR) ()
   (LET (NOT-SATISFIEDS
         (W-LEVEL *BIG-INTEGER*)
         (W-CYCLE-NO (IF (EQ CONTROL :FIFO) *BIG-INTEGER* 0)))
     (DOLIST (X INVOKABLE-KSARS)
       (IF (EQ (SEND X :PRECONDITION? :BIND T) 'NIX)
           (PUSH X NOT-SATISFIEDS)
           (LET* ((TO-LEVEL (SEND (SEND X :KS) :TO-LEVEL))
                  (BB (SYMBOL-VALUE (OR (SEND (SEND X :KS) :BB) (SEND CURRENT-PROBLEM :BB))))
                  (LEVEL (OR (POSITION TO-LEVEL (SEND BB :LEVEL-NAMES))
                             (1+ (LENGTH (SEND BB :LEVEL-NAMES)))))
                  (CYCLE-NO (SEND X :TRIGGER-CYCLE)))
             (OR (> LEVEL W-LEVEL)
                 (AND (= LEVEL W-LEVEL)
                      (IF (EQ CONTROL :FIFO)
                          (> CYCLE-NO W-CYCLE-NO) (< CYCLE-NO W-CYCLE-NO)))
                 (SETQ SELECTED-KSAR X W-LEVEL LEVEL W-CYCLE-NO CYCLE-NO)))))
     (SEND SELECTED-KSAR :SET-STATUS :SCHEDULED)
     (AND NOT-SATISFIEDS
          (DOLIST (X NOT-SATISFIEDS)
            (DELETE X INVOKABLE-KSARS)))
     SELECTED-KSAR))

;;; Execute the action part of the selected KSAR. Execution of a KSAR causes some changes in the
;;;   blackboard and creates events.
(DEFMETHOD (SCHEDULER :INTERPRET-KSAR) ()
   (WHEN SELECTED-KSAR
     (SEND SELECTED-KSAR :EXECUTE)
     (SEND SELECTED-KSAR :SET-STATUS :EXECUTED)))

(DEFMETHOD (SCHEDULER :ADD-EVENT) (&KEY TYPE LEVEL MSG)
   (PUSH (MAKE-EVENT :NO (INCREMENT EVENT-NO) :TYPE TYPE :LEVEL LEVEL :MSG MSG) EVENTS))

;;; Association list keeping knowledges sources triggered on each blackboard levels.
(DEFPARAMETER *EVENT-KSS-TABLE*
   '((INPUT
       KS$UPDATE-INT-HISTORY KS$INFER-CONFIRM-SEMANTICS KS$INFER-TOP-SEMANTICS
       KS$INFER-ACCESS-SEMANTICS KS$INFER-STEP-SEMANTICS KS$INITIALIZE-NEW-SESSION
       KS$RECOGNIZE-USER KS$SAVE-SESSION KS$HANDLE-FOLLOW-UP)
     (SEMANTICS
       KS$INFER-STEP-GOAL KS$INFER-DEC-START-GOAL KS$INFER-DEC-SET-GOAL KS$INFER-DEC-CHANGE-GOAL
       KS$INFER-ACCESS-GOAL KS$INFER-CURSOR-MOVE-GOAL
       KS$INFER-WANT-INFO-GOAL KS$UPDATE-USER-UNDERSTANDING KS$UPDATE-DEC-PATTERN-HIS
       KS$UPDATE-CONFIRM-PREFERENCE KS$UPDATE-STEP-CONTEXT KS$UPDATE-DECISION-CONTEXT)
     (HISTORY
       KS$INFER-SESSION-GOAL KS$RECOVER-PREVIOUS-SESSION KS$INFER-USER-PROFICIENCY
       KS$COMPUTE-INT-CHRS KS$INFER-DEVICE-PREFERENCE KS$INFER-DECISION-PATTERN)
     (UNDERSTAND KS$SUPPORT-USER-EVOLUTION)
```

```
    (USER-CHAR KS$SUPPORT-PROFICIENCY-CHANGE KS$SUPPORT-DEVICE-PREFERENCE)
    (GOAL
      KS$SUPPORT-DECISION-MAKING KS$SUPPORT-MENU-ACCESS KS$SUPPORT-ITEM-ACCESS
      KS$SUPPORT-SESSION-CONTINUE KS$UPDATE-GOAL-INFO-TABLE KS$UPDATE-FORM-PREFERENCE
      KS$UPDATE-VIEW-PREFERENCE)
    (NEED
      KS$DESIGN-INFORMATION KS$DESIGN-NETWORK-FLOW KS$RECOVER-PREVIOUS-SESSION
      KS$DECIDE-INTERACTION-MODE KS$DECIDE-ERROR-MESSAGE-P KS$DECIDE-SHOW-SUGG-MENU-P
      KS$DECIDE-MENU-POSITION KS$DESIGN-CURSOR-POSITION)
    (DESIGN
      KS$REASON-INFO-ITEM KS$DECIDE-VIEW-POINT KS$DECIDE-POP-UP KS$DECIDE-DISPLAY-FORM
      KS$INFO-DISPLAY KS$ARRANGE-INFO-DISPLAY KS$ARRANGE-NETWORK KS$ARRANGE-SYSTEM-PARAMETERS
      KS$DECIDE-INFO-MENU-LOOP-P KS$PLACE-CVV-CURSOR)))
```

2. Knowledge Sources

```
;;;
;;; Flavor for knowledge sources
;;;
(DEFFLAVOR KNOWLEDGE-SOURCE
   ((NAME NIL)
    (PROBLEM-DOMAIN NIL)
    (FUNCTION NIL)
    (DESCRIPTION NIL)
    (BB NIL)
    (TRIGGER-CONDITION NIL)
    (TRIGGER-BINDING-CONDS NIL)
    (BINDING-CHECK-CONDS NIL)
    (PRECONDITION NIL)
    (CONDITION-VARS NIL)
    (SCHEDULING-VARS NIL)
    (FROM-LEVEL NIL)
    (TO-LEVEL NIL)
    (KSAR-CONTEXTS NIL)
    (ACTION NIL))
   ()
   :GETTABLE-INSTANCE-VARIABLES
   (:INITTABLE-INSTANCE-VARIABLES NAME PROBLEM-DOMAIN FUNCTION DESCRIPTION BB TRIGGER-CONDITION
                                  PRECONDITION FROM-LEVEL TO-LEVEL ACTION))

;;; Parse trigger-condition if it contains :BIND and :TEST keyword.
(DEFMETHOD (KNOWLEDGE-SOURCE :AFTER :INIT) (&REST IGNORE)
   (IF (CONSP TRIGGER-CONDITION)
       (SETQ BINDING-CHECK-CONDS   (GETF TRIGGER-CONDITION :TEST)
             TRIGGER-BINDING-CONDS (GETF TRIGGER-CONDITION :BIND TRIGGER-CONDITION))
       (SETQ TRIGGER-BINDING-CONDS TRIGGER-CONDITION)))

;;;
;;; Check the trigger condition of a knowledge source with an event and return:
;;;    a binding list -- can be triggered and the condition variables are bound,
;;;    NIL            -- can be triggered but no bindings, or
;;;    NIX            -- can not be triggered.
;;;
(DEFMETHOD (KNOWLEDGE-SOURCE :TRIGGERED?) (EVENT)
   (COND ((NULL TRIGGER-CONDITION) NIL)
         ((EQ FROM-LEVEL (EVENT-LEVEL EVENT))
          (LET ((BINDINGS (TRIGGERED? TRIGGER-BINDING-CONDS (EVENT-MSG EVENT))))
            (IF (OR (NULL BINDING-CHECK-CONDS) (NULL BINDINGS) (EQ BINDINGS 'NIX))
                BINDINGS
                (PROGWB BINDINGS
                  (IF (*EVAL BINDING-CHECK-CONDS) BINDINGS 'NIX)))))
         (T 'NIX)))

;;;
;;; The following function returns a variable binding list.
```

```
;;; For example, '((?x A)), which means that the variable ?x has the value of 'A.
;;; This binding list is used both when testing precondition and when executing the action
;;;      as the first argument of PROGWB.
;;; This function handles AND and OR operators in the trigger conditions.
;;;
(DEFUN TRIGGERED? (CONDITION EVENT &OPTIONAL (OPERATOR NIL))
   (COND ((NULL CONDITION) (IF (EQ OPERATOR 'AND) NIL 'NIX))
         ((ATOM CONDITION) (DB-UNIFY CONDITION EVENT))
         ((EQ (CAR CONDITION) 'OR)
          (TRIGGERED? (CDR CONDITION) EVENT 'OR))
         ((EQ (CAR CONDITION) 'AND)
          (TRIGGERED? (CDR CONDITION) EVENT 'AND))
         ((CONSP (CAR CONDITION))
          (LET ((HEAD (TRIGGERED? (CAR CONDITION) EVENT)))
            (IF (EQ OPERATOR 'AND)
                (IF (EQ HEAD 'NIX)
                    'NIX
                    (LET ((TAIL (TRIGGERED? (CDR CONDITION) EVENT 'AND)))
                      (IF (EQ TAIL 'NIX) 'NIX (APPEND HEAD TAIL))))
                (IF (EQ HEAD 'NIX)
                    (LET ((TAIL (TRIGGERED? (CDR CONDITION) EVENT 'OR)))
                      (IF (EQ TAIL 'NIX) 'NIX TAIL))
                    HEAD))))
         ((ATOM (CAR CONDITION)) (DB-UNIFY CONDITION EVENT))
         (T NIL)))
```

3. Knowledge Source Activation Records (KSAR's)

```
;;;
;;; Flavor for knowledge source activation records (KSAR's)
;;;
(DEFFLAVOR KSAR
    ((NAME NIL)
     (KS NIL)
     (TRIGGER-CYCLE NIL)
     (TRIGGER-EVENT NIL)
     (TRIGGER-DECISION NIL)
     (PRECONDITION-VALUES NIL)
     (CONDITION-VALUES NIL)
     (SCHEDULING-VALUES NIL)
     (RATINGS NIL)
     (PRIORITY NIL)
     (STATUS NIL))
    ()
    :GETTABLE-INSTANCE-VARIABLES
    (:SETTABLE-INSTANCE-VARIABLES STATUS)
    :INITTABLE-INSTANCE-VARIABLES)

;;;
;;; Check precondition of a KSAR within the bindings of trigger conditions
;;;  and append the bindings of precondition to those of trigger condition bindings.
;;; Return  NIL if the condition is satisfied but no variable bindings,
;;;         NIX if the condition is not satisfied, or a variable binding list.
;;;
(DEFMETHOD (KSAR :PRECONDITION?) (&KEY (BIND T))
   (LET ((CONDS (SEND KS :PRECONDITION)))
     (COND ((OR (NULL CONDS) (EQ CONDS T)) NIL)
           (T (PROGWB CONDITION-VALUES
                (LET ((BIND-COND (AND (CONSP CONDS) (GETF CONDS :BIND)))
                      CON-VAL)
                  (IF BIND-COND
                      (PROGN (SETQ CON-VAL (*EVAL BIND-COND))
                             (OR (NULL CON-VAL)
                                 (PROGWB CON-VAL (*EVAL (GETF CONDS :TEST)))
                                 (SETQ CON-VAL NIL)))
```

```
                    (SETQ CON-VAL (*EVAL CONDS)))
                  (COND ((NULL CON-VAL) 'NIX)
                        ((CONSP CON-VAL)
                         (IF BIND
                             (DOLIST (X CON-VAL)
                               (AND (CONSP X) (IS-VAR (CAR X))
                                    (PUSH-END X CONDITION-VALUES)))
                             NIL))
                        (T CON-VAL))))))))

;;; Execute the action part of the selected KSAR within the bindings of all conditions.
(DEFMETHOD (KSAR :EXECUTE) ()
   (PROGWB CONDITION-VALUES
     (*EVAL (SEND KS :ACTION))))
```

4. Blackboards

```
;;;
;;; Flavor for blackboards
;;;
(DEFFLAVOR BLACKBOARD
   ((NAME NIL)
    (PROBLEMS NIL)
    (LEVEL-NAMES NIL)
    (MAX-LENGTH NIL)
    (CONTENTS NIL))
   ()
   (:GETTABLE-INSTANCE-VARIABLES NAME PROBLEMS LEVEL-NAMES CONTENTS)
   :INITTABLE-INSTANCE-VARIABLES)

(DEFMETHOD (BLACKBOARD :AFTER :INIT) (&REST IGNORE)
   (SEND SELF :CLEAR))

;;; Clear the blackboard. Erase all existing messages in the blackboard.
(DEFMETHOD (BLACKBOARD :CLEAR) ()
   (SETQ CONTENTS (MAKE-LIST (LENGTH LEVEL-NAMES))))

;;;
;;; Access a message in the blackboard that satisfies a given condition.
;;; A condition can have three types:
;;;   a  unifying list  -- example: '(:command ?time ?key ?device ?window ignore),
;;;   a  function       --          '(lambda (msg) (eq (input-type msg) :command)), or
;;;   an attribute list --          '(type :command).
;;; If BIND is t, return a binding list. Otherwise, return the whole message.
;;; If START is specified, search the blackboard from the STARTth message.
;;; If POS is specified, return the POSth message satisfying the condition.
;;;
(DEFMETHOD (BLACKBOARD :ACCESS-MSG) (LEVEL &OPTIONAL CONDITION &KEY (POS 0) (START 0)
                                     (BIND (IF (CONTAIN-VAR CONDITION) T NIL)) VALUE-ONLY)
   (COND ((NULL CONDITION) (NTH POS (NTHCDR START (NTH (POSITION LEVEL LEVEL-NAMES) CONTENTS))))
         ((CONTAIN-VAR CONDITION)
          (COLLECT-UNIFYING-MSGS
            (NTHCDR START (NTH (POSITION LEVEL LEVEL-NAMES) CONTENTS))
            CONDITION :BIND BIND :POS POS :VALUE-ONLY VALUE-ONLY))
         ((FUNCTIONP CONDITION)
          (COLLECT-SATISFYING-MSGS
            (NTHCDR START (NTH (POSITION LEVEL LEVEL-NAMES) CONTENTS)) CONDITION :POS POS))
         ((ATOM CONDITION)
          (COLLECT-SATISFYING-MSGS
            (NTHCDR START (NTH (POSITION LEVEL LEVEL-NAMES) CONTENTS))
            `(LAMBDA (MSG) (EQ (CAR MSG) ',CONDITION)) :POS POS))
         ((AND (= (LENGTH CONDITION) 2) (ATOM (CAR CONDITION)))
          (COLLECT-SATISFYING-MSGS
            (NTHCDR START (NTH (POSITION LEVEL LEVEL-NAMES) CONTENTS))
            (ACCESS-COND LEVEL CONDITION) :POS POS))))
```

```
;;; Access all messages in the blackboard satisfying a given condition.
;;; See the method :ACCESS-MSG.
(DEFMETHOD (BLACKBOARD :ACCESS-ALL-MSGS) (LEVEL &OPTIONAL CONDITION &KEY (START 0) END)
   (COND ((NULL CONDITION) (NTHCDR START (NTH (POSITION LEVEL LEVEL-NAMES) CONTENTS)))
         ((CONTAIN-VAR CONDITION)
          (COLLECT-UNIFYING-MSGS (NTHCDR START (NTH (POSITION LEVEL LEVEL-NAMES) CONTENTS))
                                 CONDITION :ALL T :END END))
         ((FUNCTIONP CONDITION)
          (COLLECT-SATISFYING-MSGS (NTHCDR START (NTH (POSITION LEVEL LEVEL-NAMES) CONTENTS))
                                   CONDITION :ALL T))
         ((ATOM CONDITION)
          (COLLECT-SATISFYING-MSGS
            (NTHCDR START (NTH (POSITION LEVEL LEVEL-NAMES) CONTENTS))
            `(LAMBDA (MSG) (EQ (CAR X) ,CONDITION)) :ALL T))
         ((AND (= (LENGTH CONDITION) 2) (ATOM (CAR CONDITION)))
          (COLLECT-SATISFYING-MSGS (NTHCDR START (NTH (POSITION LEVEL LEVEL-NAMES) CONTENTS))
                                   (ACCESS-COND LEVEL CONDITION) :ALL T))))

;;; Post a message on a level of the blackboard and create an event.
(DEFMETHOD (BLACKBOARD :POST-MSG) (LEVEL MSG)
   (PUSHHARD MSG (NTH (POSITION LEVEL LEVEL-NAMES) CONTENTS) MAX-LENGTH)
   (SEND BB-SCHEDULER :ADD-EVENT :TYPE :POST :LEVEL LEVEL :MSG MSG))

;;; Update an attribute value of an existing message that satisfies a given condition.
;;; For a condition, see :ACCESS-MSG.
(DEFMETHOD (BLACKBOARD :UPDATE-MSG) (LEVEL TITLE NEW &KEY OPERATION &AUX MSG)
   (IF (SETQ MSG (SEND SELF :ACCESS-MSG LEVEL TITLE :BIND NIL))
       (PROGN
         (SETF (CADR MSG) (IF OPERATION (FUNCALL OPERATION NEW (CADR MSG)) NEW))
         (SEND BB-SCHEDULER :ADD-EVENT :TYPE :UPDATE :LEVEL LEVEL :MSG MSG))
       (SEND SELF :POST-MSG LEVEL (LIST TITLE (IF (EQ OPERATION 'CONS) (LIST NEW) NEW)))))

;;;
;;; Fill in an attribute value to a message.
;;; STRC-NAME is for the case that the type of the design is different from the data structure name.
;;; Example: INFO-MENU-DESIGN uses the structure of INFO-DESIGN.
;;;
(DEFMETHOD (BLACKBOARD :FILL-MSG) (LEVEL TYPE ATTR VALUE &OPTIONAL STRC-NAME)
   (LET ((MSG (SEND SELF :ACCESS-MSG LEVEL `(TYPE ,TYPE) :BIND NIL)))
     (OR STRC-NAME (SETQ STRC-NAME TYPE))
     (WHEN MSG
       (EVAL `(SETF (,(CONCAT STRC-NAME '- ATTR) (CADR ',MSG)) ',VALUE)))))

;;; Return a lambda function that can be used to the COLLECT-SATISFYING-MSGS funtion.
;;; Example: (access-cond 'input '(type xxx)) -> '(lambda (msg) (eq (input-type msg) 'xxx)).
(DEFUN ACCESS-COND (LEVEL COND)
   `(LAMBDA (MSG) (EQ (,(CONCAT LEVEL '- (CAR COND)) MSG) ',(CADR COND))))

;;; Collect messages satisfying a given condition (function).
;;; If the key ALL is non-NIL, collect all the messages. Otherwise, collect just one message.
(DEFUN COLLECT-SATISFYING-MSGS (MSGS COND &KEY ALL (POS 0))
   (COND ((NULL MSGS) NIL)
         ((FUNCALL COND (CAR MSGS))
          (COND (ALL (CONS (CAR MSGS) (COLLECT-SATISFYING-MSGS (CDR MSGS) COND :ALL T)))
                ((= POS 0) (CAR MSGS))
                (T (COLLECT-SATISFYING-MSGS (CDR MSGS) COND :POS (1- POS)))))
         (T (COLLECT-SATISFYING-MSGS (CDR MSGS) COND :ALL ALL :POS POS))))

;;; Collect messages unifying to a given condition (list).
;;; If the key ALL is non-NIL, collect all the messages. Otherwise, collect just one message.
(DEFUN COLLECT-UNIFYING-MSGS (MSGS COND &KEY ALL BIND VALUE-ONLY (POS 0) END)
   (COND ((NULL MSGS) NIL)
         ((AND END (NOT (EQ (DB-UNIFY (CAR MSGS) END) 'NIX))) NIL)
         ((EQ (DB-UNIFY (CAR MSGS) COND) 'NIX)
          (COLLECT-UNIFYING-MSGS (CDR MSGS) COND :ALL ALL :BIND BIND :POS POS
                                 :VALUE-ONLY VALUE-ONLY :END END))
```

```
        (ALL (CONS (CAR MSGS)
                   (COLLECT-UNIFYING-MSGS (CDR MSGS) COND :ALL T :END END)))
        ((= POS 0)
         (IF BIND
             (LET ((UNIFY-LIST (DB-UNIFY (CAR MSGS) COND)))
               (IF (AND VALUE-ONLY (= (LENGTH UNIFY-LIST) 1))
                   (CADAR UNIFY-LIST)
                   UNIFY-LIST))
             (CAR MSGS)))
        (T (COLLECT-UNIFYING-MSGS (CDR MSGS) COND :ALL NIL :BIND BIND :POS (1- POS)
                                  :VALUE-ONLY VALUE-ONLY))))

;;;
;;; Functions that can be used in the specification of ACTION of a knowledge source:
;;;           POST, ACCESS, ACCESS-ALL, UPDATE, FILL.
;;;
(DEFUN POST-MSG (MSG &KEY LEVEL)
   (DECLARE (:SELF-FLAVOR KSAR))
   (OR LEVEL (SETQ LEVEL (SEND KS :TO-LEVEL)))
   (SEND (SYMBOL-VALUE (SEND KS :BB)) :POST-MSG LEVEL MSG))

(DEFUN ACCESS-MSG (LEVEL &OPTIONAL CONDITION &KEY (POS 0) (START 0)
                   (BIND (IF (CONTAIN-VAR CONDITION) T NIL)) VALUE-ONLY)
   (DECLARE (:SELF-FLAVOR KSAR))
   (SEND (SYMBOL-VALUE (SEND KS :BB)) :ACCESS-MSG LEVEL CONDITION :POS POS :BIND BIND
         :START START :VALUE-ONLY VALUE-ONLY))

(DEFUN ACCESS-ALL-MSGS (LEVEL &OPTIONAL VARIABLE CONDITION &KEY (START 0))
   (DECLARE (:SELF-FLAVOR KSAR))
   (IF VARIABLE
       (DB-UNIFY VARIABLE
                 (SEND (EVAL (SEND KS :BB)) :ACCESS-ALL-MSGS LEVEL CONDITION :START START))
       (SEND (SYMBOL-VALUE (SEND KS :BB)) :ACCESS-ALL-MSGS LEVEL CONDITION :START START)))

(DEFUN UPDATE-MSG (TITLE NEW &KEY OPERATION LEVEL)
   (DECLARE (:SELF-FLAVOR KSAR))
   (OR LEVEL (SETQ LEVEL (SEND KS :TO-LEVEL)))
   (SEND (SYMBOL-VALUE (SEND KS :BB)) :UPDATE-MSG LEVEL TITLE NEW :OPERATION OPERATION))

(DEFUN FILL-MSG (TYPE ATTR VALUE &OPTIONAL STRC-NAME &KEY LEVEL)
   (DECLARE (:SELF-FLAVOR KSAR))
   (OR STRC-NAME (SETQ STRC-NAME TYPE))
   (OR LEVEL (SETQ LEVEL (SEND KS :TO-LEVEL)))
   (SEND (SYMBOL-VALUE (SEND KS :BB)) :FILL-MSG LEVEL TYPE ATTR VALUE STRC-NAME))
```

Appendix E. Program Source Defining Knowledge Sources

```
(DEFKS ACCEPT-REPORT
   :DESCRIPTION       "accept and process the report sent by the reporters in the application process."
   :PROBLEM-DOMAIN    'OBSERVE
   :BB                'BB$MAIN
   :PRECONDITION      '*REPORT*
   :TO-LEVEL          'INPUT
   :ACTION
   '(PROGN
      (POST-MSG *REPORT*)
      (SEND BB-SCHEDULER :SET-CURRENT-PROBLEM
            (CASE (CAR *REPORT*)
              ((:NEW-SESSION :START-SESSION) P$TUNE)
              (:END-SESSION P$SAVE-HISTORY)
              (:FOLLOW-UP P$FOLLOW-UP)
              (:IDLE P$IDLE)
              (OTHERWISE P$GENERAL)))))

(DEFKS INITIALIZE-NEW-SESSION
   :DESCRIPTION       "initialize the values of the responsive layer process parameters
                           before starting a new session."
   :PROBLEM-DOMAIN    'INITIALIZE
   :BB                'BB$MAIN
   :FROM-LEVEL        'INPUT
   :TRIGGER-CONDITION '(:NEW-SESSION)
   :PRECONDITION      T
   :ACTION
   '(PROGN
      (CLEAR-BBS)
      (SETQ *CONTEXT* (MAKE-CONTEXT))
      (SETQ *USER-NAME* NIL *USER-NAME-LIST* NIL *GOAL-INFO-TABLE* NIL)
      (LOAD-SYMBOLS '(*USER-INTERACTION-SUMMARY*  *SESSION-HISTORY* *USER-GOAL-INFO-TABLE*))
      (SETQ *USER-NAME-LIST* (MAPCAR #'(LAMBDA (X) (CAR X)) *USER-INTERACTION-SUMMARY*))))

(DEFKS RECOGNIZE-USER
   :DESCRIPTION       "recognizes the user by asking the name and retrieves information
                           about the user."
   :PROBLEM-DOMAIN    'OBSERVE
   :BB                'BB$MAIN
   :FROM-LEVEL        'INPUT
   :TO-LEVEL          'HISTORY
   :TRIGGER-CONDITION '(:START-SESSION)
   :PRECONDITION      T
   :ACTION
   '(PROGN
      (SETQ *USER-NAME* (STRING-CAPITALIZE (GET-USER-NAME)))
      (IF (MEMBER *USER-NAME* *USER-NAME-LIST* :TEST #'STRING-EQUAL)
          (LET ((INT-HIST  (ASSOC *USER-NAME* *USER-INTERACTION-SUMMARY* :TEST #'EQUAL))
                (PREV-SESS (ASSOC *USER-NAME* *SESSION-HISTORY* :TEST #'EQUAL))
                (CON-INFO  (CDR (ASSOC *USER-NAME* *USER-GOAL-INFO-TABLE* :TEST #'EQUAL))))
            (WHEN INT-HIST
              (POST-MSG `(T-NO-OF-USE        ,(TOTAL-NO-OF-USE        INT-HIST)))
              (POST-MSG `(T-NO-OF-COMMANDS   ,(TOTAL-NO-OF-COMMANDS   INT-HIST)))
              (POST-MSG `(T-NO-OF-ERRORS     ,(TOTAL-NO-OF-ERRORS     INT-HIST)))
              (POST-MSG `(T-NO-OF-MOUSE-USE ,(TOTAL-NO-OF-MOUSE-USE INT-HIST))))
            (WHEN PREV-SESS
              (POST-MSG `(PREV-PROFICIENCY  ,(SESSION-PROFICIENCY PREV-SESS)) :LEVEL 'USER-CHAR)
              (POST-MSG `(DEVICE-PREFERENCE ,(SESSION-DEVICE-PREF PREV-SESS)) :LEVEL 'USER-CHAR)
              (POST-MSG `(DEC-PATTERN-HIS   ,(SESSION-DEC-PAT-HIS PREV-SESS)))
              (POST-MSG `(DECISION-PATTERN  ,(SESSION-DEC-PATTERN PREV-SESS)) :LEVEL 'USER-CHAR)
              (POST-MSG `(FORM-PREFERENCE   ,(SESSION-FORM-PREF   PREV-SESS)) :LEVEL 'USER-CHAR)
              (POST-MSG `(VIEW-PREFERENCE   ,(SESSION-VIEW-PREF   PREV-SESS)) :LEVEL 'USER-CHAR)
              (POST-MSG `(PREVIOUS-SESSION ,PREV-SESS)))
            (SETQ *GOAL-INFO-TABLE* (OR CON-INFO (COPY-ALL *GIVEN-GOAL-INFO-TABLE*))))
          (PROGN
            (POST-MSG '(T-NO-OF-USE 0))
            (SETQ *GOAL-INFO-TABLE* (COPY-ALL *GIVEN-GOAL-INFO-TABLE*))))))
```

```
(DEFKS INFER-SESSION-GOAL
   :DESCRIPTION       "infer whether the user wants to continue the previous session."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'HISTORY
   :TO-LEVEL          'GOAL
   :TRIGGER-CONDITION '(PREVIOUS-SESSION ?SESSION)
   :PRECONDITION      T
   :ACTION
   '(IF (SESSION-COMPLETE-P ?SESSION)
        (POST-MSG '(SESSION (NEW)))
        (AND (SEND ASK-CONTINUE-PREVIOUS-SESSION :ASK NIL *MSL*)
             (POST-MSG '(SESSION (CONTINUE))))))

(DEFKS SUPPORT-SESSION-CONTINUE
   :DESCRIPTION       "infer the user's needs when the user wants to continue
                           the previous session they didn't finish.
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'GOAL
   :TO-LEVEL          'NEED
   :TRIGGER-CONDITION '(SESSION (CONTINUE))
   :PRECONDITION      T
   :ACTION            '(POST-MSG '(SESSION RECOVER NIL)))

(DEFKS RECOVER-PREVIOUS-SESSION
   :DESCRIPTION       "recover the previous session the user did."
   :PROBLEM-DOMAIN    'IMPLEMENT
   :BB                'BB$MAIN
   :FROM-LEVEL        'NEED
   :TRIGGER-CONDITION '(SESSION RECOVER IGNORE)
   :PRECONDITION      '(ACCESS-MSG 'HISTORY '(PREVIOUS-SESSION ?SESSION))
   :ACTION
   '(PROGN
      (SETQ *CURRENT-DOMAIN*     (SESSION-DOMAIN            ?SESSION)
            *NEEDED-PACKAGES*    (SESSION-NEEDED-PACKAGES   ?SESSION)
            *PRIOR-DATE-LIST*    (SESSION-PRIOR-DATE-LIST   ?SESSION)
            *ORG-PROF-BASE*      (SESSION-ORG-PROF-BASE     ?SESSION)
            *DESIRED-PROF-BASE*  (SESSION-DESIRED-PROF-BASE ?SESSION)
            *SCHEDULE-BASE*      (SESSION-SCHEDULE-BASE     ?SESSION)
            *NO-OF-PRIORITY*     (LENGTH *PRIOR-DATE-LIST*)
            *TRAINEES*           (EXTRACT-ONE-FROM-LISTS *DESIRED-PROF-BASE* 0 NIL)
            *TRAIN-COMMIT-BASE*  (BUILD-TC-BASE-FROM-OP))
      (SETQ *PACKAGE-MENU* (MAKE-MENU-ITEMS *NEEDED-PACKAGES*))
      (AND *PRIOR-DATE-LIST* (UPDATE-INFO-HEADINGS))))

(DEFKS INFER-USER-PROFICIENCY
   :DESCRIPTION       "infer the user's proficiency level on MSLTRAIN from the
                           user's interaction history."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'HISTORY
   :TO-LEVEL          'USER-CHAR
   :TRIGGER-CONDITION '(T-NO-OF-USE ?NO-OF-USE)
   :PRECONDITION      '(OR (= ?NO-OF-USE 0) (ACCESS-MSG 'HISTORY '(T-ERROR-RATIO ?RATIO)))
   :ACTION
   '(IF (= ?NO-OF-USE 0)
        (POST-MSG '(PROFICIENCY :NOVICE))
      (LET ((D-GOAL (CADR (ACCESS-MSG 'GOAL 'DECISION))))
        (COND ((OR (NULL D-GOAL) (EQ D-GOAL '(START)))
               (LET ((SESSION (CADR (ACCESS-MSG 'HISTORY 'PREVIOUS-SESSION))))
                 (WHEN SESSION
                   (LET ((PREV-ERR-RATIO (COMPUTE-RATIO (SESSION-NO-OF-ERRORS   SESSION)
                                                        (SESSION-NO-OF-COMMANDS SESSION) T))
                         (BETWEEN-DAYS (ROUND (- *START-TIME* (SESSION-END-TIME SESSION))
                                              (* 24 60 60))))
                     (POST-MSG '(PROFICIENCY
```

```
                                        ,(COND ((AND (> ?NO-OF-USE  3) (< BETWEEN-DAYS 90)) :EXPERT)
                                               ((> ?NO-OF-USE  1) :REGULAR)
                                               ((OR (> PREV-ERR-RATIO 20) (> ?RATIO 30)) :NOVICE)
                                               (T :REGULAR))))))))))))

(DEFKS INFER-DEVICE-PREFERENCE
   :DESCRIPTION        "infer which input device the user prefers from the
                            user's interaction history."
   :PROBLEM-DOMAIN     'UNDERSTAND
   :BB                 'BB$MAIN
   :FROM-LEVEL         'HISTORY
   :TO-LEVEL           'USER-CHAR
   :TRIGGER-CONDITION '(T-MOUSE-USE-RATIO ?RATIO)
   :PRECONDITION       T
   :ACTION
   '(LET ((D-GOAL (CADR (ACCESS-MSG 'GOAL 'DECISION))))
      (COND ((OR (NULL D-GOAL) (EQ D-GOAL '(START)))
             (LET ((SESSION (CADR (ACCESS-MSG 'HISTORY 'PREVIOUS-SESSION))))
               (WHEN SESSION
                 (LET ((P-MOUSE-USE-RATIO (COMPUTE-RATIO (SESSION-NO-OF-MOUSE-USE SESSION)
                                                         (SESSION-NO-OF-COMMANDS  SESSION))))
                   (UPDATE-MSG 'DEVICE-PREFERENCE (IF (OR (> ?RATIO 30) (> P-MOUSE-USE-RATIO 40))
                                                      :MOUSE :KEY)))))))))

(DEFKS SUPPORT-PROFICIENCY-CHANGE
   :DESCRIPTION        "infer the user's needs when the user's proficiency has changed."
   :PROBLEM-DOMAIN     'UNDERSTAND
   :BB                 'BB$MAIN
   :FROM-LEVEL         'USER-CHAR
   :TO-LEVEL           'NEED
   :TRIGGER-CONDITION '(PROFICIENCY ?PROFICIENCY)
   :PRECONDITION       T
   :ACTION
   '(POST-MSG
      (IF (EQ (INPUT-KEY (ACCESS-MSG 'INPUT)) 'CHANGE-MODE)
          `(SET PARAMETER ((ERROR-MESSAGE-P ,?PROFICIENCY)))
          `(SET PARAMETER ((CURRENT-MODE ,?PROFICIENCY) (ERROR-MESSAGE-P ,?PROFICIENCY))))))

(DEFKS SUPPORT-DEVICE-PREFERENCE
   :DESCRIPTION        "infer the user's needs when the user has certain preference on input devices.
   :PROBLEM-DOMAIN     'UNDERSTAND
   :BB                 'BB$MAIN
   :FROM-LEVEL         'USER-CHAR
   :TO-LEVEL           'NEED
   :TRIGGER-CONDITION '(DEVICE-PREFERENCE ?DEVICE)
   :PRECONDITION       T
   :ACTION
   '(POST-MSG
      (IF (EQ (INPUT-KEY (ACCESS-MSG 'INPUT)) 'CHANGE-POS)
          `(SET PARAMETER ((SHOW-SUGG-MENU-P ,?DEVICE)))
          `(SET PARAMETER ((SHOW-SUGG-MENU-P ,?DEVICE) (MENU-POSITION ,?DEVICE))))))

(DEFKS DECIDE-INTERACTION-MODE
   :DESCRIPTION        "decide the interaction mode appropriate to the user."
   :PROBLEM-DOMAIN     'INTERPRET
   :BB                 'BB$MAIN
   :FROM-LEVEL         'NEED
   :TO-LEVEL           'INTERPRET
   :TRIGGER-CONDITION '(:BIND (SET PARAMETER ?PARAMETERS)
                        :TEST (ASSOC 'CURRENT-MODE ?PARAMETERS))
   :PRECONDITION       T
   :ACTION
   '(POST-MSG `(PARAMETER
                 (CURRENT-MODE
                   ,(CASE (CADR (ASSOC 'CURRENT-MODE ?PARAMETERS))
                      (:NOVICE  :NOVICE-MODE) (:REGULAR :REGULAR-MODE) (:EXPERT :EXPERT-MODE)
                      (:UNCLEAR (IF (SEND CONFIRM-CURRENT-MODE :ASK NIL *MSL*)
```

```
                                    :NOVICE-MODE :REGULAR-MODE))
                        (OTHERWISE :NOVICE-MODE))))))

(DEFKS DECIDE-ERROR-MESSAGE-P
   :DESCRIPTION       "decide whether to show error messages or just to beep."
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'NEED
   :TO-LEVEL          'INTERPRET
   :TRIGGER-CONDITION '(:BIND (SET PARAMETER ?PARAMETERS)
                        :TEST (ASSOC 'ERROR-MESSAGE-P ?PARAMETERS))
   :PRECONDITION      T
   :ACTION
   '(POST-MSG
      `(PARAMETER (ERROR-MESSAGE-P
                   ,(CASE (CADR (ASSOC 'ERROR-MESSAGE-P ?PARAMETERS))
                      ((:NOVICE :REGULAR :UNCLEAR) T) (:EXPERT NIL) (OTHERWISE NIL))))))

(DEFKS DECIDE-SHOW-SUGG-MENU-P
   :DESCRIPTION       "decide whether to show the information menu pane or not."
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'NEED
   :TO-LEVEL          'INTERPRET
   :TRIGGER-CONDITION '(:BIND (SET PARAMETER ?PARAMETERS)
                        :TEST (ASSOC 'SHOW-SUGG-MENU-P ?PARAMETERS))
   :PRECONDITION      '(NOT (EQ *CURRENT-MODE* :NOVICE-MODE))
   :ACTION
   '(POST-MSG `(PARAMETER (SHOW-SUGG-MENU-P
                           ,(IF (ACCESS-MSG 'INPUT '(:COMMAND ?TIME IGNORE MOUSE "SUGG" IGNORE))
                                T
                                (CASE (CADR (ASSOC 'SHOW-SUGG-MENU-P ?PARAMETERS))
                                  (:MOUSE T) (:KEY NIL) (OTHERWISE NIL)))))))

(DEFKS DECIDE-MENU-POSITION
   :DESCRIPTION       "decide the position of the guide pane."
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'NEED
   :TO-LEVEL          'INTERPRET
   :TRIGGER-CONDITION '(:BIND (SET PARAMETER ?PARAMETERS)
                        :TEST (ASSOC 'MENU-POSITION ?PARAMETERS))
   :PRECONDITION      T
   :ACTION
   '(POST-MSG `(PARAMETER (MENU-POSITION
                           ,(CASE (CADR (ASSOC 'MENU-POSITION ?PARAMETERS))
                              (:MOUSE :TOP) (:KEY :BOTTOM) (OTHERWISE :TOP))))))

(DEFKS ARRANGE-SYSTEM-PARAMETERS
   :DESCRIPTION       "set the values of system parameters."
   :PROBLEM-DOMAIN    'IMPLEMENT
   :BB                'BB$MAIN
   :FROM-LEVEL        'INTERPRET
   :TRIGGER-CONDITION '(PARAMETER (?NAME ?VALUE))
   :PRECONDITION      T
   :ACTION
   '(LET ((D-GOAL (CADR (ACCESS-MSG 'GOAL 'DECISION))))
      (IF (OR (NULL D-GOAL) (EQ D-GOAL '(START)))
          (SET (CONCAT '* ?NAME '*) ?VALUE)
          (COND ((EQ ?NAME 'SHOW-SUGG-MENU-P)
                 (OR (EQ ?VALUE *SHOW-SUGG-MENU-P*)
                     T))
                ((EQ ?NAME 'CURRENT-MODE)
                 (OR (EQ *CURRENT-MODE* ?VALUE)
                     (IF (EQ ?VALUE :EXPERT-MODE)
                         (SEND *MSL* :EXPERT-MODE)
                         (SEND *MSL* :CHANGE-MODE))))
```

```
                ((EQ ?NAME 'MENU-POSITION)
                 (OR (EQ *MENU-POSITION* ?VALUE)
                     (SEND *MSL* :CHANGE-CONFIGURATION)))
                (T (SET (CONCAT '* ?NAME '*) ?VALUE))))))

(DEFKS UPDATE-INT-HISTORY
   :DESCRIPTION       "update interaction history."
   :PROBLEM-DOMAIN    'OBSERVE
   :BB                'BB$MAIN
   :FROM-LEVEL        'INPUT
   :TO-LEVEL          'HISTORY
   :TRIGGER-CONDITION '(:BIND (?TYPE IGNORE IGNORE ?DEVICE IGNORE IGNORE)
                        :TEST (MEMBER ?TYPE '(:COMMAND :ERROR :CONTINUE)))
   :PRECONDITION      T
   :ACTION
   '(PROGN
      (COND ((EQ ?TYPE :COMMAND)
             (UPDATE-MSG 'T-NO-OF-COMMANDS 1 :OPERATION '+)
             (UPDATE-MSG 'S-NO-OF-COMMANDS 1 :OPERATION '+))
            ((EQ ?TYPE :ERROR)
             (UPDATE-MSG 'T-NO-OF-ERRORS 1 :OPERATION '+)
             (UPDATE-MSG 'S-NO-OF-ERRORS 1 :OPERATION '+)))
      (WHEN (EQ ?DEVICE 'MOUSE)
        (UPDATE-MSG 'T-NO-OF-MOUSE-USE 1 :OPERATION '+)
        (UPDATE-MSG 'S-NO-OF-MOUSE-USE 1 :OPERATION '+))))

(DEFKS COMPUTE-INT-CHRS
   :DESCRIPTION       "update the user's interaction characteristics."
   :PROBLEM-DOMAIN    'OBSERVE
   :BB                'BB$MAIN
   :FROM-LEVEL        'HISTORY
   :TO-LEVEL          'HISTORY
   :TRIGGER-CONDITION '(:BIND (?VAR IGNORE)
                        :TEST (MEMBER ?VAR
                                      '(T-NO-OF-ERRORS T-NO-OF-MOUSE-USE)))
   :PRECONDITION      T
   :ACTION
   '(LET ((T-NO-OF-COMMANDS (CADR (ACCESS-MSG 'HISTORY 'T-NO-OF-COMMANDS)))
          (S-NO-OF-COMMANDS (CADR (ACCESS-MSG 'HISTORY 'S-NO-OF-COMMANDS))))
      (WHEN (AND T-NO-OF-COMMANDS (> T-NO-OF-COMMANDS 0))
        (COND ((EQ ?VAR 'T-NO-OF-ERRORS)
               (UPDATE-MSG 'T-ERROR-RATIO
                           (COMPUTE-RATIO (CADR (ACCESS-MSG 'HISTORY 'T-NO-OF-ERRORS))
                                          T-NO-OF-COMMANDS T))
               (WHEN (AND S-NO-OF-COMMANDS (> S-NO-OF-COMMANDS 0))
                 (UPDATE-MSG 'S-ERROR-RATIO
                             (COMPUTE-RATIO (CADR (ACCESS-MSG 'HISTORY 'S-NO-OF-ERRORS))
                                            S-NO-OF-COMMANDS T))))
              ((EQ ?VAR 'T-NO-OF-MOUSE-USE)
               (UPDATE-MSG 'T-MOUSE-USE-RATIO
                           (COMPUTE-RATIO (CADR (ACCESS-MSG 'HISTORY 'T-NO-OF-MOUSE-USE))
                                          T-NO-OF-COMMANDS))
               (WHEN (AND S-NO-OF-COMMANDS (> S-NO-OF-COMMANDS 0))
                 (UPDATE-MSG 'S-MOUSE-USE-RATIO
                             (COMPUTE-RATIO (CADR (ACCESS-MSG 'HISTORY 'S-NO-OF-MOUSE-USE))
                                            S-NO-OF-COMMANDS)))))))))

(DEFKS INFER-TOP-SEMANTICS
   :DESCRIPTION       "infer what the user's input means at the top level commands."
   :PROBLEM-DOMAIN    'OBSERVE
   :BB                'BB$MAIN
   :FROM-LEVEL        'INPUT
   :TO-LEVEL          'SEMANTICS
   :TRIGGER-CONDITION '(:BIND (:COMMAND IGNORE ?KEY IGNORE "TOP" ?NODE)
                        :TEST (NOT (MEMBER ?KEY *ACCESS-KEYS*)))
   :PRECONDITION      T
   :ACTION
```

```
    '(COND ((EQ *CURRENT-MODE* :NOVICE-MODE)
            (LET (IN)
              (COND ((OR (EQ ?KEY 'RIGHT) (EQ ?KEY 'SPACE))
                     (SETQ IN (IF *NOT-COMPLETE-STEP-P* 2 1)))
                    ((EQ ?KEY 'LEFT)  (SETQ IN 2))
                    ((EQ ?KEY 'UP)    (SETQ IN 3))
                    ((NUMBERP ?KEY)   (SETQ IN 'JUMP)))
              (WHEN IN
                (UPDATE-MSG 'NODES (SEND ?NODE :NAME) :OPERATION 'CONS :LEVEL 'HISTORY)
                (OR (EQ (SEND ?NODE :TYPE) :MAIN-STEP)
                    (POST-MSG `(,(CASE IN (1 'CONTINUE) (2 'GO-BACK) (3 'QUICK-BACK) (JUMP 'JUMP))
                                NODE ,?NODE)))
                (LET (NEXT-STEP
                      (NEXT-NODE (IF (EQ IN 'JUMP)
                                     (LET ((NEXT (GET-FIRST-NODE-IN-STEP ?KEY)))
                                       (AND (MEMBER NEXT *THRU-NODES*) NEXT))
                                     (SEND ?NODE :NEXT IN))))
                  (WHEN NEXT-NODE
                    (SETQ NEXT-STEP (SEND (SYMBOL-VALUE NEXT-NODE) :STEP))
                    (AND (EQ NEXT-STEP T) (SETQ NEXT-STEP *CURRENT-STEP*))
                    (CASE (SEND (SYMBOL-VALUE NEXT-NODE) :TYPE)
                      (:MAIN-STEP
                       (POST-MSG `(START STEP ,NEXT-STEP))
                       (UPDATE-MSG 'STEPS NEXT-STEP :OPERATION 'CONS :LEVEL 'HISTORY))
                      (:RESULT-DISPLAY
                       (POST-MSG
                         `(INFO-ITEM SET
                           (:ITEM ,(STEP-DECISION-INFO (ASSOC NEXT-STEP *STEP-ASSOCIATES*)))
                           :DISPLAY-FORM)) :LEVEL 'NEED))
                      (:CONSQ-DISPLAY
                       (POST-MSG
                         `(INFO-ITEM SET
                           (:ITEM ,(STEP-CONSQ-INFO (ASSOC NEXT-STEP *STEP-ASSOCIATES*)))
                           :DISPLAY-FORM)) :LEVEL 'NEED))))))))
           ((OR (EQ *CURRENT-MODE* :REGULAR-MODE) (EQ *CURRENT-MODE* :EXPERT-MODE))
            (COND ((EQ ?KEY 'END) (POST-MSG '(END SESSION NIL)))
                  ((NUMBERP ?KEY)
                   (LET ((REQ-BASE (STEP-REQUIRED-BASE (ASSOC ?KEY *STEP-ASSOCIATES*))))
                     (WHEN (IF (ATOM REQ-BASE) (SYMBOL-VALUE REQ-BASE)
                               (DOLIST (X REQ-BASE T) (OR (SYMBOL-VALUE X) (RETURN NIL))))
                       (POST-MSG `(START STEP ,?KEY))
                       (UPDATE-MSG 'STEPS ?KEY :OPERATION 'CONS :LEVEL 'HISTORY))))))))

(DEFKS INFER-STEP-SEMANTICS
   :DESCRIPTION       "infer what the user's input means at the step level commands."
   :PROBLEM-DOMAIN    'OBSERVE
   :BB                'BB$MAIN
   :FROM-LEVEL        'INPUT
   :TO-LEVEL          'SEMANTICS
   :TRIGGER-CONDITION '(:BIND (:COMMAND IGNORE ?KEY IGNORE ?WINDOW ?REFERENCE)
                       :TEST (AND (EQ (WINDOW-TYPE ?WINDOW) :STEP)
                                  (NOT (MEMBER ?KEY *ACCESS-KEYS*))))
   :PRECONDITION      T
   :ACTION
   '(COND ((OR (EQ ?KEY 'END) (EQ ?KEY 'QUIT))
           (COND ((OR (STRING-EQUAL (NSUBSTRING ?WINDOW 0 4) "MENU") (EQUAL ?WINDOW "CVV 2"))
                  (POST-MSG `(,?KEY STEP NIL)))
                 ((EQUAL (NSUBSTRING ?WINDOW 0 3) "CVV")
                  (IF (EQ ?KEY 'END)
                      (POST-MSG '(END DECISION NIL))
                      (POST-MSG `(CONTINUE STEP ,(CONTEXT-STEP)))))))
          ((EQ ?KEY 'SELECT)
           (AND (EQUAL (NSUBSTRING ?WINDOW 0 4) "MENU")
             (POST-MSG `(START DECISION ,?REFERENCE))))
          ((MEMBER ?KEY '(UP DOWN))
           (AND (EQUAL (NSUBSTRING ?WINDOW 0 3) "CVV")
                (POST-MSG `(CURSOR ,?KEY ,?REFERENCE))))
```

```
        ((MEMBER ?KEY '(SET-VALUE CHANGE-VALUE))
         (AND (EQUAL (NSUBSTRING ?WINDOW 0 3) "CVV")
              (NOT (MEMBER (CADR ?REFERENCE)
                           (CDR (ASSOC (SEND (CONTEXT-PERSON) :NAME)
                                       *INDI-BAD-MONTHS* :TEST #'EQUAL))))
              (POST-MSG `(,(CASE ?KEY
                                 (SET-VALUE 'SET) (CHANGE-VALUE 'CHANGE))
                          DECISION ,?REFERENCE))))))

(DEFKS INFER-ACCESS-SEMANTICS
   :DESCRIPTION       "infer what the user's input means when the input is related
                           to information access."
   :PROBLEM-DOMAIN    'OBSERVE
   :BB                'BB$MAIN
   :FROM-LEVEL        'INPUT
   :TO-LEVEL          'SEMANTICS
   :TRIGGER-CONDITION '(:BIND (?TYPE ?TIME ?KEY IGNORE ?WINDOW ?ITEM)
                        :TEST (OR (AND (EQ ?TYPE :CONTINUE) (EQ ?KEY 'SPACE))
                                  (AND (EQ ?TYPE :COMMAND)
                                       (OR (MEMBER ?KEY *ACCESS-KEYS*)
                                           (EQ (WINDOW-TYPE ?WINDOW) :ACCESS)))))
   :PRECONDITION      T
   :ACTION
   '(COND ((EQ ?TYPE 'CONTINUE)
           (POST-MSG `(QUIT ACCESS ,?ITEM)))
          ((MEMBER ?KEY *ACCESS-KEYS* :TEST #'EQUAL)
           (POST-MSG `(WANT ,?KEY NIL)))
          ((EQ ?KEY 'QUIT)
           (POST-MSG '(STOP ACCESS NIL)))
          ((EQ ?KEY 'SELECT)
           (WHEN (EQUAL ?WINDOW "Package Menu")
             (POST-MSG `(SET ACCESS-PKG ,?ITEM))))
          ((AND (CONSP ?KEY) (EQ (CAR ?KEY) 'SELECT))
           (LET ((SELECT-TYPE (CADR ?KEY)))
             (COND ((EQ ?ITEM :OTHERS) (POST-MSG '(WANT :OTHERS NIL)))
                   (T (POST-MSG `(REQUEST ,(SYMBOL-VALUE ?ITEM) ,SELECT-TYPE))
                      (LET ((GOAL-FLAGS (BUILD-FLAGS-FROM-GOAL (CDR (ACCESS-MSG 'GOAL 'DECISION)))))
                        (PUSH (LIST ?TIME GOAL-FLAGS ?ITEM
                                    (AND (>= SELECT-TYPE 1) (IF (>= SELECT-TYPE 3) 0 1))
                                    (AND (>= SELECT-TYPE 1) (IF (EVENP SELECT-TYPE) 1 0)))
                              *ACCESS-INFO-HISTORY*))))))))

(DEFKS HANDLE-FOLLOW-UP
   :DESCRIPTION       "handle when the application process ends updating the database
                           about the user's decision."
   :PROBLEM-DOMAIN    'OBSERVE
   :BB                'BB$MAIN
   :FROM-LEVEL        'INPUT
   :TO-LEVEL          'SEMANTICS
   :TRIGGER-CONDITION '(:FOLLOW-UP CONFIRM)
   :PRECONDITION      '(CONTEXT-PERSON)
   :ACTION            '(POST-MSG `(CONTINUE STEP ,(CONTEXT-STEP))))

(DEFKS INFER-CONFIRM-SEMANTICS
   :DESCRIPTION       "infer what the user's input means when the user responded on
                           the "YES" "NO" question of the system."
   :PROBLEM-DOMAIN    'OBSERVE
   :BB                'BB$MAIN
   :FROM-LEVEL        'INPUT
   :TO-LEVEL          'SEMANTICS
   :TRIGGER-CONDITION '(:BIND (:CONTINUE ?TIME ?KEY IGNORE IGNORE IGNORE)
                        :TEST (MEMBER ?KEY '(YES NO)))
   :PRECONDITION      '(ACCESS-MSG 'INPUT '(:COMMAND ?PREV-TIME ?PREV-KEY IGNORE IGNORE IGNORE))
   :ACTION
   '(WHEN (MEMBER ?PREV-KEY '(END QUIT))
      (POST-MSG (LIST ?KEY ?PREV-KEY
                      (IF (AND (EQ ?KEY 'NO) (< (- ?TIME ?PREV-TIME) 150))
```

```
                        'NO-USE 'USE)))
      (IF (AND (EQ ?KEY 'YES) (EQ ?PREV-KEY 'END))
          (POST-MSG `(CONTINUE DECISION ,(CONTEXT-PERSON)))
          (POST-MSG `(CONTINUE STEP     ,(CONTEXT-STEP))))))

(DEFKS UPDATE-CONFIRM-PREFERENCE
   :DESCRIPTION       "update the user's preference on portrayal format."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'USER-CHAR
   :TRIGGER-CONDITION '(YES END NO-USE)
   :PRECONDITION      T
   :ACTION            '(POST-MSG '(CONFIRM-PREFERENCE :NO)))

(DEFKS UPDATE-DEC-PATTERN-HIS
   :DESCRIPTION       "update the user's decision pattern history."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'HISTORY
   :TRIGGER-CONDITION '(SET DECISION IGNORE)
   :PRECONDITION      '(APPLY-ALL (CONTEXT-CURRENT-STAT) '>= 2)
   :ACTION
   '(LET ((CURRENT-PATTERNS (COPY-ALL (CADR (ACCESS-MSG 'HISTORY 'DECISION-PATTERN))))
          PATTERN
          (S-STAT (CONTEXT-START-STAT)))
      (WHEN (AND S-STAT (APPLY-ALL S-STAT '= 0))
        (LET* ((DECISIONS (MAPCAR #'(LAMBDA (X) (CAR (THIRD X)))
                                  (SEND BB$MAIN :ACCESS-ALL-MSGS 'SEMANTICS '(SET DECISION ?Z)
                                        :END '(START DECISION ?A))))
               (PKG-MODES (EXTRACT-MODES (EXTRACT-PKGS DECISIONS) NIL))
               (PKGS (CONTEXT-PKGS)))
          (COND ((EQUAL PKGS (REVERSE PKG-MODES)) (SETQ PATTERN :PKG-DOWN))
                ((EQUAL PKGS PKG-MODES) (SETQ PATTERN :PKG-UP))
                ((= (LENGTH PKGS) (LENGTH PKG-MODES)) (SETQ PATTERN :PKG))
                (T (SETQ PATTERN :RANDOM)))
          (IF CURRENT-PATTERNS
              (LET ((HIST (ASSOC PATTERN CURRENT-PATTERNS)))
                (IF HIST
                    (INCREMENT (CDR HIST))
                    (PUSH (CONS PATTERN 1) CURRENT-PATTERNS))
                (UPDATE-MSG 'DECISION-PATTERN CURRENT-PATTERNS))
              (POST-MSG `(DECISION-PATTERN (,(CONS PATTERN 1))))))))

(DEFKS INFER-DECISION-PATTERN
   :DESCRIPTION       "infer the user's decision pattern from the user's interaction history."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'HISTORY
   :TO-LEVEL          'USER-CHAR
   :TRIGGER-CONDITION '(DECISION-PATTERN ?PATTERN)
   :PRECONDITION      T
   :ACTION
   '(LET ((MAX-P 0) MAX-ITEM)
      (DOLIST (X ?PATTERN)
        (WHEN (> (CDR X) MAX-P)
          (SETQ MAX-P (CDR X) MAX-ITEM (CAR X))))
      (UPDATE-MSG 'DECISION-PATTERN MAX-ITEM)))

(DEFKS UPDATE-STEP-CONTEXT
   :DESCRIPTION       "update the context of the user's decisions when the user starts
                           or continues a step."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'CONTEXT
```

```
   :TRIGGER-CONDITION '(:BIND (?X STEP ?STEP)
                       :TEST (MEMBER ?X '(START CONTINUE)))
   :PRECONDITION      T
   :ACTION
   '(WHEN (= ?STEP 5)
      (COND ((OR (EQ ?X 'START) (EQ ?X 'CONTINUE))
             (AND (EQ ?X 'START) (SETF (CONTEXT-STEP) ?STEP))
             (LET* ((VIEW-PREF (CADR (ACCESS-MSG 'USER-CHAR 'VIEW-PREFERENCE)))
                    (W-PREF (CDR (ASSOC :WHOLE VIEW-PREF)))
                    (P-PREF (CDR (ASSOC :PART  VIEW-PREF)))
                    (FORM-PREF (CADR (ACCESS-MSG 'USER-CHAR 'FORM-PREFERENCE)))
                    (T-PREF (CDR (ASSOC :TABLE FORM-PREF)))
                    (G-PREF (CDR (ASSOC :GRAPH FORM-PREF))))
               (SETF (CONTEXT-VIEW) (IF (AND W-PREF (> W-PREF (OR P-PREF 0))) :WHOLE :PART))
               (SETF (CONTEXT-FORM) (IF (AND G-PREF (> G-PREF (OR T-PREF 0))) :GRAPH :TABLE)))
             (MULTIPLE-VALUE-BIND (X Y) (OP-CONST-OK?)
               (SETF (CONTEXT-COMP-STAT) X)
               (SETF (CONTEXT-MISS-SESSIONS) Y))
             (MULTIPLE-VALUE-BIND (X Y) (DUBL-CONST-OK?)
               (SETF (CONTEXT-DUBL-STAT) X)
               (SETF (CONTEXT-MISS-PERSONS) Y))
             (MULTIPLE-VALUE-BIND (X Y) (DAYS-CONST-OK?)
               (SETF (CONTEXT-DAYS-STAT) X)
               (SETF (CONTEXT-MISS-MONTHS) Y))
             (SETF (CONTEXT-PKGS)   NIL)
             (SETF (CONTEXT-IN-PKG)  NIL)
             (SETF (CONTEXT-IN-LEVEL)  NIL)
             (SETF (CONTEXT-SESSIONS)     NIL)
             (SETF (CONTEXT-START-STAT)   NIL)
             (SETF (CONTEXT-CURRENT-STAT) NIL)
             (SETF (CONTEXT-PERSON) NIL)
             (SETF (CONTEXT-PKG)    NIL)
             (SETF (CONTEXT-LEVEL)    NIL)
             (POST-MSG (COPY-ALL *CONTEXT*)))
            ((EQ ?X 'QUIT)
             (SETF (CONTEXT-STEP)      NIL)
             (SETF (CONTEXT-COMP-STAT) NIL)
             (SETF (CONTEXT-DUBL-STAT) NIL)
             (SETF (CONTEXT-DAYS-STAT) NIL)
             (POST-MSG (COPY-ALL *CONTEXT*))))))

(DEFKS UPDATE-DECISION-CONTEXT
   :DESCRIPTION       "infer the context of the user's decisions when the user starts
                           or continues a decision in a step."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'CONTEXT
   :TRIGGER-CONDITION '(?X DECISION ?Y)
   :PRECONDITION      '(CONTEXT-STEP)
   :ACTION
   '(WHEN (= (CONTEXT-STEP) 5)
      (COND ((OR (EQ ?X 'START) (EQ ?X 'CONTINUE))
             (WHEN (EQ ?X 'START)
               (SCH-SETUP ?Y)
               (SETF (CONTEXT-PERSON) ?Y)
               (MULTIPLE-VALUE-BIND (X Y) (COLLECT-PKGS-SESSIONS *CHOOSE-VALUES*)
                 (SETF (CONTEXT-PKGS) X)
                 (SETF (CONTEXT-SESSIONS) Y)))
             (SETF (CONTEXT-START-STAT) (SCH-DECISION-STATUS *CHOOSE-VALUES*))
             (SETF (CONTEXT-CURRENT-STAT) (CONTEXT-START-STAT))
             (POST-MSG (COPY-ALL *CONTEXT*)))
            ((MEMBER ?X '(SET CHANGE))
             (LET* ((PKG (GET-PKG-FROM-SESSION (CAR ?Y)))
                    (LEVEL (GET-LEVEL-FROM-SESSION (CAR ?Y)))
                    (PKG-POS (POSITION PKG (CONTEXT-PKGS) :TEST #'EQUAL)))
               (SETF (CONTEXT-IN-PKG) PKG)
```

```
            (SETF (CONTEXT-IN-LEVEL) LEVEL)
            (SETF (CADR (ASSOC LEVEL (NTH PKG-POS (CONTEXT-SESSIONS))))
                  (GET-MILE-OF-A-MONTH (CADR ?Y)))
            (WHEN (EQ ?X 'SET)
              (IF (AND (>= (NTH PKG-POS (CONTEXT-CURRENT-STAT)) 2) (PLUSP (CADR ?Y)))
                  (AND (APPLY-ALL (CONTEXT-CURRENT-STAT) '>= 2)
                       (SETF (NTH PKG-POS (CONTEXT-START-STAT)) 2))
                  (SETF (CONTEXT-CURRENT-STAT) (SCH-DECISION-STATUS *CHOOSE-VALUES*)))))
          (MULTIPLE-VALUE-BIND (X Y) (OP-CONST-OK?)
            (SETF (CONTEXT-COMP-STAT) X)
            (SETF (CONTEXT-MISS-SESSIONS) Y))
          (MULTIPLE-VALUE-BIND (X Y) (DUBL-CONST-OK?)
            (SETF (CONTEXT-DUBL-STAT) X)
            (SETF (CONTEXT-MISS-PERSONS) Y))
          (MULTIPLE-VALUE-BIND (X Y) (DAYS-CONST-OK?)
            (SETF (CONTEXT-DAYS-STAT) X)
            (SETF (CONTEXT-MISS-MONTHS) Y))
          (POST-MSG (COPY-ALL *CONTEXT*))))))

(DEFKS INFER-STEP-GOAL
   :DESCRIPTION       "infer the user's goal when the user starts or continues a step."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'GOAL
   :TRIGGER-CONDITION '(?X STEP ?Z)
   :PRECONDITION      '(CONTEXT-COMP-STAT)
   :ACTION
   '(COND ((OR (EQ ?X 'START) (EQ ?X 'CONTINUE))
           (LET (PURPOSE)
             (COND ((APPLY-ALL (CONTEXT-COMP-STAT) '<= 1) (SETQ PURPOSE 0))
                   ((APPLY-ALL (CONTEXT-COMP-STAT) '>= 4)
                    (COND ((= (CONTEXT-DAYS-STAT) 0) (SETQ PURPOSE 3))
                          ((= (CONTEXT-DUBL-STAT) 0) (SETQ PURPOSE 2))
                          (T (SETQ PURPOSE 0))))
                   ((APPLY-ONE (CONTEXT-COMP-STAT) '>= 2) (SETQ PURPOSE 1))
                   (T (SETQ PURPOSE 0)))
             (POST-MSG `(DECISION 0 ,PURPOSE))))
          ((EQ ?X 'QUIT)
           (POST-MSG `(CONTINUE ,(CONTEXT-STEP)
                                ,(IF (APPLY-ALL (OP-CONST-OK?) '>= 4) 1 0)))))))

(DEFKS INFER-DEC-START-GOAL
   :DESCRIPTION       "infer the user's goal when the user started a decision in a step."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'GOAL
   :TRIGGER-CONDITION '(:BIND (?X DECISION ?ITEM)
                       :TEST (MEMBER ?X '(START CONTINUE)))
   :PRECONDITION      '(CONTEXT-START-STAT)
   :ACTION
   '(LET ((PURPOSE (THIRD (ACCESS-MSG 'GOAL 'DECISION)))
          (PATTERN (CADR (ACCESS-MSG 'USER-CHAR 'DECISION-PATTERN)))
          PKG C-MISS-PKG)
      (COND ((SETQ C-MISS-PKG (GET-OP-MISS-PKG))
             (SETF (CONTEXT-PKG) (CAR C-MISS-PKG))
             (SETF (CONTEXT-LEVEL) (CADR C-MISS-PKG))
             (POST-MSG `(DECISION ,(IF (CONTEXT-PKG) 1 2) 1)))
            ((MEMBER (CONTEXT-PERSON) (CONTEXT-MISS-PERSONS))
             (POST-MSG '(DECISION 2 2)))
            ((SETQ PKG (GET-UNSCHEDULED-PKG))
             (SETF (CONTEXT-PKG) (CAR PKG))
             (SETF (CONTEXT-LEVEL) (CADR PKG))
             (POST-MSG `(DECISION ,(IF (CONTEXT-PKG) 1 2) 0)))
            (T (SETF (CONTEXT-PKG) NIL)
               (POST-MSG `(DECISION 2 3)))))))
```

```
(DEFKS INFER-DEC-SET-GOAL
   :DESCRIPTION       "infer the user's goal when the user made a decision in a step."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'GOAL
   :TRIGGER-CONDITION '(SET DECISION ?ITEM)
   :PRECONDITION      '(CONTEXT-START-STAT)
   :ACTION
   '(LET* ((GOAL (ACCESS-MSG 'GOAL 'DECISION))
           (PURPOSE (THIRD GOAL))
           (KIND    (SECOND GOAL))
           (PATTERN (CADR (ACCESS-MSG 'USER-CHAR 'DECISION-PATTERN)))
           (PKG     (CONTEXT-IN-PKG))
           (LEVEL   (CONTEXT-IN-LEVEL))
           (PKG-POS (POSITION PKG (CONTEXT-PKGS) :TEST #'EQUAL))
           UNSCH-PKG C-MISS-PKG)
      (COND ((INTERSECTION (NTH PKG-POS (CONTEXT-SESSIONS))
                           (NTH (POSITION PKG *NEEDED-PACKAGES* :TEST #'EQUAL)
                                (CONTEXT-MISS-SESSIONS))
                           :TEST #'EQUAL)
             (SETF (CONTEXT-PKG) PKG)
             (SETF (CONTEXT-LEVEL) LEVEL)
             (NORMAL-BEEP)
             (POST-MSG '(DECISION 1 1)))
            ((CONTAIN-SAME-MONTH? ?ITEM)
             (NORMAL-BEEP) (SETF (CONTEXT-PKG) NIL)
             (POST-MSG '(DECISION 2 2)))
            ((SETF (CONTEXT-LEVEL)
                   (DOLIST (X (NTH PKG-POS (CONTEXT-SESSIONS)))
                     (WHEN (ZEROP (CADR X)) (RETURN (CAR X)))))
             (SETF (CONTEXT-PKG) PKG)
             (POST-MSG '(DECISION 1 0)))
            ((SETQ C-MISS-PKG (GET-OP-MISS-PKG))
             (SETF (CONTEXT-PKG) (CAR C-MISS-PKG))
             (SETF (CONTEXT-LEVEL) (CADR C-MISS-PKG))
             (POST-MSG `(DECISION ,(IF (CONTEXT-PKG) 1 2) 1)))
            ((AND (NOT (APPLY-ALL (CONTEXT-START-STAT) '>= 2))
                  (APPLY-ALL (CONTEXT-CURRENT-STAT) '>= 2))
             (SETF (CONTEXT-PKG) NIL)
             (COND ((AND (CONTEXT-MISS-MONTHS) (CONTAIN-MISS-MONTH?)) (POST-MSG '(DECISION 2 3)))
                   ((= (CONTEXT-DUBL-STAT) 0) (POST-MSG '(DECISION 2 2)))
                   (T (POST-MSG `(DECISION 3 0)))))
            (T
             (WHEN (>= (NTH PKG-POS (CONTEXT-CURRENT-STAT)) 2)
               (IF (CADR (ACCESS-MSG 'USER-CHAR 'CONFIRM-PREFERENCE))
                   (PROGN (SETF (CONTEXT-PKG) PKG)
                          (SETF (CONTEXT-LEVEL) LEVEL)
                          (POST-MSG '(DECISION 1 0)))
                   (WHEN (SETQ UNSCH-PKG (GET-UNSCHEDULED-PKG))
                     (SETF (CONTEXT-PKG) (CAR UNSCH-PKG))
                     (SETF (CONTEXT-LEVEL) (CADR UNSCH-PKG))
                     (POST-MSG `(DECISION ,(IF (CONTEXT-PKG) 1 2) 0)))))))))

(DEFKS INFER-DEC-CHANGE-GOAL
   :DESCRIPTION       "infer the user's goal when the user changed a decision in a step."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'GOAL
   :TRIGGER-CONDITION '(CHANGE DECISION ?ITEM)
   :PRECONDITION      '(CONTEXT-START-STAT)
   :ACTION
   '(LET* ((GOAL (ACCESS-MSG 'GOAL 'DECISION))
           (PURPOSE (THIRD GOAL))
           (KIND    (SECOND GOAL))
```

```
          (PATTERN (CADR (ACCESS-MSG 'USER-CHAR 'DECISION-PATTERN)))
          (PKG     (CONTEXT-IN-PKG))
          (LEVEL   (CONTEXT-IN-LEVEL))
          (PKG-POS (POSITION PKG (CONTEXT-PKGS) :TEST #'EQUAL))
          UNSCH-PKG MISS-PKG)
      (COND ((INTERSECTION (NTH PKG-POS (CONTEXT-SESSIONS))
                           (NTH (POSITION PKG *NEEDED-PACKAGES* :TEST #'EQUAL)
                                (CONTEXT-MISS-SESSIONS))
                           :TEST #'EQUAL)
             (SETF (CONTEXT-PKG) PKG)
             (SETF (CONTEXT-LEVEL) LEVEL)
             (NORMAL-BEEP)
             (POST-MSG `(DECISION 1 1)))
            ((CONTAIN-SAME-MONTH? ?ITEM)
             (NORMAL-BEEP) (SETF (CONTEXT-PKG) NIL)
             (POST-MSG '(DECISION 2 2)))
            ((SETF (CONTEXT-LEVEL)
                   (DOLIST (X (NTH PKG-POS (CONTEXT-SESSIONS)))
                     (WHEN (ZEROP (CADR X)) (RETURN (CAR X)))))
             (SETF (CONTEXT-PKG) PKG)
             (POST-MSG '(DECISION 1 0)))
            ((SETQ MISS-PKG (GET-OP-MISS-PKG))
             (SETF (CONTEXT-PKG) (CAR MISS-PKG))
             (SETF (CONTEXT-LEVEL) (CADR MISS-PKG))
             (POST-MSG `(DECISION ,(IF (CONTEXT-PKG) 1 2) 1)))
            ((AND (NOT (APPLY-ALL (CONTEXT-START-STAT) '>= 2))
                  (APPLY-ALL (CONTEXT-CURRENT-STAT) '>= 2))
             (SETF (CONTEXT-PKG) NIL)
             (COND ((AND (CONTEXT-MISS-MONTHS) (CONTAIN-MISS-MONTH?)) (POST-MSG '(DECISION 2 3)))
                   ((= (CONTEXT-DUBL-STAT) 0) (POST-MSG '(DECISION 2 2)))
                   (T (POST-MSG `(DECISION 3 0)))))
            ((=  KIND 3) (POST-MSG `(DECISION 3 ,PURPOSE)))
            (T
             (WHEN (>= (NTH PKG-POS (CONTEXT-CURRENT-STAT)) 2)
               (IF (CADR (ACCESS-MSG 'USER-CHAR 'CONFIRM-PREFERENCE))
                   (PROGN (SETF (CONTEXT-PKG) PKG)
                          (SETF (CONTEXT-LEVEL) LEVEL)
                          (POST-MSG '(DECISION 1 0)))
                   (WHEN (SETQ UNSCH-PKG (GET-UNSCHEDULED-PKG))
                     (SETF (CONTEXT-PKG) (CAR UNSCH-PKG))
                     (SETF (CONTEXT-LEVEL) (CADR UNSCH-PKG))
                     (POST-MSG `(DECISION ,(IF (CONTEXT-PKG) 1 2) 0)))))))))

(DEFKS INFER-CURSOR-MOVE-GOAL
   :DESCRIPTION       "infer the user's goal when the user moved the cursor."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'GOAL
   :TRIGGER-CONDITION '(CURSOR ?DIRECTION ?LINE-NO)
   :PRECONDITION      '(CONTEXT-PKG)
   :ACTION
   '(MULTIPLE-VALUE-BIND (PKG PKG-POS)
        (DO ((X (CONTEXT-PKGS) (CDR X))
             (Y (CONTEXT-SESSIONS) (CDR Y))
             (SESSION-LENGTH 0)
             (LINE 3 (+ LINE SESSION-LENGTH 2))
             (N 0 (1+ N)))
            ((ENDP X))
          (SETQ SESSION-LENGTH (LENGTH (CAR Y)))
          (WHEN (AND (>= ?LINE-NO LINE) (<= ?LINE-NO (+ LINE SESSION-LENGTH)))
            (RETURN (VALUES (CAR X) N))))
      (WHEN (AND PKG (NOT (EQUAL PKG (CONTEXT-PKG))))
        (SETF (CONTEXT-PKG) PKG)
        (COND ((INTERSECTION (NTH PKG-POS (CONTEXT-SESSIONS))
                             (NTH (POSITION PKG *NEEDED-PACKAGES* :TEST #'EQUAL)
                                  (CONTEXT-MISS-SESSIONS))
```

```
                              :TEST #'EQUAL)
                 (POST-MSG '(DECISION 1 1)))
                (T
                 (POST-MSG '(DECISION 1 0)))))))

(DEFKS UPDATE-USER-UNDERSTANDING
   :DESCRIPTION       "update the understanding level of the user on MSLTRAIN steps."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'UNDERSTAND
   :TRIGGER-CONDITION '(:BIND (?ACTION NODE ?NODE)
                        :TEST (OR (AND (EQ ?ACTION 'CONTINUE)
                                       (EQ (SEND ?NODE :SEND-IF-HANDLES :TYPE) :RESULT-DISPLAY))
                                  (SEND ?NODE :SEND-IF-HANDLES :CONCEPT)))
   :PRECONDITION      T
   :ACTION
   '(UPDATE-MSG (OR (SEND ?NODE :CONCEPT) (SEND ?NODE :NAME))
             (COND ((EQ (SEND ?NODE :TYPE) :RESULT-DISPLAY) 100)
                   ((< (- (INPUT-TIME (ACCESS-MSG 'INPUT NIL))
                          (OR (INPUT-TIME
                                 (ACCESS-MSG 'INPUT
                                   #'(LAMBDA (MSG)
                                       (MEMBER (INPUT-KEY MSG) '(SPACE GO-BACK QUICK-BACK)))
                                   :START 1))
                              *START-TIME*)) 100) 100)
                   (T 40))
             :OPERATION #'+))

(DEFKS INFER-WANT-INFO-GOAL
   :DESCRIPTION       "infer the user's goal when the user started to access information."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'GOAL
   :TRIGGER-CONDITION '(WANT INFO ?REFERENCE)
   :PRECONDITION      '(ACCESS-MSG 'GOAL '(DECISION ?KIND ?PURPOSE))
   :ACTION
   '(LET ((GOAL-FLAGS (BUILD-FLAGS-FROM-GOAL (LIST ?KIND ?PURPOSE))))
      (WHEN (>= (GOAL-STEP) 5)
        (IF (>= (GOAL-PURPOSE) 1)
            (POST-MSG '(ACCESS INFO RELATED COMPARE))
            (POST-MSG '(ACCESS INFO RELATED REFER))))))

(DEFKS INFER-ACCESS-GOAL
   :DESCRIPTION       "infer the user's goal when the user requested an information item."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'SEMANTICS
   :TO-LEVEL          'GOAL
   :TRIGGER-CONDITION '(:BIND (?ACTION ?ITEM ?KEY)
                       :TEST (OR (EQ ?ACTION 'REQUEST) (SUBSTRING? 'ACCESS ?ITEM)))
   :PRECONDITION      '(ACCESS-MSG 'GOAL '(DECISION ?KIND ?PURPOSE))
   :ACTION
   '(PROGWB (ACCESS-MSG 'GOAL '(ACCESS ?X ?Y ?Z))
      (COND ((EQ ?ACTION 'STOP)
             (POST-MSG '(ACCESS NIL NIL NIL)))
            ((EQ ?ACTION 'QUIT)
             (POST-MSG '(ACCESS QUIT ,?Y ,?Z)))
            ((EQ ?ACTION 'SET)
             (WHEN (EQ ?ITEM 'ACCESS-PKG)
               (SETF (CONTEXT-PKG) ?KEY)))
            ((EQ ?ACTION 'REQUEST)
             (POST-MSG '(ACCESS REQUEST ,?ITEM ,?KEY))))))

(DEFKS UPDATE-FORM-PREFERENCE
   :DESCRIPTION       "update the user's preference on information portrayal format."
```

```
    :PROBLEM-DOMAIN     'UNDERSTAND
    :BB                 'BB$MAIN
    :FROM-LEVEL         'GOAL
    :TO-LEVEL           'USER-CHAR
    :TRIGGER-CONDITION '(ACCESS REQUEST ?ITEM ?KEY)
    :PRECONDITION       '(ACCESS-MSG 'GOAL '(DECISION ?KIND ?PURPOSE))
    :ACTION
    '(LET* ((C-PREF (COPY-ALL (CADR (ACCESS-MSG 'USER-CHAR 'FORM-PREFERENCE))))
            (ITEM   (SEND ?ITEM :IDENTIFIER))
            (D-ITEM (AND *DISPLAYED-INFO-ITEM* (SEND (CAR *DISPLAYED-INFO-ITEM*) :IDENTIFIER)))
            (PREF   (DISPLAY-FORM-TYPE ITEM))
            DEG)
       (WHEN (COND ((AND D-ITEM
                         (NOT (EQ (CAR *DISPLAYED-INFO-ITEM*) ?ITEM))
                         (EQ (SEND (CAR *DISPLAYED-INFO-ITEM*) :PARENT) (SEND ?ITEM :PARENT))
                         (NOT (EQ (DISPLAY-FORM-TYPE D-ITEM) PREF)))
                    (SETQ DEG 3))
                   ((AND (SEND ?ITEM :PARENT)
                         (>= (LENGTH (SEND (SYMBOL-VALUE (SEND ?ITEM :PARENT)) :FORMS)) 2))
                    (SETQ DEG 1)))
         (SETF (CONTEXT-FORM) PREF)
         (IF C-PREF
             (LET ((PREFS (ASSOC PREF C-PREF)))
               (IF PREFS
                   (INCREMENT (CDR PREFS) DEG)
                   (PUSH (CONS PREF DEG) C-PREF))
               (UPDATE-MSG 'FORM-PREFERENCE C-PREF))
             (UPDATE-MSG 'FORM-PREFERENCE (LIST (CONS PREF DEG)))))))

(DEFKS UPDATE-VIEW-PREFERENCE
    :DESCRIPTION        "update the user's preference on information scope."
    :PROBLEM-DOMAIN     'UNDERSTAND
    :BB                 'BB$MAIN
    :FROM-LEVEL         'GOAL
    :TO-LEVEL           'USER-CHAR
    :TRIGGER-CONDITION '(ACCESS REQUEST ?ITEM ?KEY)
    :PRECONDITION       '(ACCESS-MSG 'GOAL '(DECISION ?KIND ?PURPOSE))
    :ACTION
    '(LET* ((C-PREF (CADR (ACCESS-MSG 'USER-CHAR 'VIEW-PREFERENCE)))
            (ITEM   (SEND ?ITEM :IDENTIFIER))
            (D-ITEM (AND *DISPLAYED-INFO-ITEM* (SEND (CAR *DISPLAYED-INFO-ITEM*) :IDENTIFIER)))
            (PREF   (VIEW-POINT-TYPE ITEM)))
       (WHEN (AND D-ITEM
                  (NOT (EQ (CAR *DISPLAYED-INFO-ITEM*) ?ITEM))
                  (EQ (SEND (CAR *DISPLAYED-INFO-ITEM*) :TOP-INFO) (SEND ?ITEM :TOP-INFO))
                  (NOT (EQ (VIEW-POINT-TYPE D-ITEM) PREF)))
         (SETF (CONTEXT-VIEW) PREF)
         (IF C-PREF
             (LET ((PREFS (ASSOC PREF C-PREF)))
               (IF PREFS
                   (INCREMENT (CDR PREFS))
                   (PUSH (CONS PREF 1) C-PREF))
               (UPDATE-MSG 'VIEW-PREFERENCE C-PREF))
             (POST-MSG '(VIEW-PREFERENCE (,(CONS PREF 1)))))))


(DEFKS SUPPORT-DECISION-MAKING
    :DESCRIPTION        "infer the user's needs when their goal is to make decisions."
    :PROBLEM-DOMAIN     'UNDERSTAND
    :BB                 'BB$MAIN
    :FROM-LEVEL         'GOAL
    :TO-LEVEL           'NEED
    :TRIGGER-CONDITION '(:BIND (DECISION ?KIND ?PURPOSE)
                         :TEST (AND (TYPEP ?KIND '(INTEGER 0 3)) (TYPEP ?PURPOSE '(INTEGER 0 3))))
    :PRECONDITION       T
    :ACTION
    '(PROGN
```

```
      (POST-MSG `(INFO SET (:ITEM
                            (:VIEW-POINT
                              ,(COND ((= ?KIND 1)
                                      (IF (EQ (CONTEXT-VIEW) :WHOLE) :WHOLE :BY-PKG))
                                     ((AND (= ?KIND 2) (OR (= ?PURPOSE 1) (= ?PURPOSE 0))) :WHOLE)
                                     ((AND (PLUSP ?KIND) (= ?PURPOSE 2)) :BY-PERSON)
                                     ((= ?KIND 3)
                                      (IF (EQ (CONTEXT-VIEW) :WHOLE) :WHOLE NIL))
                                     (T NIL)))
                            :DISPLAY-FORM (:POP-UP 0))))
      (WHEN (AND (= ?KIND 0) (= ?PURPOSE 1))
        (POST-MSG `(INFO EXTRA (:ITEM (:VIEW-POINT :WHOLE) :DISPLAY-FORM (:POP-UP 1)))))
      (WHEN (AND (= ?KIND 1) (NOT (EQ (CAR (ACCESS-MSG 'SEMANTICS)) 'CURSOR)))
        (POST-MSG `(CURSOR PLACE (:ROW)))))

(DEFKS SUPPORT-USER-EVOLUTION
   :DESCRIPTION       "infer the user's needs when the user seems to understand
                            certain step in MSLTRAIN."
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'UNDERSTAND
   :TO-LEVEL          'NEED
   :TRIGGER-CONDITION '(:BIND (?CONCEPT ?DEGREE)
                       :TEST (AND (NUMBERP ?DEGREE) (>= ?DEGREE 100)))
   :PRECONDITION      T
   :ACTION            '(POST-MSG `(CONCEPT DONT-SHOW ,?CONCEPT)))

(DEFKS SUPPORT-MENU-ACCESS
   :DESCRIPTION       "infer the user's need when their goal is to access information
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'GOAL
   :TO-LEVEL          'NEED
   :TRIGGER-CONDITION '(ACCESS INFO ?SCOPE ?PURPOSE)
   :PRECONDITION      T
   :ACTION
   '(COND ((EQ ?SCOPE 'RELATED)
           (POST-MSG '(INFO-MENU SET (:ITEM :VIEW-POINT :DISPLAY-FORM :LOOP-P))))
          ((EQ ?SCOPE 'CONSQ)
           (POST-MSG '(INFO-MENU SET (:ITEM :VIEW-POINT :DISPLAY-FORM :LOOP-P))))))

(DEFKS SUPPORT-ITEM-ACCESS
   :DESCRIPTION       "infer the user's need when their goal is to access information"
   :PROBLEM-DOMAIN    'UNDERSTAND
   :BB                'BB$MAIN
   :FROM-LEVEL        'GOAL
   :TO-LEVEL          'NEED
   :TRIGGER-CONDITION '(ACCESS REQUEST ?ITEM ?KEY)
   :PRECONDITION      T
   :ACTION
   '(LET ((POP-UP (AND (>= ?KEY 1) (IF (>= ?KEY 3)  0 1)))
          (FORM   (AND (>= ?KEY 1) (IF (EVENP ?KEY) 1 0))))
      (OR (INSTANCEP ?ITEM) (SETQ ?ITEM (SYMBOL-VALUE ?ITEM)))
      (POST-MSG `(INFO-ITEM SET (:ITEM ,?ITEM) (:DISPLAY-FORM ,FORM) (:POP-UP ,POP-UP)
                                :VIEW-POINT)))))

(DEFKS DESIGN-NETWORK-FLOW
   :DESCRIPTION       "decide which nodes in the full guide mode should be removed or added."
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'NEED
   :TO-LEVEL          'INTERPRET
   :TRIGGER-CONDITION '(CONCEPT ?NEED ?CONCEPT)
   :ACTION
   '(LET ((ACTION (COND ((EQ ?NEED 'SHOW)      :RECOVER)
                        ((EQ ?NEED 'DONT-SHOW) :REMOVE))))
      (OR (INSTANCEP ?CONCEPT) (SETQ ?CONCEPT (SYMBOL-VALUE ?CONCEPT)))
```

```
      (AND ACTION
           (POST-MSG `(NETWORK-DESIGN (,ACTION ,(OR (SEND ?CONCEPT :SEND-IF-HANDLES :CORR-NODE)
                                                    (SEND ?CONCEPT :NAME))))))))

(DEFKS DESIGN-INFORMATION
   :DESCRIPTION       "infer which elements in the information design should be determined."
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'NEED
   :TO-LEVEL          'INTERPRET
   :TRIGGER-CONDITION '(:BIND (?NAME ?X ?VARS)
                        :TEST (AND (MEMBER ?NAME '(INFO INFO-ITEM INFO-MENU))
                                   (MEMBER ?X '(SET EXTRA))))
   :PRECONDITION      T
   :ACTION
   '(POST-MSG (LIST (CONCAT ?NAME '-DESIGN)
                    (EVAL `(,(IF (EQUAL ?NAME 'INFO-MENU) 'MAKE-INFO-MENU-DESIGN 'MAKE-INFO-DESIGN)
                            ,@(LET (OUT)
                                (DOLIST (X ?VARS)
                                  (SETQ OUT (APPEND OUT (IF (CONSP X) X (LIST X NIL)))))
                                OUT))))))

(DEFKS DESIGN-CURSOR-POSITION
   :DESCRIPTION       "decide where to put the cursor."
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'NEED
   :TO-LEVEL          'INTERPRET
   :TRIGGER-CONDITION '(CURSOR PLACE (:ROW))
   :PRECONDITION      '(CONTEXT-LEVEL)
   :ACTION
   '(UNLESS (OR (AND (EQUAL (CONTEXT-PKG) (CONTEXT-IN-PKG))
                     (AND (= (CONTEXT-LEVEL) (1+ (CONTEXT-IN-LEVEL)))))
                (AND (CONTEXT-IN-PKG) (NEXT? (CONTEXT-IN-PKG) (CONTEXT-PKG) (CONTEXT-PKGS))))
      (LET ((LINE 0))
        (BLOCK XXX
          (DO ((PKG (CONTEXT-PKGS) (CDR PKG))
               (SES (CONTEXT-SESSIONS) (CDR SES)))
              ((ENDP PKG))
            (IF (EQUAL (CONTEXT-PKG) (CAR PKG))
                (DOLIST (X (CAR SES))
                  (IF (= (CONTEXT-LEVEL) (CAR X))
                      (RETURN-FROM XXX NIL)
                      (INCREMENT LINE)))
                (INCREMENT LINE (LENGTH (CAR SES))))))
        (POST-MSG `(CURSOR-POSITION ,LINE)))))

(DEFKS REASON-INFO-ITEM
   :DESCRIPTION       "reason the information item the user may need."
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'INTERPRET
   :TO-LEVEL          'INTERPRET
   :TRIGGER-CONDITION '(:BIND (?NAME ?DESIGN)
                        :TEST (AND (MEMBER ?NAME '(INFO-MENU-DESIGN INFO-DESIGN))
                                   (NEED-TO-BE-SET? :ITEM ?DESIGN ?NAME)))
   :PRECONDITION      T
   :ACTION
   '(LET ((GOAL-FLAGS (BUILD-FLAGS-FROM-GOAL (CDR (ACCESS-MSG 'GOAL 'DECISION))))
          ITEM WORK-ITEM)
      (WHEN (SETQ ITEM (CDR (ASSOC GOAL-FLAGS *GOAL-INFO-TABLE*)))
        (SETQ WORK-ITEM (CONTEXT-PERSON))
        (COND ((EQ ?NAME 'INFO-DESIGN)
               (AND (SETQ ITEM (GET-MAX-PROB-ITEM ITEM WORK-ITEM))
                    (OR (INSTANCEP ITEM) (INSTANCEP (SETQ ITEM (SYMBOL-VALUE ITEM))))
                    (FILL-MSG 'INFO-DESIGN :ITEM ITEM)))
              ((EQ ?NAME 'INFO-MENU-DESIGN)
```

```
            (WHEN (SETQ ITEM (GET-MAX-PROB-ITEM ITEM WORK-ITEM (CAR *DISPLAYED-INFO-ITEM*)))
              (FILL-MSG 'INFO-MENU-DESIGN :ITEM ITEM)))))))

(DEFKS DECIDE-DISPLAY-FORM
   :DESCRIPTION       "reason the information portrayal format the user may prefer."
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'INTERPRET
   :TO-LEVEL          'INTERPRET
   :TRIGGER-CONDITION '(:BIND (?NAME ?DESIGN)
                        :TEST (AND (MEMBER ?NAME '(INFO-ITEM-DESIGN INFO-DESIGN INFO-MENU-DESIGN))
                                   (NEED-TO-BE-SET? :DISPLAY-FORM ?DESIGN ?NAME)))
   :PRECONDITION      '(:BIND (ACCESS-MSG 'INTERPRET `(,?NAME ?C-DESIGN))
                        :TEST (FIRST ?C-DESIGN))
   :ACTION
   '(LET (FORMS (?ITEM (CAR ?C-DESIGN)))
      (OR (INSTANCEP ?ITEM) (SETQ ?ITEM (SYMBOL-VALUE ?ITEM)))
      (SETQ FORMS (SEND ?ITEM :SEND-IF-HANDLES :FORMS))
      (FILL-MSG ?NAME :DISPLAY-FORM
                (COND ((NULL FORMS) :N/A)
                      ((NULL (CDR FORMS)) (CAR FORMS))
                      (T (CASE (CONTEXT-FORM)
                           (:TABLE 0) (:GRAPH 1) (OTHERWISE 0))))
                (IF (EQ ?NAME 'INFO-ITEM-DESIGN) 'INFO-DESIGN ?NAME))))

(DEFKS DECIDE-POP-UP
   :DESCRIPTION       "reason where to display information."
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'INTERPRET
   :TO-LEVEL          'INTERPRET
   :TRIGGER-CONDITION '(:BIND (INFO-ITEM-DESIGN ?DESIGN)
                        :TEST (NEED-TO-BE-SET? :POP-UP ?DESIGN))
   :PRECONDITION      '(:BIND (ACCESS-MSG 'INTERPRET '(INFO-ITEM-DESIGN (?ITEM IGNORE IGNORE IGNORE)))
                        :TEST ?ITEM)
   :ACTION            '(FILL-MSG 'INFO-ITEM-DESIGN :POP-UP 0 'INFO-DESIGN))

(DEFKS DECIDE-INFO-MENU-LOOP-P
   :DESCRIPTION       "Decide whether "
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'INTERPRET
   :TO-LEVEL          'INTERPRET
   :TRIGGER-CONDITION '(:BIND (INFO-MENU-DESIGN ?DESIGN)
                        :TEST (NULL (INFO-MENU-DESIGN-LOOP-P ?DESIGN)))
   :PRECONDITION      '(:BIND (ACCESS-MSG 'INTERPRET '(INFO-MENU-DESIGN ?C-DESIGN))
                        :TEST (INFO-MENU-DESIGN-ITEM ?C-DESIGN))
   :ACTION            '(FILL-MSG 'INFO-MENU-DESIGN :LOOP-P T))

(DEFKS DECIDE-VIEW-POINT
   :DESCRIPTION       "reason the information scope the user may prefer."
   :PROBLEM-DOMAIN    'INTERPRET
   :BB                'BB$MAIN
   :FROM-LEVEL        'INTERPRET
   :TO-LEVEL          'INTERPRET
   :TRIGGER-CONDITION '(:BIND (?NAME ?DESIGN)
                        :TEST (AND (MEMBER ?NAME '(INFO-DESIGN INFO-ITEM-DESIGN INFO-MENU-DESIGN))
                                   (NEED-TO-BE-SET? :VIEW-POINT ?DESIGN)))
   :PRECONDITION      '(:BIND (ACCESS-MSG 'INTERPRET `(,?NAME (?ITEM IGNORE IGNORE IGNORE)))
                        :TEST ?ITEM)
   :ACTION
   '(LET (VIEW-POINTS
          (GOAL-FLAGS (BUILD-FLAGS-FROM-GOAL (CDR (ACCESS-MSG 'GOAL 'DECISION)))))
      (OR (INSTANCEP ?ITEM) (SETQ ?ITEM (SYMBOL-VALUE ?ITEM)))
      (SETQ VIEW-POINTS (SEND ?ITEM :SEND-IF-HANDLES :VIEW-POINTS))
      (FILL-MSG ?NAME :VIEW-POINT
                (COND ((NULL VIEW-POINTS) :N/A)
```

```
                    ((OR (NULL (CDR VIEW-POINTS)) (NULL *CURRENT-STEP*) (< *CURRENT-STEP* 3))
                     (CAR VIEW-POINTS))
                    ((EQ (CONTEXT-VIEW) :WHOLE) :WHOLE)
                    ((EQ (CONTEXT-VIEW) :PART)
                     (COND ((AND (CONTEXT-PERSON) (MEMBER :BY-PERSON VIEW-POINTS :TEST #'EQUAL))
                            :BY-PERSON)
                           ((AND (CONTEXT-PKG) (MEMBER :BY-PKG VIEW-POINTS :TEST #'EQUAL))
                            :BY-PKG)
                           (T (CAR VIEW-POINTS))))
                    (T (CAR VIEW-POINTS)))
                'INFO-DESIGN)))

(DEFKS INFO-DISPLAY
   :DESCRIPTION       "display information designed."
   :PROBLEM-DOMAIN    'IMPLEMENT
   :BB                'BB$MAIN
   :FROM-LEVEL        'INTERPRET
   :TO-LEVEL          NIL
   :TRIGGER-CONDITION '(INFO-DESIGN IGNORE)
   :PRECONDITION      '(:BIND (ACCESS-MSG 'INTERPRET
                                          '(INFO-DESIGN (?INFO-ITEM ?VIEW-POINT ?FORM IGNORE)))
                       :TEST (AND ?INFO-ITEM ?VIEW-POINT ?FORM))
   :ACTION
   '(LET* ((INFO-NAME (SEND ?INFO-ITEM :IDENTIFIER))
           (ITEM (COND ((OR (EQ ?VIEW-POINT :BY-PKG) (SUBSTRING? 'PKG INFO-NAME))
                        (IF (INSTANCEP (CONTEXT-PKG)) (CONTEXT-PKG) (GET-OBJECT (CONTEXT-PKG))))
                       ((OR (EQ ?VIEW-POINT :BY-PERSON) (SUBSTRING? 'PERSON INFO-NAME))
                        (CONTEXT-PERSON)))))
      (WHEN ?INFO-ITEM
        (UNLESS (AND (INSTANCEP (CAR *DISPLAYED-INFO-ITEM*))
                     (OR (EQUAL (CAR *DISPLAYED-INFO-ITEM*) ?INFO-ITEM)
                         (EQ (SEND (CAR *DISPLAYED-INFO-ITEM*) :IDENTIFIER)
                             (CONCAT INFO-NAME
                                     (AND (NOT (EQ ?VIEW-POINT :N/A)) (CONCAT '- ?VIEW-POINT))
                                     (AND (NOT (EQ ?FORM :N/A))
                                          (CONCAT '- (CASE ?FORM (0 'TABLE) (1 'CHART)))))))
                     (OR (NULL (CADR *DISPLAYED-INFO-ITEM*))
                         (EQUAL ITEM (CADR *DISPLAYED-INFO-ITEM*)))))
          (IF (AND (MEMBER (CAR (ACCESS-MSG 'SEMANTICS)) '(SET CHANGE)) *DISPLAYED-INFO-ITEM*)
              (PROGN (SET-PARAMETER *CURRENT-INFO-ITEM*   ?INFO-ITEM)
                     (SET-PARAMETER *DEFAULT-ACCESS-ITEM* ITEM)
                     (SET-PARAMETER *LOCAL-DISPLAY-FORM*  ?FORM)
                     (SET-PARAMETER *LOCAL-VIEW-POINT*    ?VIEW-POINT))
              (DISPLAY-INFO-ITEM ?INFO-ITEM :POPUP 0 :VIEW-POINT ?VIEW-POINT
                                 :FORM ?FORM :ACCESS-VAR ITEM))))))

(DEFKS PLACE-CVV-CURSOR
   :DESCRIPTION       "order the executor, CVV-CURSOR-PLACER, to place the cursor
                           at the designed location.
   :PROBLEM-DOMAIN    'IMPLEMENT
   :BB                'BB$MAIN
   :FROM-LEVEL        'INTERPRET
   :TO-LEVEL          NIL
   :TRIGGER-CONDITION '(CURSOR-POSITION ?LINE)
   :PRECONDITION      T
   :ACTION
   '(WHEN (AND ?LINE (NUMBERP ?LINE))
      (IF (EQ (SEND MSL-CVV-WINDOW-5 :STATUS) :SELECTED)
          (PROGN (SEND MSL-CVV-WINDOW-5 :MOVE-CURSOR NIL)
                 (DOTIMES (N ?LINE)
                   (SEND MSL-CVV-WINDOW-5 :MOVE-CURSOR 'DOWN)))
          (SEND MSL-CVV-WINDOW-5 :SET-CURSOR-LINE ?LINE))))

(DEFKS ARRANGE-INFO-DISPLAY
   :DESCRIPTION       "set the values of information display parameters."
   :PROBLEM-DOMAIN    'IMPLEMENT
   :BB                'BB$MAIN
```

```
    :FROM-LEVEL         'INTERPRET
    :TO-LEVEL           NIL
    :TRIGGER-CONDITION '(:BIND (?NAME IGNORE)
                         :TEST (MEMBER ?NAME '(INFO-MENU-DESIGN INFO-ITEM-DESIGN)))
    :PRECONDITION       '(:BIND (ACCESS-MSG 'INTERPRET `(,?NAME ?DESIGN))
                         :TEST (ALL-ELEMENT-NON-NIL? ?DESIGN))
    :ACTION
    '(COND ((EQ ?NAME 'INFO-ITEM-DESIGN)
            (LET ((PKG (IF (INSTANCEP (CONTEXT-PKG)) (CONTEXT-PKG) (GET-OBJECT (CONTEXT-PKG)))))
              (OR *LOCAL-POPUP*
                  (EQ (INFO-DESIGN-POP-UP ?DESIGN) :N/A)
                  (SET-PARAMETER *LOCAL-POPUP* (INFO-DESIGN-POP-UP ?DESIGN)))
              (OR *LOCAL-DISPLAY-FORM*
                  (EQ (INFO-DESIGN-DISPLAY-FORM ?DESIGN) :N/A)
                  (SET-PARAMETER *LOCAL-DISPLAY-FORM* (INFO-DESIGN-DISPLAY-FORM ?DESIGN)))
              (OR *LOCAL-VIEW-POINT* (EQ (INFO-DESIGN-VIEW-POINT ?DESIGN) :N/A)
                  (SET-PARAMETER *LOCAL-VIEW-POINT* (INFO-DESIGN-VIEW-POINT ?DESIGN)))
              (LET* ((ITEM (INFO-DESIGN-ITEM ?DESIGN))
                     (ITEM-NAME (SEND ITEM :IDENTIFIER)))
                (UNLESS (AND (INSTANCEP (CAR *DISPLAYED-INFO-ITEM*))
                             (CADR *DISPLAYED-INFO-ITEM*)
                             (EQ (CAR *DISPLAYED-INFO-ITEM*) ITEM)
                             (EQUAL (CADR *DISPLAYED-INFO-ITEM*)
                                    (COND ((SUBSTRING? 'BY-PKG ITEM-NAME) PKG)
                                          ((SUBSTRING? 'BY-PERSON ITEM-NAME) (CONTEXT-PERSON)))))
                  (CASE (SEND ITEM :TYPE)
                    (:TOP (CASE *LOCAL-VIEW-POINT*
                            (:BY-PERSON (SET-PARAMETER *ACCESS-PERSON* (CONTEXT-PERSON)))
                            (:BY-PKG    (SET-PARAMETER *ACCESS-PACKAGE* PKG))))
                    (OTHERWISE
                     (COND ((SUBSTRING? 'BY-PKG ITEM-NAME)
                            (SET-PARAMETER *ACCESS-PACKAGE* PKG))
                           ((SUBSTRING? 'BY-PERSON ITEM-NAME)
                            (SET-PARAMETER *ACCESS-PERSON* (CONTEXT-PERSON)))))))))))
          ((EQ ?NAME 'INFO-MENU-DESIGN)
           (SET-PARAMETER *INFO-MENU-LOOP-P*
                          (IF (EQ (INFO-MENU-DESIGN-LOOP-P ?DESIGN) :NO) NIL T))
           (OR (EQ (INFO-DESIGN-ITEM ?DESIGN) :ALL)
               (LET* ((ITEM (INFO-MENU-DESIGN-ITEM ?DESIGN))
                      (TYPE (SEND (SYMBOL-VALUE ITEM) :TYPE)))
                 (SET-PARAMETER *DEFAULT-INFO-ITEM*
                     (CONCAT ITEM
                             (WHEN (EQ TYPE :TOP)
                               (CONCAT '- (OR (INFO-MENU-DESIGN-VIEW-POINT ?DESIGN) 'WHOLE)))
                             (WHEN (MEMBER TYPE '(:TOP :MIXED))
                               (CASE (INFO-MENU-DESIGN-DISPLAY-FORM ?DESIGN)
                                 (0 '-TABLE) (1 '-CHART)
                                 (OTHERWISE '-TABLE)))))))))

(DEFKS ARRANGE-NETWORK
   :DESCRIPTION       "remove or add nodes in the full guide interaction mode.
   :PROBLEM-DOMAIN    'IMPLEMENT
   :BB                'BB$MAIN
   :FROM-LEVEL        'INTERPRET
   :TO-LEVEL          NIL
   :TRIGGER-CONDITION '(NETWORK-DESIGN (?ACTION ?NODE))
   :ACTION
   '(PROGN (OR (INSTANCEP ?NODE) (SETQ ?NODE (SYMBOL-VALUE ?NODE)))
           (SEND ?NODE :SEND-IF-HANDLES ?ACTION)))

(DEFKS SAVE-SESSION
   :DESCRIPTION       "save information about the session and the user's characteristics
                           when the user ends the session."
   :PROBLEM-DOMAIN    'SAVE
   :FROM-LEVEL        'INPUT
   :TRIGGER-CONDITION '(:END-SESSION)
   :PRECONDITION      '(NULL *SAVED-SESSION-P*)
```

```
:ACTION
'(WHEN (SEND CONFIRM-SAVE :ASK NIL *MSL*)
   (SAVE-THIS-SESSION) (SETQ *SAVED-SESSION-P* T)))
```

Appendix F. Computer Traces of the Responsiveness Layer Process

The following computer traces were taken from a MSLTRAIN session. The traces show what the responsive layer process did internally. The purpose of providing the traces is to demonstrate MSLTRAIN is a responsive system in terms of the responsiveness process. The traces include:

- Name of the knowledge sources triggered and executed during the session, which appears right after `">> KS:"`.
- Name of the level in the blackboard whose change caused the knowledge source to be triggered.
- Actions the knowledge source took:
    - POST-MSG — post a message on a level in the blackboard,
    - UPDATE-MSG — update a message already posted on a level in the blackboard,
    - FILL-MSG — fill in an attribute value to a blank message on a level in the blackboard,
    - SET-PARAMETER — set the value of a parameter.

- Name of the level in the blackboard where the knowledge source posted a message.
- Content of the message, which appears right after "MSG:".

To make it easier to understand the traces, I added comments to the traces such as:

- Description of the user's action, which starts with "***";
- Description of the system's response, which starts with "ZZZ"; and
- Explanation of the internal responsiveness process, which starts with "###".

```
*** The user starts the session.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (NEW-SESSION)

>> KS: INITIALIZE-NEW-SESSION
    ### The system initializes the values of system variables and parameters.

>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
    ### Accept a report and post a message saying the user started the session.
        POST-MSG on INPUT Level, MSG: (START-SESSION)

### START OF THE INITIAL TUNING PROCESS

>> KS: RECOGNIZE-USER
        Triggered by the Event in INPUT Level
    %%% Ask the name of the user (Figure 26 on page 262).
    *** The user types "Kwang" and hit the "RETURN" key.
    ### Recognize the user used the system before and retrieve the previous
    ###  interaction history and the characteristics of the user.
        POST-MSG on HISTORY Level, MSG: (T-NO-OF-USE 1)
        POST-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 50)
        POST-MSG on HISTORY Level, MSG: (T-NO-OF-ERRORS 0)
        POST-MSG on HISTORY Level, MSG: (T-NO-OF-MOUSE-USE 0)
        POST-MSG on USER-CHAR Level, MSG: (PREV-PROFICIENCY NOVICE)
        POST-MSG on USER-CHAR Level, MSG: (DEVICE-PREFERENCE KEY)
        POST-MSG on HISTORY Level, MSG: (DEC-PATTERN-HIS NIL)
        POST-MSG on USER-CHAR Level, MSG: (DECISION-PATTERN PKG-DOWN)
        POST-MSG on USER-CHAR Level, MSG: (FORM-PREFERENCE NIL)
        POST-MSG on USER-CHAR Level, MSG: (VIEW-PREFERENCE NIL)
        POST-MSG on HISTORY Level,
         MSG: (PREVIOUS-SESSION (Kwang ABC 2865533033 2865533294 50 0 0 0 NOVICE
                                      KEY NIL PKG-DOWN NIL NIL T ....

>> KS: COMPUTE-INT-CHRS
        Triggered by the Event in HISTORY Level
    ### Compute how often the user used the mouse before.
        UPDATE-MSG on HISTORY Level, MSG: (T-MOUSE-USE-RATIO 0 NIL)

>> KS: COMPUTE-INT-CHRS
        Triggered by the Event in HISTORY Level
    ### Compute how often the user made errors before.
        UPDATE-MSG on HISTORY Level, MSG: (T-ERROR-RATIO 0 NIL)

>> KS: INFER-USER-PROFICIENCY
        Triggered by the Event in HISTORY Level
    ### Infer the proficiency level of the user on MSLTRAIN.
        POST-MSG on USER-CHAR Level, MSG: (PROFICIENCY REGULAR)

>> KS: INFER-DEVICE-PREFERENCE
        Triggered by the Event in HISTORY Level
    ### Infer which input device the user prefers (keyboard or mouse).
        UPDATE-MSG on USER-CHAR Level, MSG: (DEVICE-PREFERENCE KEY NIL)

>> KS: INFER-SESSION-GOAL
        Triggered by the Event in HISTORY Level
    ### Check whether the user wants a new session or to continue the previous session.
        POST-MSG on GOAL Level, MSG: (SESSION (NEW))

>> KS: SUPPORT-PROFICIENCY-CHANGE
        Triggered by the Event in USER-CHAR Level
    ### Infer which system parameters should be reset to support the
    ###   change in the user's proficiency level.
        POST-MSG on NEED Level, MSG: (SET PARAMETER ((CURRENT-MODE REGULAR) (ERROR-MESSAGE-P REGULAR)))

>> KS: SUPPORT-DEVICE-PREFERENCE
```

```
        Triggered by the Event in USER-CHAR Level
    ### Infer which system parameters should be reset to support the
    ###   input device preference of the user.
        POST-MSG on NEED Level, MSG: (SET PARAMETER ((SHOW-SUGG-MENU-P KEY) (MENU-POSITION KEY)))

>> KS: DECIDE-SHOW-SUGG-MENU-P
        Triggered by the Event in NEED Level
    ### Decide whether to show the permanent information menu.
        POST-MSG on DESIGN Level, MSG: (PARAMETER (SHOW-SUGG-MENU-P NIL))

>> KS: DECIDE-MENU-POSITION
        Triggered by the Event in NEED Level
    ### Decide appropriate position (top or bottom) of the guide window.
        POST-MSG on DESIGN Level, MSG: (PARAMETER (MENU-POSITION TOP))

>> KS: DECIDE-INTERACTION-MODE
        Triggered by the Event in NEED Level
    ### Decide the appropriate interaction mode.
        POST-MSG on DESIGN Level, MSG: (PARAMETER (CURRENT-MODE REGULAR-MODE))

>> KS: DECIDE-ERROR-MESSAGE-P
        Triggered by the Event in NEED Level
    ### Decide whether to show error messages or to just beep when the user makes an error.
        POST-MSG on DESIGN Level, MSG: (PARAMETER (ERROR-MESSAGE-P T))

 ### Set the values of the system parameters.
>> KS: ARRANGE-SYSTEM-PARAMETERS

>> KS: ARRANGE-SYSTEM-PARAMETERS

>> KS: ARRANGE-SYSTEM-PARAMETERS

>> KS: ARRANGE-SYSTEM-PARAMETERS

### THE END OF THE INITIAL TUNING PROCESS

%%% The system shows a screen like Figure 27 on page 263.
*** The user enters "5".
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
    ### Recognize the user entered "5".
        POST-MSG on INPUT Level, MSG: (COMMAND 13957 5 KEY TOP STEP-5-EXP-NODE)

>> KS: INFER-TOP-SEMANTICS
        Triggered by the Event in INPUT Level
    ### Recognize the user want to start Step 5.
        POST-MSG on SEMANTICS Level, MSG: (START STEP 5)
        UPDATE-MSG on HISTORY Level, MSG: (STEPS 5 CONS)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
    ### Update the interaction history.
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-STEP-CONTEXT
        Triggered by the Event in SEMANTICS Level
    ### Update the values of the context variables.
        POST-MSG on CONTEXT Level,
         MSG: (5 NIL NIL NIL NIL NIL NIL NIL NIL NIL (0 0 0) (NIL NIL NIL) 1 NIL 1 NIL PART TABLE)

>> KS: INFER-STEP-GOAL
        Triggered by the Event in SEMANTICS Level
    ### Infer the user's goal is to set training schedules.
        POST-MSG on GOAL Level, MSG: (DECISION 0 0)

>> KS: SUPPORT-DECISION-MAKING
```

```
        Triggered by the Event in GOAL Level
    ### Infer that the user needs some information.
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT NIL) DISPLAY-FORM (POP-UP 0)))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
    ### Start to design an information.
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL NIL NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
    ### Reason the information item the user needs most.
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM DESIRED-PROF-WHOLE-TABLE)

>> KS: DECIDE-VIEW-POINT
        Triggered by the Event in DESIGN Level
    ### Reason which viewpoint (whole or part) the user wants.
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN VIEW-POINT N/A)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
    ### Reason which portrayal format (table or graph) the user wants.
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM N/A)

>> KS: INFO-DISPLAY
    %%% Display the "Desired Individual Proficiencies" table.

%%% Show the employee menu showing the names of eight people.
%%% The screen looks like Figure 28 on page 264.
*** The user selects "Bill Muraham" in the menu.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 14410 SELECT KEY MENU 5 Bill Muraham)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (START DECISION Bill Muraham)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bill Muraham NIL NIL NIL NIL (MS Word Designer)
               (((1 0) (2 0)) ((2 0) (3 0))) (0 0) (0 0) (0 0 0) (NIL NIL NIL) 1 NIL 1
               NIL PART TABLE)

>> KS: INFER-DEC-START-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 1 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0)))
        POST-MSG on NEED Level, MSG: (CURSOR PLACE (ROW))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PKG NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM COMP-RT-TS)
```

```
>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 0)

>> KS: DESIGN-CURSOR-POSITION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (CURSOR-POSITION 0)

>> KS: PLACE-CVV-CURSOR

>> KS: INFO-DISPLAY
    %%% Display the "Gap between RT and TS - By PKG" table of "MS Word."

%%% The system shows a decision window listing the required training courses for "Bill Muraham".
%%% The screen looks like Figure 29 on page 265.
*** The user selects "Mar" for the "MS Word" "Novice" session.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 15389 SET-VALUE KEY CVV 5 (MS Word..Novice 5))

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (SET DECISION (MS Word..Novice 5))

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bill Muraham MS Word MS Word 1 1 (MS Word Designer)
               (((1 2) (2 0)) ((2 0) (3 0))) (0 0) (1 0) (2 1 1)
               (((1 2)) NIL NIL) 1 NIL 1 NIL PART TABLE)

>> KS: INFER-DEC-SET-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 1 1)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0)))
        POST-MSG on NEED Level, MSG: (CURSOR PLACE (ROW))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PKG NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM COMP-RT-TS)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 0)

>> KS: DESIGN-CURSOR-POSITION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (CURSOR-POSITION 0)

>> KS: PLACE-CVV-CURSOR

>> KS: INFO-DISPLAY

*** The user selects "Nov" for the "MS Word" "Novice" session.
>> KS: ACCEPT-REPORT
```

```
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 15604 CHANGE-VALUE KEY CVV 5 (MS Word..Novice 1))

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (CHANGE DECISION (MS Word..Novice 1))

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
        MSG: (5 Bill Muraham MS Word MS Word 1 1 (MS Word Designer)
              (((1 1) (2 0)) ((2 0) (3 0))) (0 0) (1 0) (1 1 1) (NIL NIL NIL)
              1 NIL 1 NIL PART TABLE)

>> KS: INFER-DEC-CHANGE-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 1 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0)))
        POST-MSG on NEED Level, MSG: (CURSOR PLACE (ROW))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PKG NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM COMP-RT-TS)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 0)

>> KS: DESIGN-CURSOR-POSITION

>> KS: INFO-DISPLAY

*** The user selects "Jul" for the "MS Word" "Regular" session.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 15856 SET-VALUE KEY CVV 5 (MS Word..Regular 9))

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (SET DECISION (MS Word..Regular 9))

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bill Muraham MS Word MS Word 2 2 (MS Word Designer)
               (((1 1) (2 3)) ((2 0) (3 0))) (0 0) (2 0) (1 1 1) (NIL NIL NIL)
               1 NIL 1 NIL PART TABLE)

>> KS: INFER-DEC-SET-GOAL
        Triggered by the Event in SEMANTICS Level
```

```
        POST-MSG on GOAL Level, MSG: (DECISION 1 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0)))
        POST-MSG on NEED Level, MSG: (CURSOR PLACE (ROW))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PKG NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM COMP-RT-TS)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 0)

>> KS: DESIGN-CURSOR-POSITION

>> KS: INFO-DISPLAY
        Triggered by the Event in DESIGN Level
        SET-PARAMETER on NIL Level, MSG: (*CURRENT-INFO-ITEM* COMP-RT-TS)
        SET-PARAMETER on NIL Level, MSG: (*DEFAULT-ACCESS-ITEM* Designer)
        SET-PARAMETER on NIL Level, MSG: (*LOCAL-DISPLAY-FORM* 0)
        SET-PARAMETER on NIL Level, MSG: (*LOCAL-VIEW-POINT* BY-PKG)
    %%% Display "Gap between TC and TS; By PKG (Table)" of "Designer."

*** The user hits the "INFO" key.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 16074 INFO KEY CVV 5 Bill Muraham)

>> KS: INFER-ACCESS-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (WANT INFO NIL)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: INFER-WANT-INFO-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (ACCESS INFO RELATED REFER)

>> KS: SUPPORT-MENU-ACCESS
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO-MENU SET (ITEM VIEW-POINT DISPLAY-FORM LOOP-P))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-MENU-DESIGN (NIL NIL NIL NIL))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-MENU-DESIGN ITEM SCHEDULE-BY-PERSON)

>> KS: DECIDE-INFO-MENU-LOOP-P
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-MENU-DESIGN LOOP-P T)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-MENU-DESIGN DISPLAY-FORM 0)
```

```
>> KS: DECIDE-VIEW-POINT
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-MENU-DESIGN VIEW-POINT N/A)

>> KS: ARRANGE-INFO-DISPLAY
        Triggered by the Event in DESIGN Level
        SET-PARAMETER on NIL Level, MSG: (*INFO-MENU-LOOP-P* T)
        SET-PARAMETER on NIL Level, MSG: (*DEFAULT-INFO-ITEM* SCHEDULE-BY-PERSON-TABLE)

%%% The cursor moves to the "TS: Training Schedules; By Person" item in
%%%  the information menu pane (Figure 30 on page 266).
*** The user selects the "Gap between RT and TS; By PKG (Graph)" item.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 16320 (SELECT 0) KEY INFO MENU COMP-RT-TS-BY-PKG-CHART)

>> KS: INFER-ACCESS-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (REQUEST COMP-RT-TS-BY-PKG-CHART 0)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: INFER-ACCESS-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (ACCESS REQUEST COMP-RT-TS-BY-PKG-CHART 0)

>> KS: UPDATE-VIEW-PREFERENCE

>> KS: UPDATE-FORM-PREFERENCE
        Triggered by the Event in GOAL Level
        UPDATE-MSG on USER-CHAR Level, MSG: (FORM-PREFERENCE ((GRAPH . 3)) NIL)

>> KS: SUPPORT-ITEM-ACCESS
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level,
         MSG: (INFO-ITEM SET ((ITEM COMP-RT-TS-BY-PKG-CHART) (DISPLAY-FORM NIL) (POP-UP NIL) VIEW-POINT))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-ITEM-DESIGN (COMP-RT-TS-BY-PKG-CHART NIL NIL NIL))

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-ITEM-DESIGN DISPLAY-FORM N/A)

>> KS: DECIDE-POP-UP
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-ITEM-DESIGN POP-UP 0)

>> KS: DECIDE-VIEW-POINT
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-ITEM-DESIGN VIEW-POINT N/A)

>> KS: ARRANGE-INFO-DISPLAY
        Triggered by the Event in DESIGN Level
        SET-PARAMETER on NIL Level, MSG: (*LOCAL-POPUP* 0)
        SET-PARAMETER on NIL Level, MSG: (*ACCESS-PACKAGE* Designer)
%%% The system displays "Gap between TC and TS; By PKG (Graph)" of
%%%  "Designer" (Figure 31 on page 267).

*** The user hits the "END" key.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 16464 QUIT KEY INFO MENU NIL)
```

```
>> KS: INFER-ACCESS-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (STOP ACCESS NIL)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: INFER-ACCESS-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (ACCESS NIL NIL NIL)

%%% The cursor moves to the decision window.
*** The user selects "Feb" for the "Designer" "Regular" session.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 17012 SET-VALUE KEY CVV 5 (Designer..Regular 4))

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (SET DECISION (Designer..Regular 4))

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bill Muraham Designer Designer 2 2 (MS Word Designer)
               (((1 1) (2 3)) ((2 1) (3 0))) (0 0) (2 1) (1 1 1) (NIL NIL NIL)
               1 NIL 1 NIL PART GRAPH)

>> KS: INFER-DEC-SET-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 1 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0)))
        POST-MSG on NEED Level, MSG: (CURSOR PLACE (ROW))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PKG NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM COMP-RT-TS)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 1)

>> KS: DESIGN-CURSOR-POSITION

>> KS: INFO-DISPLAY

*** The user selects "Jun" for the "Designer" "Expert" session.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 17240 SET-VALUE KEY CVV 5 (Designer..Expert 8))

>> KS: INFER-STEP-SEMANTICS
```

```
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (SET DECISION (Designer..Expert 8))

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bill Muraham Designer Designer 3 3 (MS Word Designer)
               (((1 1) (2 3)) ((2 1) (3 2))) (0 0) (2 2) (1 1 1) (NIL NIL NIL)
               1 NIL 1 NIL PART GRAPH)

>> KS: UPDATE-DEC-PATTERN-HIS
        Triggered by the Event in SEMANTICS Level
        POST-MSG on HISTORY Level, MSG: (DECISION-PATTERN ((PKG-DOWN . 1)))

>> KS: INFER-DECISION-PATTERN
        Triggered by the Event in HISTORY Level
        UPDATE-MSG on USER-CHAR Level, MSG: (DECISION-PATTERN PKG-DOWN NIL)

>> KS: INFER-DEC-SET-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 3 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT NIL) DISPLAY-FORM (POP-UP 0)))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL NIL NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM SCHEDULE)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 1)

>> KS: DECIDE-VIEW-POINT
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN VIEW-POINT BY-PERSON)

>> KS: INFO-DISPLAY
        Triggered by the Event in DESIGN Level
        SET-PARAMETER on NIL Level, MSG: (*CURRENT-INFO-ITEM* SCHEDULE)
        SET-PARAMETER on NIL Level, MSG: (*DEFAULT-ACCESS-ITEM* Bill Muraham)
        SET-PARAMETER on NIL Level, MSG: (*LOCAL-DISPLAY-FORM* 1)
        SET-PARAMETER on NIL Level, MSG: (*LOCAL-VIEW-POINT* BY-PERSON)

%%% The system displays "TS: Training Schedules; By Person (Table)" of "Bill Muraham."
*** The user hits the "END" key.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 17497 END KEY CVV 5 Bill Muraham)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (END DECISION NIL)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
```

```
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT

>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (FOLLOW-UP CONFIRM)

>> KS: HANDLE-FOLLOW-UP
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (CONTINUE STEP 5)

>> KS: UPDATE-STEP-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 NIL NIL NIL NIL NIL NIL NIL NIL NIL (1 1 1) (NIL NIL NIL)
               1 NIL 1 NIL PART GRAPH)

>> KS: INFER-STEP-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 0 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT NIL) DISPLAY-FORM (POP-UP 0)))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL NIL NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM SCHEDULE-WHOLE)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 0)

>> KS: DECIDE-VIEW-POINT
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN VIEW-POINT N/A)

>> KS: INFO-DISPLAY

%%% The screen goes back to Figure 28 on page 264.
*** The user selects "Bill Muraham".
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 17695 SELECT KEY MENU 5 Bill Muraham)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (START DECISION Bill Muraham)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bill Muraham NIL NIL NIL NIL (MS Word Designer)
               (((1 1) (2 3)) ((2 1) (3 2))) (2 2) (2 2) (1 1 1) (NIL NIL NIL)
               1 NIL 1 NIL PART GRAPH)

>> KS: INFER-DEC-START-GOAL
```

```
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 2 3)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT NIL) DISPLAY-FORM (POP-UP 0)))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL NIL NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM SCHEDULE-WHOLE)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 0)

>> KS: DECIDE-VIEW-POINT
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN VIEW-POINT N/A)

>> KS: INFO-DISPLAY

%%% The system displays "TS: Training Schedule; Whole (Table)."
*** The user hits the down-arrow key.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 17979 DOWN KEY CVV 5 5)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (CURSOR DOWN 5)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

*** The user hits the down-arrow key.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 17993 DOWN KEY CVV 5 8)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (CURSOR DOWN 8)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

*** The user hits the "END" key.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 18013 END KEY CVV 5 Bill Muraham)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (END DECISION NIL)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)
```

```
>> KS: UPDATE-DECISION-CONTEXT

>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (FOLLOW-UP CONFIRM)

>> KS: HANDLE-FOLLOW-UP
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (CONTINUE STEP 5)

>> KS: UPDATE-STEP-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 NIL NIL NIL NIL NIL NIL NIL NIL NIL (1 1 1) (NIL NIL NIL)
               1 NIL 1 NIL PART GRAPH)

>> KS: INFER-STEP-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 0 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT NIL) DISPLAY-FORM (POP-UP 0)))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL NIL NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM SCHEDULE-WHOLE)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 0)

>> KS: DECIDE-VIEW-POINT
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN VIEW-POINT N/A)

>> KS: INFO-DISPLAY

%%% The screen goes back to Figure 28 on page 264.
*** The user selects "Bob Hamilton".
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 18144 SELECT KEY MENU 5 Bob Hamilton)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (START DECISION Bob Hamilton)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bob Hamilton NIL NIL NIL NIL (MS Word Lotus 123 Designer)
               (((1 0)) ((1 0)) ((1 0))) (0 0 0) (0 0 0) (1 1 1) (NIL NIL NIL)
               1 NIL 1 NIL PART GRAPH)

>> KS: INFER-DEC-START-GOAL
        Triggered by the Event in SEMANTICS Level
```

```
        POST-MSG on GOAL Level, MSG: (DECISION 1 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0)))
        POST-MSG on NEED Level, MSG: (CURSOR PLACE (ROW))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PKG NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM COMP-RT-TS)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 1)

>> KS: DESIGN-CURSOR-POSITION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (CURSOR-POSITION 0)

>> KS: PLACE-CVV-CURSOR

>> KS: INFO-DISPLAY

%%% The system displays "Gap between TC and TS; By PKG (Graph)" of "MS Word."
*** The user selects "Jul" for the "MS Word" "Novice" session.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 18673 SET-VALUE KEY CVV 5 (MS Word..Novice 9))

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (SET DECISION (MS Word..Novice 9))

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bob Hamilton MS Word MS Word 1 1 (MS Word Lotus 123 Designer)
               (((1 3)) ((1 0)) ((1 0))) (0 0 0) (2 0 0) (1 1 1) (NIL NIL NIL)
               1 NIL 1 NIL PART GRAPH)

>> KS: INFER-DEC-SET-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 1 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0)))
        POST-MSG on NEED Level, MSG: (CURSOR PLACE (ROW))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PKG NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM COMP-RT-TS)

>> KS: DECIDE-DISPLAY-FORM
```

```
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 1)

>> KS: DESIGN-CURSOR-POSITION

>> KS: INFO-DISPLAY
        Triggered by the Event in DESIGN Level
        SET-PARAMETER on NIL Level, MSG: (*CURRENT-INFO-ITEM* COMP-RT-TS)
        SET-PARAMETER on NIL Level, MSG: (*DEFAULT-ACCESS-ITEM* Lotus 123)
        SET-PARAMETER on NIL Level, MSG: (*LOCAL-DISPLAY-FORM* 1)
        SET-PARAMETER on NIL Level, MSG: (*LOCAL-VIEW-POINT* BY-PKG)

*** The user hits the up-arrow key.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 18760 UP KEY CVV 5 4)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (CURSOR UP 4)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: INFER-CURSOR-MOVE-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 1 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0)))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PKG NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM COMP-RT-TS)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 1)

>> KS: INFO-DISPLAY

*** The user hits the down-arrow key.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 18834 DOWN KEY CVV 5 7)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (CURSOR DOWN 7)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: INFER-CURSOR-MOVE-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 1 0)

>> KS: SUPPORT-DECISION-MAKING
```

```
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0)))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PKG NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM COMP-RT-TS)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 1)

>> KS: INFO-DISPLAY

*** The user selects "Aug" for the "Lotus 123" "Novice" session.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 19015 SET-VALUE KEY CVV 5 (Lotus 123..Novice 10))

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (SET DECISION (Lotus 123..Novice 10))

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bob Hamilton Lotus 123 Lotus 123 1 1 (MS Word Lotus 123 Designer)
               (((1 3)) ((1 3)) ((1 0))) (0 0 0) (2 2 0) (1 1 1) (NIL NIL NIL)
               1 NIL 1 NIL PART GRAPH)

>> KS: INFER-DEC-SET-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 1 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PKG) DISPLAY-FORM (POP-UP 0)))
        POST-MSG on NEED Level, MSG: (CURSOR PLACE (ROW))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PKG NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM COMP-RT-TS)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 1)

>> KS: DESIGN-CURSOR-POSITION

>> KS: INFO-DISPLAY
        Triggered by the Event in DESIGN Level
        SET-PARAMETER on NIL Level, MSG: (*CURRENT-INFO-ITEM* COMP-RT-TS)
        SET-PARAMETER on NIL Level, MSG: (*DEFAULT-ACCESS-ITEM* Designer)
        SET-PARAMETER on NIL Level, MSG: (*LOCAL-DISPLAY-FORM* 1)
        SET-PARAMETER on NIL Level, MSG: (*LOCAL-VIEW-POINT* BY-PKG)
```

```
*** The user selects "Jul" for the "Lotus 123" "Novice" session.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 19145 SET-VALUE KEY CVV 5 (Designer..Novice 9))

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (SET DECISION (Designer..Novice 9))

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bob Hamilton Designer Designer 1 1 (MS Word Lotus 123 Designer)
               (((1 3)) ((1 3)) ((1 3))) (0 0 0) (2 2 2) (1 1 1) (NIL NIL NIL)
               0 ((Bob Hamilton (9))) 1 NIL PART GRAPH)

>> KS: UPDATE-DEC-PATTERN-HIS
        Triggered by the Event in SEMANTICS Level
        UPDATE-MSG on HISTORY Level, MSG: (DECISION-PATTERN ((PKG-DOWN . 2)) NIL)

>> KS: INFER-DECISION-PATTERN
        Triggered by the Event in HISTORY Level
        UPDATE-MSG on USER-CHAR Level, MSG: (DECISION-PATTERN PKG-DOWN NIL)

>> KS: INFER-DEC-SET-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 2 2)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT BY-PERSON) DISPLAY-FORM (POP-UP 0)))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL BY-PERSON NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM SCHEDULE-BY-PERSON)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 1)

>> KS: INFO-DISPLAY
        Triggered by the Event in DESIGN Level
        SET-PARAMETER on NIL Level, MSG: (*CURRENT-INFO-ITEM* SCHEDULE-BY-PERSON)
        SET-PARAMETER on NIL Level, MSG: (*DEFAULT-ACCESS-ITEM* Bob Hamilton)
        SET-PARAMETER on NIL Level, MSG: (*LOCAL-DISPLAY-FORM* 1)
        SET-PARAMETER on NIL Level, MSG: (*LOCAL-VIEW-POINT* BY-PERSON)

*** The user hits the down-arrow key.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 19313 DOWN KEY CVV 5 7)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (CURSOR DOWN 7)

>> KS: UPDATE-INT-HISTORY
```

```
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

*** The user hits the down-arrow key.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 19351 DOWN KEY CVV 5 10)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (CURSOR DOWN 10)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

*** The user selects "Oct" for the "Designer" "Novice" session.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 19479 CHANGE-VALUE KEY CVV 5 (Designer..Novice 12))

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (CHANGE DECISION (Designer..Novice 12))

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 Bob Hamilton Designer NIL 1 1 (MS Word Lotus 123 Designer)
               (((1 3)) ((1 3)) ((1 3))) (0 0 0) (2 2 2) (1 1 1) (NIL NIL NIL)
               1 NIL 1 NIL PART GRAPH)

>> KS: INFER-DEC-CHANGE-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 3 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT NIL) DISPLAY-FORM (POP-UP 0)))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL NIL NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM SCHEDULE)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 1)

>> KS: DECIDE-VIEW-POINT
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN VIEW-POINT BY-PERSON)

>> KS: INFO-DISPLAY

*** The user hits the "END" key.
>> KS: ACCEPT-REPORT
```

```
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (COMMAND 19581 END KEY CVV 5 Bob Hamilton)

>> KS: INFER-STEP-SEMANTICS
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (END DECISION NIL)

>> KS: UPDATE-INT-HISTORY
        Triggered by the Event in INPUT Level
        UPDATE-MSG on HISTORY Level, MSG: (T-NO-OF-COMMANDS 1 +)
        UPDATE-MSG on HISTORY Level, MSG: (S-NO-OF-COMMANDS 1 +)

>> KS: UPDATE-DECISION-CONTEXT

>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (FOLLOW-UP CONFIRM)

>> KS: HANDLE-FOLLOW-UP
        Triggered by the Event in INPUT Level
        POST-MSG on SEMANTICS Level, MSG: (CONTINUE STEP 5)

>> KS: UPDATE-STEP-CONTEXT
        Triggered by the Event in SEMANTICS Level
        POST-MSG on CONTEXT Level,
         MSG: (5 NIL NIL NIL NIL NIL NIL NIL NIL NIL (1 1 1) (NIL NIL NIL)
               1 NIL 1 NIL PART GRAPH)

>> KS: INFER-STEP-GOAL
        Triggered by the Event in SEMANTICS Level
        POST-MSG on GOAL Level, MSG: (DECISION 0 0)

>> KS: SUPPORT-DECISION-MAKING
        Triggered by the Event in GOAL Level
        POST-MSG on NEED Level, MSG: (INFO SET (ITEM (VIEW-POINT NIL) DISPLAY-FORM (POP-UP 0)))

>> KS: DESIGN-INFORMATION
        Triggered by the Event in NEED Level
        POST-MSG on DESIGN Level, MSG: (INFO-DESIGN (NIL NIL NIL 0))

>> KS: REASON-INFO-ITEM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN ITEM SCHEDULE-WHOLE)

>> KS: DECIDE-DISPLAY-FORM
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN DISPLAY-FORM 0)

>> KS: DECIDE-VIEW-POINT
        Triggered by the Event in DESIGN Level
        FILL-MSG on DESIGN Level, MSG: (INFO-DESIGN VIEW-POINT N/A)

>> KS: INFO-DISPLAY

*** The user ends the session.
>> KS: ACCEPT-REPORT
        Triggered by the Event in NIL Level
        POST-MSG on INPUT Level, MSG: (END-SESSION)

>> KS: SAVE-SESSION
```

Appendix G. MSLTRAIN Screens in the Sample Session

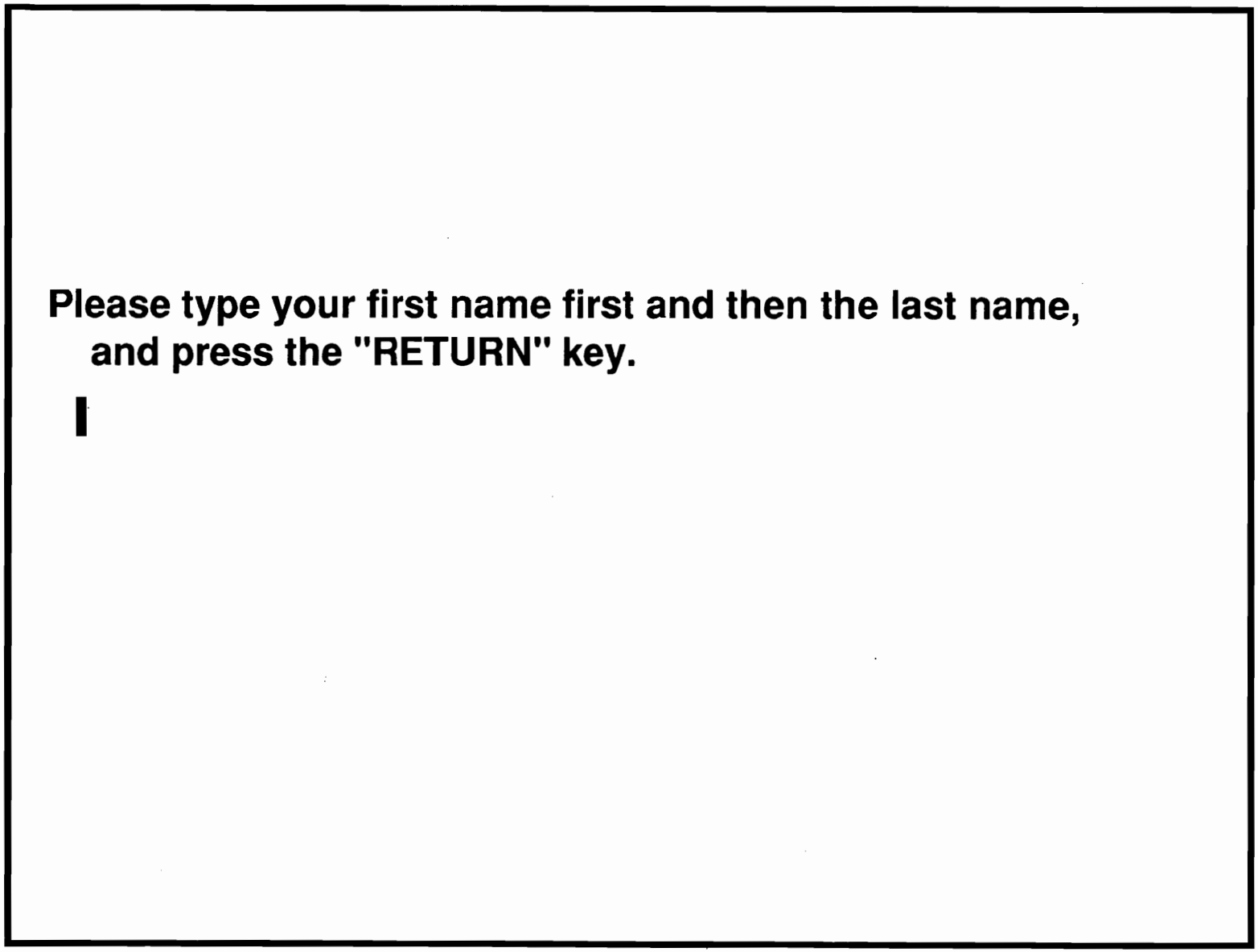


Figure 26. MSLTRAIN asks the user's name.

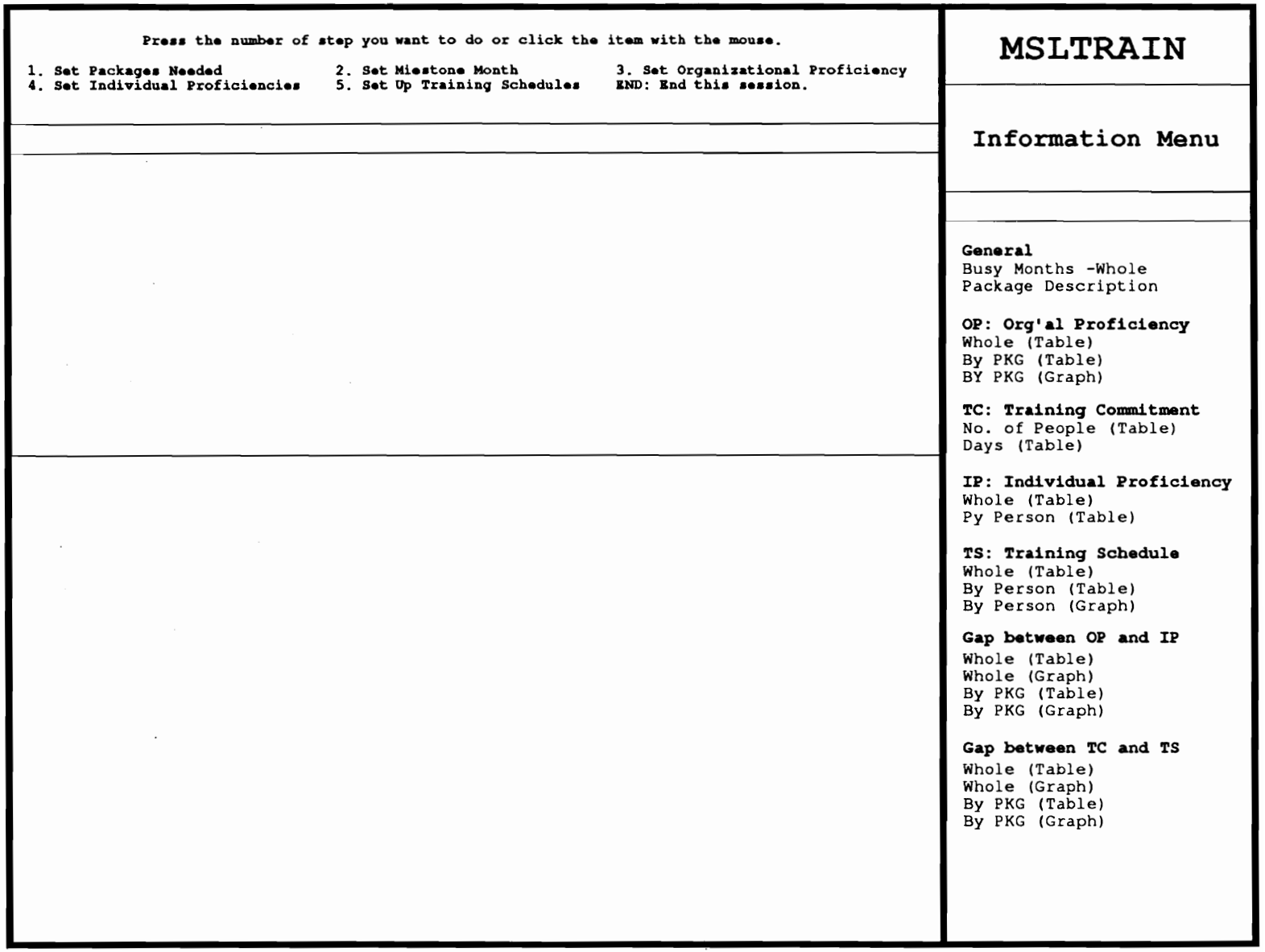


Figure 27. The manager starts Step 5 of MSLTRAIN.

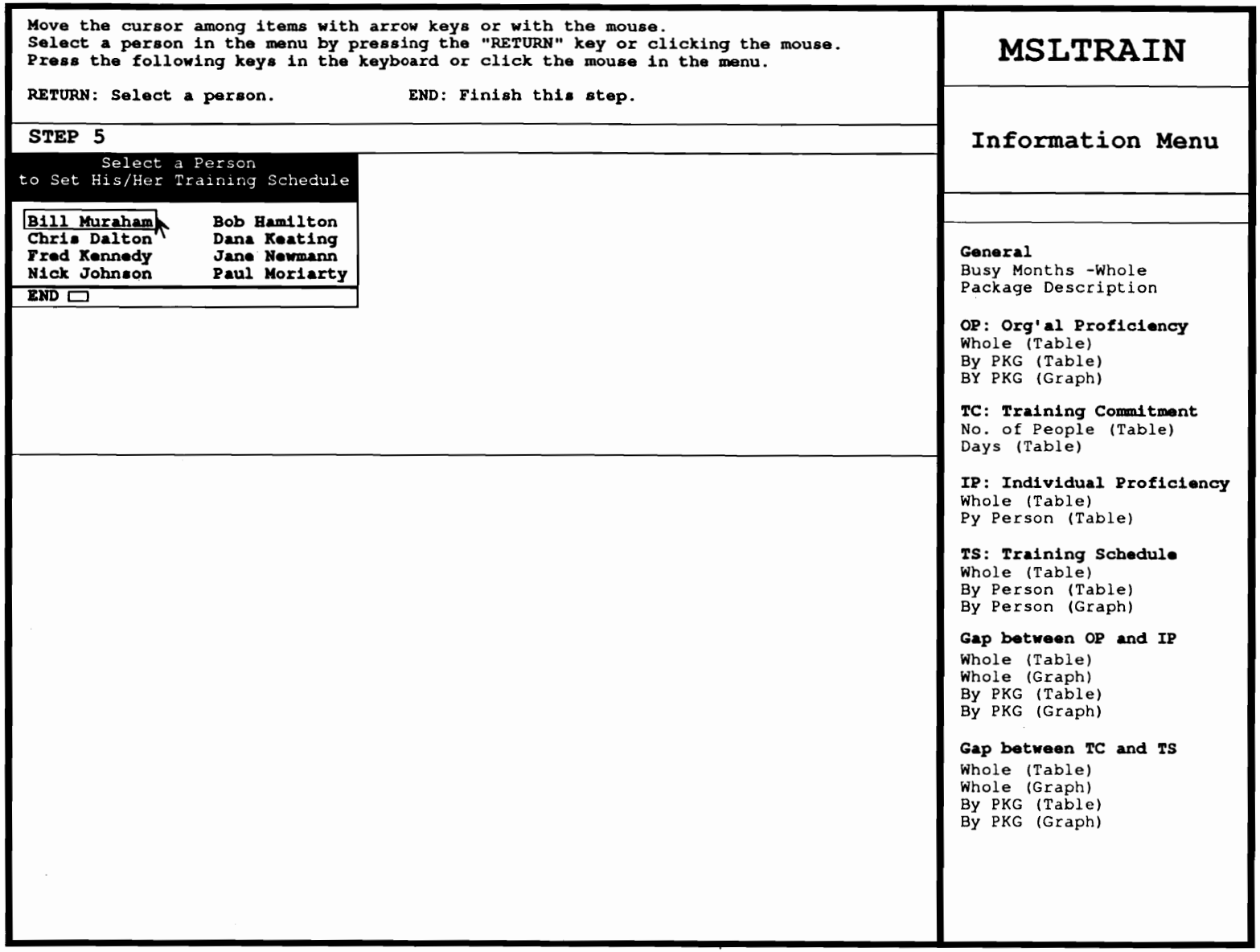


Figure 28. The manager selects a person to schedule him or her.

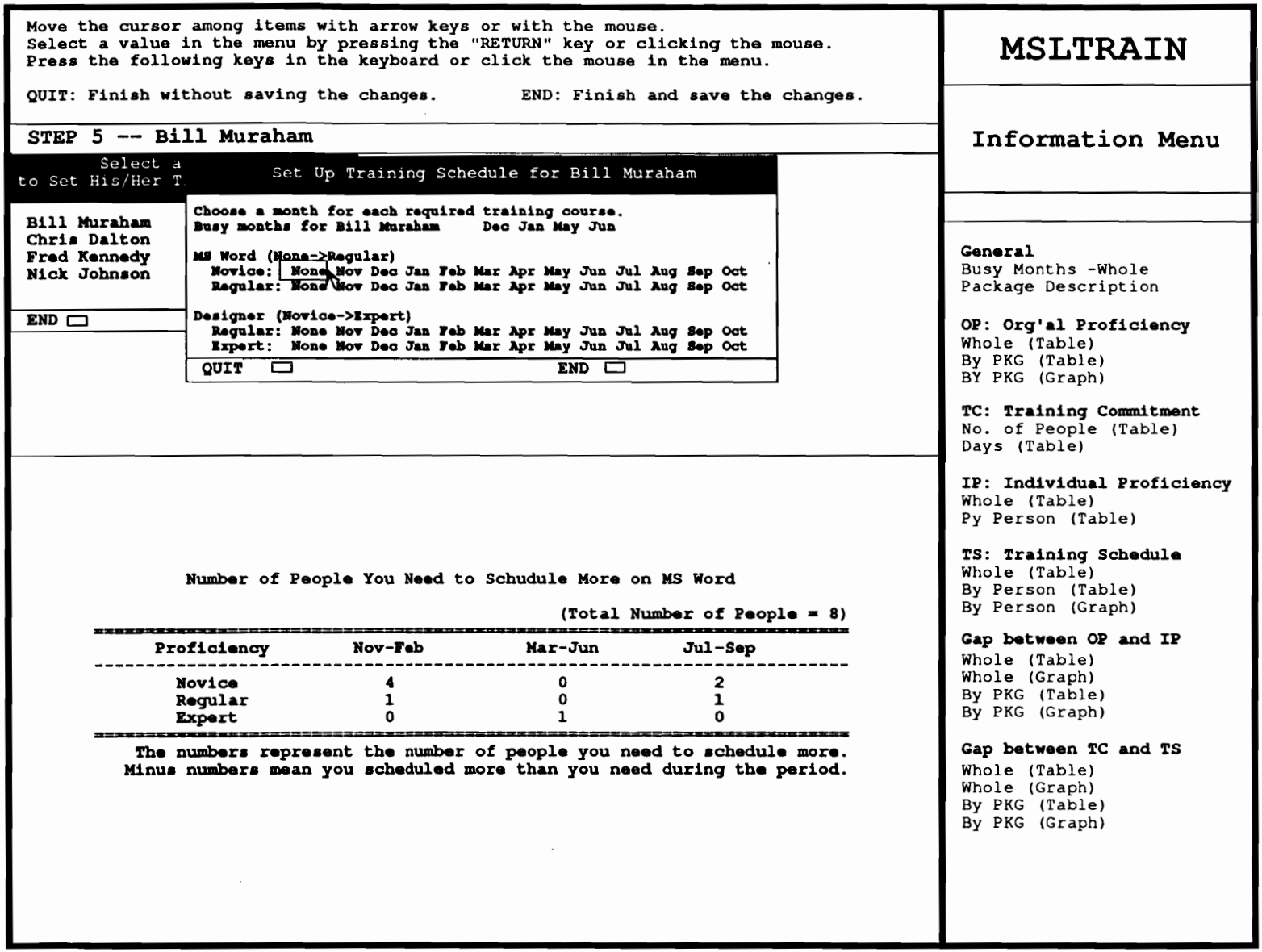


Figure 29. The manager selects a month for each session the person should take.

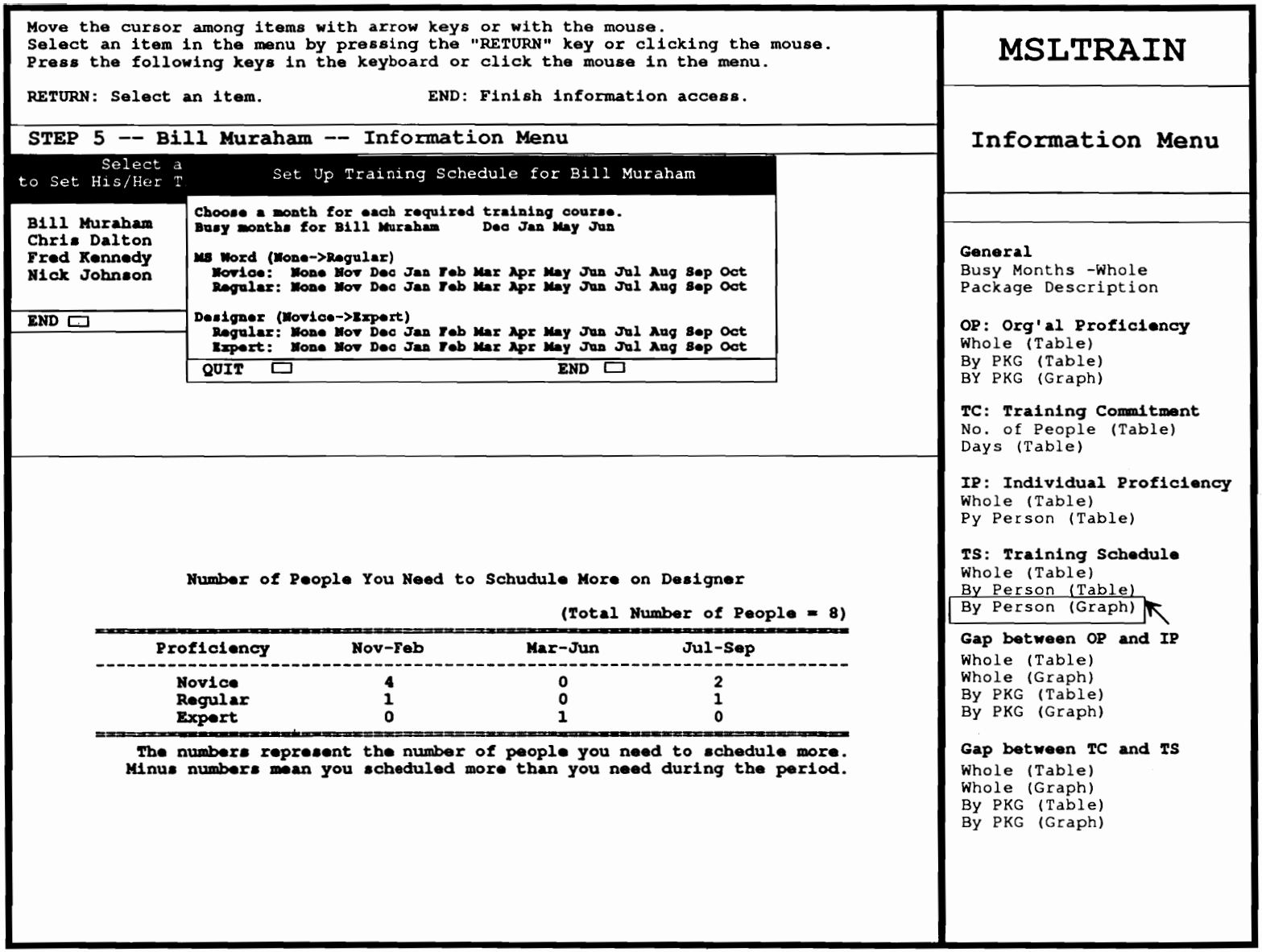


Figure 30. The manager started to access information.

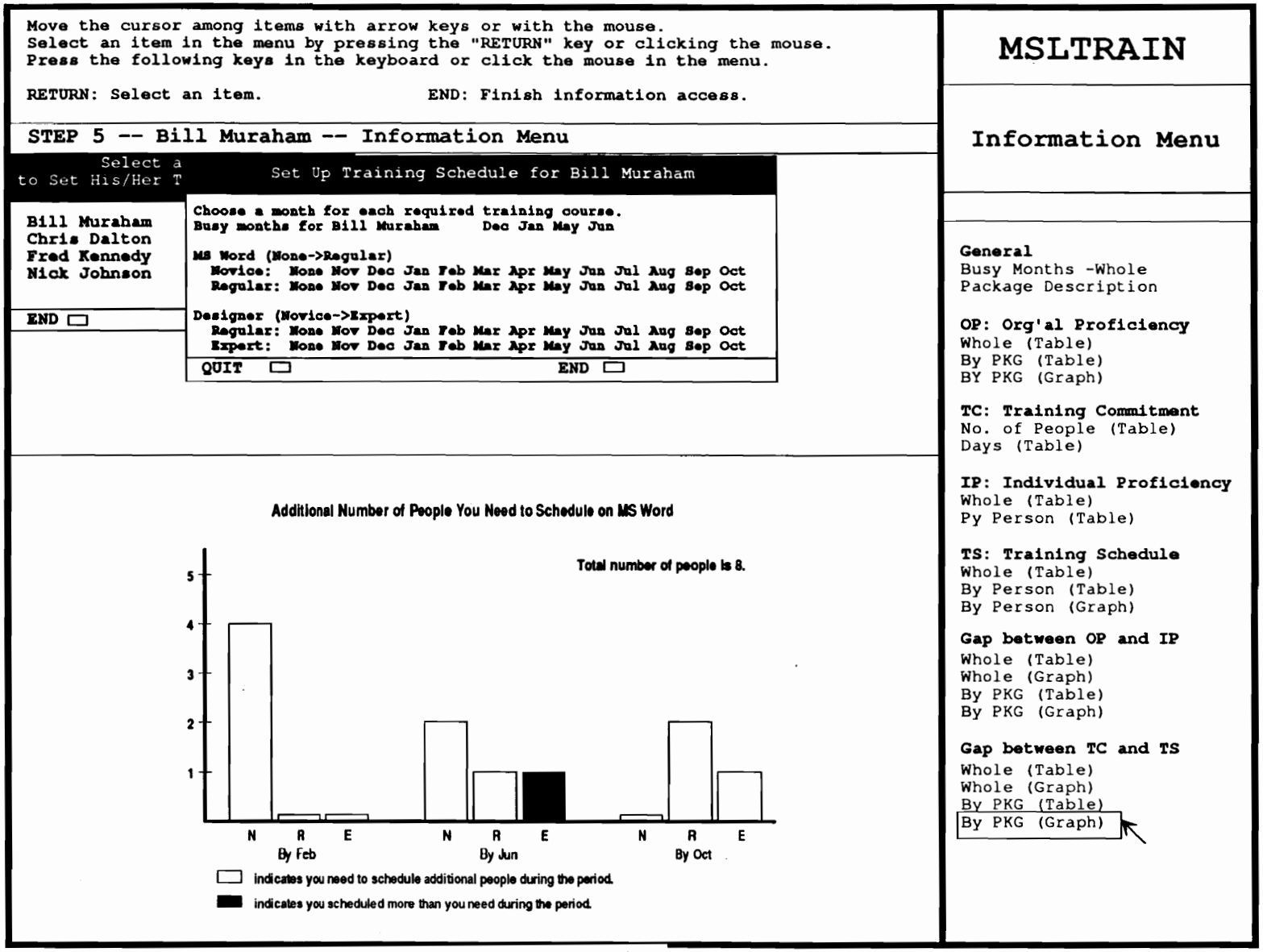


Figure 31. The manager requested an information item.

Appendix H. User Perception Questionnaire

Listed below is a series of statements. Please indicate how strongly you agree or disagree by circling the appropriate number.

	STRONGLY DISAGREE	DISAGREE	TEND TO DISAGREE	NEITHER DISAGREE OR AGREE	TEND TO AGREE	AGREE	STRONGLY AGREE
1. The system helped me schedule training needs.	1	2	3	4	5	6	7
2. I am likely to use the system again.	1	2	3	4	5	6	7
3. The system helped me spend time efficiently.	1	2	3	4	5	6	7
4. The system helped me hit as few keys as possible.	1	2	3	4	5	6	7
5. The system provided me information I needed for decision making.	1	2	3	4	5	6	7
6. It was easy to understand information the system provided.	1	2	3	4	5	6	7
7. The system provided me information I didn't use.	1	2	3	4	5	6	7
8. The responses of the system often distracted me.	1	2	3	4	5	6	7
9. I often felt overloaded with information.	1	2	3	4	5	6	7
10. The system provided me with enough information to make scheduling decisions.	1	2	3	4	5	6	7
11. Whenever I hit a key, the system responded as I expected.	1	2	3	4	5	6	7
12. The responses of the system were consistent.	1	2	3	4	5	6	7
13. The system responded quickly enough for me.	1	2	3	4	5	6	7
14. The screen changed too often for me.	1	2	3	4	5	6	7
15. The system seemed to anticipate what I intended to do.	1	2	3	4	5	6	7

	STRONGLY DISAGREE	DISAGREE	TEND TO DISAGREE	NEITHER DISAGREE OR AGREE	TEND TO AGREE	AGREE	STRONGLY AGREE
16. I felt confident using the system.	1	2	3	4	5	6	7
17. Overall, the system was easy to use.	1	2	3	4	5	6	7

18. Overall, how would you rate the system ? Please circle the most accurate one.

excellent good fair poor unsatisfactory

19. Please use this space for any comments you have regarding the system.

Thanks for your cooperation.

Appendix I. Introduction Packet

Decision Support System Study

Introduction

The purpose of this study is to evaluate two versions of a decision support system called MSLTRAIN. The study is being conducted by Management Systems Laboratories, a research arm of the Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24060 (Phone number: (703) 231-3501), funded on DOE Grant II. The research team consists of Kwang S. Lee, a graduate student in Industrial and Systems Engineering, under the direction of Dr. Harold A. Kurstedt, Jr., the principal investigator.

The experiment involves two sessions that will occur on separate occasions, which will be scheduled at your convenience. In the first session, you will be trained on the computer equipment you will use and the experimental task you will do. In the second session, you will do the experimental task: make scheduling decisions using two versions of MSLTRAIN. We are interested in your reaction to them.

After reading the introductory material, you will be given an informed consent form. If you understand what this experiment entails and agree to participate in it, you must sign this form.

Experimental Task

You will assume you are the manager of an organization comprising eight people in addition to yourself. Using MSLTRAIN you will decide when each person in your organization will be trained to predetermined computer packages. Information you need to make those decisions will be provided and installed in MSLTRAIN before the experiment. During the task, you can access this information whenever you want. For the completion of the task, you should satisfy several constraints that will be given before the experiment.

Experimental Procedure

In the first session, the experimenter will instruct you on how to use the keyboard and demonstrate how to use MSLTRAIN. The experimenter will explain the experimental task and the information

items you may need during the task. You can ask any question at this time. Then, using one version of MSLTRAIN, you will practice the experimental task with the experimenter's help and explanation. After having 5 minutes break, you will practice the experimental task using the other version of MSLTRAIN.

The second session will occur preferably on the next day after the first session. In the second session, you will do the experimental task. The experimenter will load one of the MSLTRAIN's and give you instruction about the situation you'll assume and constraints you should satisfy. You will schedule the training needs of your organization. After you complete the task, you will fill out a questionnaire about your satisfaction with the system you used. The experimenter will load the other version of MSLTRAIN. You will have 5 minutes break. Then you will schedule the training needs of your organization with the other MSLTRAIN. After the completion of the second run, you will fill out the same questionnaire you've filled out after the first run.

You will earn $25: $10 for the first session and $15 for the second session. The experimental sessions shouldn't last longer than an hour to an hour and half.

Additional Information

At any time during the study, if you no longer wish to continue, you have the right to terminate your participation; you will be compensated for your participation up to that point - $5 per hour.

If you have any questions about the study or your rights as a participant after reading the attached consent form, please do not hesitate to ask. We will answer your questions honestly and as openly as possible. We ask that you do not discuss the details of this study with any person, particularly those who may participate, as prior knowledge of seemingly incidental facts might compromise the data. It is expected that all data will be collected by Dec. 1, 1990; you may feel free to discuss the study with any persons after that time. All data will be analyzed with anonymity. Immediately upon completion of your experimental sessions, your data will be identified only by a randomly assigned serial number.

Participant's Informed Consent

1. You are being asked to participate in a research project whose purpose and description are contained in the document "Decision Support System Study", which you have already read.

2. A potential discomfort involved in this study includes fatigue due to interacting with a CRT for the experimental sessions. However, you will be given ample time to rest between sessions and, although there is no time limit, the experimental sessions shouldn't last longer than an hour to an hour and half total.

3. You may withdraw from participation at any time for any reason. If you decide to terminate the experiment, inform the researcher and he will pay you for the length of time you have participated.

4. You may withdraw your data from the experiment if you feel that you should for any reason. In general, data are processed and analyzed after a subject has completed the experiment. At that time, all identification information will be removed and the data treated with anonymity. Therefore, if you wish to withdraw your data, you must do so immediately after your participation is completed.

5. You should not volunteer for participation in this study if you are under 18 years of age.

Please note that the principal investigator and his graduate assistant will answer any questions you may have about this study. You should not sign this consent form unless you are sure you are certain you understand all of the previous descriptions and conditions.

You can address further comments or questions to Dr. Stout, Chairman of the Institutional Review Board for the Use of Human Subjects in Research (Phone number: (703) 231-5281).

Your signature below indicates that you have read and understood the details of the experiment and your right as a participant (as stated above), and that you consent to participate.

_____

Participant's Signature Date

_____

Research Team Member's Signature Date

Appendix J. Instruction Packet

Scenario

Today is Oct. 15, 1990. You are the manager of an organization comprising eight people besides yourself. You recently realized your people need more knowledge of computer packages. You concluded your people need training on "Microsoft Word", "Lotus 123", and "Designer". You defined three levels of proficiency:

- "Novice" who has a little knowledge of the package and uses it occasionally such as three times or less a week,
- "Regular" who has relatively much knowledge of the package and uses it everyday, and
- "Expert" who has knowledge of the package enough to consult problems other people have on the package.

A person who knows nothing about the package is referred to "None". Currently your organization has one novice and one regular on "MS Word", two novices on "Lotus 123". After considering all situations, you decided your organization should have four novices, two regulars, and one expert on each package by the end of October 1991. To get this goal, you set milestones like those in Table 1.

Table 1. Milestones in terms of the number of people proficient on the software packages (example)

Desired Organizational Proficiency

(Total Number of People = 8)

Package	By Feb			By Jun			By Oct		
	N	R	E	N	R	E	N	R	E
MS Word	3	2	0	3	1	1	4	2	1
Lotus 123	2	2	0	3	3	0	4	2	1
Designer	3	1	0	4	0	1	4	2	1

N: novice R: regular E: expert

For example, you want your organization to have three novices and three regulars on "Lotus 123" by the end of June 1991.

After setting desired organizational proficiency, you changed your perspective to individuals in your organization. You set the proficiency levels you desired each individual to have by the end of October 1991, like those in Table 2.

Table 2. Desired individual proficiencies (example)

Desired Individual Proficiencies

Name	MS Word	Lotus 123	Designer
Bill Muraham	* → R	*	N → E
Bob Hamilton	* → N	* → N	* → N
Chris Dalton	* → N	* → N	* → R
Dana Keating	*	* → N	* → N
Fred Kennedy	* → N	N → E	* → N
Jane Newmann	N → R	* → R	*
Nick Johnson	* → N	N → R	* → N
Paul Moriarty	R → E	* → N	* → R

*: none N: novice R: regular E: expert
The arrows (→) indicate upgrade.

For example, you want "Bill Muraham" to be at the regular proficiency level on "MS Word" and at the expert proficiency level on "Designer" by the end of October 1991. "Bill Muraham" is currently a novice on "Designer". Therefore, "Bill Muraham" needs four training sessions: a novice and a regular session on "MS Word" and a regular and an expert session on "Designer".

Now you are going to set months in the following twelve months (from Nov. 1990 to Oct. 1991) when each individual in your organization will get the training sessions he or she needs. For example, you may set "Bill Muraham" to be trained on a "MS Word" "Novice" session during January 1991.

Experimental Task

1. Hit the "SPACE" bar to start the experiment.

2. Type your first and last name and hit the "RETURN" key if the system asks your name.

3. The screen shows a menu, listing the names of people in the organization, on the upper left corner of the screen.

4. To select a person whose training schedule you want to set, place the cursor on the person's name in the menu, and hit the "RETURN" key.

5. The screen shows a decision window listing training sessions you desire the person to take within the following next 12 months.

6. To select a month for each session, place the cursor on the month and hit the "RETURN" key.

7. If you select a month when the person is expected to be too busy to get training, the system shows a pop-up message saying that. To erase the message, hit the "SPACE" bar. The system doesn't allow you schedule on the busy months.

8. To finish decision about the person, hit the "END" key.

9. Continue until you make all scheduling decisions on all people in the organization.

10. Whenever you want information, hit the "INFO" key. Then the cursor moves to the "Information Menu" at the right side of the screen. Move the cursor to the information item you want to see and hit the "RETURN" key. Then the information is displayed on the "Information Display Pane" at the lower side of the screen. Access as many information items as you want. To finish accessing information, hit the "END" key.

PLEASE DO EXACTLY WHAT THE FOLLOWING INSTRUCTION SAYS.

1. Start MSLTRAIN.
2. Select "Bill Muraham".
3. Access the information, "Gap between RT and TS; Whole (Graph)".
4. End information access.
5. Select the month "Mar" for the "MS Word" novice session.
6. Select the month "Jul" for the "MS Word" regular session.
7. Select the month "Jan" for the "Designer" regular session.
8. Erase the pop-up message.
9. Select the month "Feb" for the "Designer" regular session.
10. Select the month "Jun" for the "Designer" expert session.
11. Access the information, "TS: Training Schedules; By Person (Table)".
12. End information access.
13. End decision about "Bill Muraham".
14. Access the information, "TS: Training Schedules; Whole (Table)".
15. End information access.

PLEASE DO EXACTLY WHAT THE FOLLOWING INSTRUCTION SAYS.

1. Start MSLTRAIN.
2. Answer your name.
3. Select "Bill Muraham".
4. Select the month "Mar" for the "MS Word" novice session.
5. Select the month "Jul" for the "MS Word" regular session.
6. Select the month "Jan" for the "Designer" regular session.
7. Erase the pop-up message.
8. Select the month "Feb" for the "Designer" regular session.
9. Select the month "Jun" for the "Designer" expert session.
10. End decision about "Bill Muraham".

Constraints

To complete the task, you should satisfy the following constraints:

- Satisfy the organizational milestones - The information item, "Gap between RT and TS; Whole (Table)" should have all zeros.

- Schedule a person only on the month when he or she is not expected to be too busy to get training - The system shows a pop-up message when you schedule a person during a busy month.

- Schedule a person only once during a month - The information item, "TS: Training Schedule; Whole (Table)", shows "*" when you scheduled more than one session for a person during a month.

- Schedule no more than seven person-days during a month - The information item, "TS: Training Schedule; Whole (Table)", shows the total scheduled person-days during a month in the "Total Days" row.

Suggested Strategy

- Schedule your people in alphabetical order.
- To schedule a session, select an appropriate milestone period first based on the suggested strategy below and then select an appropriate month during the period.
  - If a person needs a "Novice" session only on a package, select the last milestone period during which you need to schedule more people.
  - If a person needs a "Regular" or "Expert" session only on a package, select the earliest milestone period during which you need to schedule more people.
  - If a person needs two sessions on a package, such as "Novice" and "Regular" or "Regular" and "Expert", select the earliest milestone period for the first session and select the last milestone period for the second session.
- To see during which milestone period you need to schedule more, refer the information items under "Gap between RT and TS".

Appendix K. Subject Information Questionnaire

Subject Information Questionnaire

Please circle the most accurate response.

1. **Gender:** Male Female

2. **Age:** Under 20 20-25 26-30 Over 30

3. **Major:** HF Manufacturing OR MSE Others

4. **How experienced are you using computers ?**

no experience	some experience	experienced	very experienced

5. **Have you heard about the term "Responsive System" ?**

Yes No

Appendix L. SAS Program for the ANOVA Procedure

```
OPTIONS LINESIZE=80;
CMS FI EXPDAT1 DISK QUEST DATA A;
CMS FI EXPDAT2 DISK EXP DATA A;
DATA QUES (KEEP=N VER ORDER RATE SATI);
  INFILE EXPDAT1;
  INPUT N VER Q1-Q18;
  Q7 = 8 - Q7;
  Q8 = 8 - Q8;
  Q9 = 8 - Q9;
  Q14 = 8 - Q14;
  RATE = 6 - Q18;
  ADVG = SUM (Q1,Q3,Q4,Q5,Q6,Q10,Q15);
  DISG = SUM (Q7,Q8,Q9,Q11,Q12,Q13,Q14);
  EASE = SUM(Q16,Q17,Q2);
  SATI = (ADVG+DISG+EASE)/17;
  ORDER = MOD (N,2);
DATA EXP (KEEP=N VER ORDER TCT TAT TCK TAK);
  INFILE EXPDAT2;
  INPUT N VER SC TCT TAT TR TCK TAK KR;
  TCT = TCT / 60;
  TAT = TAT / 60;
  ORDER = MOD (N,2);
  IF SC = 4 THEN DELETE;
PROC SORT DATA=QUES;
  BY N VER;
PROC SORT DATA=EXP;
  BY N VER;
DATA WHOLE;
  MERGE QUES EXP; BY N VER;
PROC SORT;
  BY ORDER VER;
PROC ANOVA;
  CLASS N ORDER VER;
  MODEL TCT TAT TCK TAK RATE SATI =
        ORDER VER N(ORDER) ORDER*VER
VER*N(ORDER);
  TEST H = ORDER          E = N(ORDER);
  TEST H = VER ORDER*VER  E = VER*N(ORDER);
```

Appendix M. Raw User Performance and User Perception Data

USER PERFORMANCE RAW DATA

OBS	SUBJECT	ORDER	RESP	TCT	TAT	TCK	TAK
1	7	O FIRST	O-MSLTRAIN	14.2500	2.7833	436	95
2	7	O FIRST	R-MSLTRAIN	12.2833	0.4000	292	16
3	5	O FIRST	O-MSLTRAIN	24.0833	6.4833	531	122
4	5	O FIRST	R-MSLTRAIN	12.6333	0.8167	298	6
5	6	R FIRST	R-MSLTRAIN	11.7000	0.0000	259	0
6	6	R FIRST	O-MSLTRAIN	11.4333	2.8833	357	105
7	1	O FIRST	O-MSLTRAIN	19.9667	4.3167	400	110
8	1	O FIRST	R-MSLTRAIN	10.8333	0.4500	234	6
9	4	R FIRST	R-MSLTRAIN	13.5500	0.0000	342	0
10	4	R FIRST	O-MSLTRAIN	12.4833	1.4667	351	28
11	9	O FIRST	O-MSLTRAIN	13.5167	2.3500	297	51
12	9	O FIRST	R-MSLTRAIN	9.1167	0.0000	216	0
13	11	O FIRST	O-MSLTRAIN	14.2333	2.6167	321	43
14	11	O FIRST	R-MSLTRAIN	12.1000	0.0000	261	0
15	16	R FIRST	R-MSLTRAIN	12.5000	0.0000	287	0
16	16	R FIRST	O-MSLTRAIN	9.9667	1.8167	321	93
17	2	R FIRST	R-MSLTRAIN	12.1333	0.0000	238	0
18	2	R FIRST	O-MSLTRAIN	11.0833	4.3000	302	89
19	8	R FIRST	R-MSLTRAIN	16.5333	0.8667	282	9
20	8	R FIRST	O-MSLTRAIN	11.0500	0.5667	273	17
21	10	R FIRST	R-MSLTRAIN	9.3500	0.0000	230	0
22	10	R FIRST	O-MSLTRAIN	10.4833	0.5667	298	17
23	12	R FIRST	R-MSLTRAIN	7.0667	0.0000	288	0
24	12	R FIRST	O-MSLTRAIN	6.8333	0.8000	329	32
25	19	O FIRST	O-MSLTRAIN	11.1167	2.3167	460	83
26	19	O FIRST	R-MSLTRAIN	10.1833	0.0000	345	0
27	15	O FIRST	O-MSLTRAIN	25.0500	12.5333	491	184
28	15	O FIRST	R-MSLTRAIN	15.5500	0.0000	245	0
29	17	O FIRST	O-MSLTRAIN	12.7833	0.8833	336	32
30	17	O FIRST	R-MSLTRAIN	10.2000	0.1000	344	2
31	21	O FIRST	O-MSLTRAIN	17.4167	2.6000	459	102
32	21	O FIRST	R-MSLTRAIN	10.1333	0.4667	269	21
33	22	R FIRST	R-MSLTRAIN	12.3833	0.7333	273	13
34	22	R FIRST	O-MSLTRAIN	19.5333	4.7000	512	137
35	14	R FIRST	R-MSLTRAIN	8.0833	0.0000	252	0
36	14	R FIRST	O-MSLTRAIN	8.2500	1.7000	311	78
37	18	R FIRST	R-MSLTRAIN	10.5167	0.2667	228	3
38	18	R FIRST	O-MSLTRAIN	10.6667	2.0500	345	109
39	13	O FIRST	O-MSLTRAIN	17.3167	1.4500	436	45
40	13	O FIRST	R-MSLTRAIN	11.1833	0.0000	256	0

USER PERCEPTION RAW DATA

OBS	SUBJECT	ORDER		RESP	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18
1	7	O	FIRST	O-MSLTRAIN	2	2	1	2	6	3	3	6	6	5	5	5	6	3	3	3	3	4
2	7	O	FIRST	R-MSLTRAIN	6	6	6	6	6	6	2	4	2	6	6	6	5	3	6	6	6	2
3	5	O	FIRST	O-MSLTRAIN	5	5	4	3	6	2	7	5	2	6	3	5	7	5	2	3	3	4
4	5	O	FIRST	R-MSLTRAIN	6	6	7	5	6	4	3	2	2	5	5	6	5	3	6	6	5	2
5	6	R	FIRST	O-MSLTRAIN	6	6	5	5	5	7	6	3	2	4	7	6	6	2	5	6	6	2
6	6	R	FIRST	R-MSLTRAIN	6	6	7	7	5	4	6	2	3	6	6	7	6	3	4	6	7	2
7	1	O	FIRST	O-MSLTRAIN	6	5	5	3	5	4	1	1	7	7	5	6	7	7	1	4	3	3
8	1	O	FIRST	R-MSLTRAIN	7	7	7	7	7	7	1	2	2	6	6	6	7	1	2	6	6	1
9	4	R	FIRST	O-MSLTRAIN	5	4	3	4	6	5	7	5	5	6	4	5	6	3	2	3	5	3
10	4	R	FIRST	R-MSLTRAIN	6	6	7	7	6	5	5	3	4	6	5	5	6	4	5	5	6	2
11	9	O	FIRST	O-MSLTRAIN	7	7	7	6	7	7	2	1	1	7	7	7	7	1	4	7	7	1
12	9	O	FIRST	R-MSLTRAIN	7	7	7	7	7	7	1	1	1	7	7	7	7	1	7	7	7	1
13	11	O	FIRST	O-MSLTRAIN	6	6	6	4	6	5	2	2	2	4	6	6	6	2	3	5	5	2
14	11	O	FIRST	R-MSLTRAIN	6	6	7	6	7	6	2	2	2	7	6	6	6	2	6	6	7	1
15	16	R	FIRST	O-MSLTRAIN	7	6	5	1	4	7	4	6	2	7	7	7	5	1	1	4	7	3
16	16	R	FIRST	R-MSLTRAIN	7	6	4	7	7	7	1	4	1	7	7	7	7	1	7	5	7	2
17	2	R	FIRST	O-MSLTRAIN	5	5	6	2	6	5	5	4	2	6	6	6	6	4	4	5	6	2
18	2	R	FIRST	R-MSLTRAIN	6	5	4	5	6	5	6	4	2	7	4	2	2	4	4	4	5	3
19	8	R	FIRST	O-MSLTRAIN	6	6	6	5	5	5	2	2	2	5	5	5	2	1	2	6	6	2
20	8	R	FIRST	R-MSLTRAIN	6	6	6	7	6	5	5	3	3	6	5	6	6	2	5	5	5	2
21	10	R	FIRST	O-MSLTRAIN	6	5	3	2	5	3	2	4	2	6	7	6	6	2	7	3	3	2
22	10	R	FIRST	R-MSLTRAIN	7	5	7	6	7	7	2	2	1	7	6	6	6	2	7	7	7	1
23	12	R	FIRST	O-MSLTRAIN	6	1	3	1	1	6	1	3	1	1	7	7	3	1	1	6	3	3
24	12	R	FIRST	R-MSLTRAIN	5	1	5	7	5	6	1	1	1	3	7	7	5	2	5	7	6	2
25	19	O	FIRST	O-MSLTRAIN	6	6	5	6	7	5	6	2	1	5	7	7	6	2	2	6	7	1
26	19	O	FIRST	R-MSLTRAIN	7	7	6	7	7	6	6	4	3	6	6	6	6	3	3	7	7	1
27	15	O	FIRST	O-MSLTRAIN	7	6	5	3	7	6	3	4	5	7	7	7	6	4	3	4	5	2
28	15	O	FIRST	R-MSLTRAIN	7	7	7	6	7	6	2	4	2	7	6	7	7	3	7	6	6	1
29	17	O	FIRST	O-MSLTRAIN	7	6	7	7	6	7	2	6	2	7	7	7	7	2	5	7	7	1
30	17	O	FIRST	R-MSLTRAIN	7	7	7	7	7	7	1	1	1	7	6	7	7	3	7	7	7	1
31	21	O	FIRST	O-MSLTRAIN	3	4	3	5	5	6	6	1	2	4	5	6	6	1	2	5	6	2
32	21	O	FIRST	R-MSLTRAIN	5	5	5	7	7	6	2	1	3	6	6	6	6	3	2	6	7	2
33	22	R	FIRST	O-MSLTRAIN	5	3	3	2	5	3	2	4	2	6	5	7	6	2	2	3	3	2
34	22	R	FIRST	R-MSLTRAIN	7	6	7	6	6	6	4	5	2	6	5	7	2	3	4	6	7	1
35	14	R	FIRST	O-MSLTRAIN	5	3	3	1	3	6	3	4	5	6	3	6	6	4	2	3	3	3
36	14	R	FIRST	R-MSLTRAIN	6	6	6	6	6	6	3	2	2	5	5	7	5	2	2	7	7	1
37	18	R	FIRST	O-MSLTRAIN	6	5	5	4	4	6	4	2	3	5	5	6	6	2	2	5	6	2
38	18	R	FIRST	R-MSLTRAIN	6	5	6	7	5	6	2	2	3	6	5	6	7	2	7	6	7	1
39	13	O	FIRST	O-MSLTRAIN	5	3	2	1	4	3	6	6	2	3	5	5	4	5	1	3	2	4
40	13	O	FIRST	R-MSLTRAIN	6	6	6	7	7	7	3	2	1	7	7	7	7	2	7	7	6	2

Vita

KWANG SEOK LEE

PRESENT ADDRESS
1406 Ascot Lane
Blacksburg, VA 24060
(703) 953-0437 (h)
(703) 231-3501 (o)

PERMANENT ADDRESS
512-6 Dukpung Dong, Hanam Shi
Kyungki Do, Korea
(0347) 792-2414

EDUCATION

Ph.D.	*VPI & SU* (Virginia Tech), Blacksburg, Virginia. Department of Industrial and Systems Engineering Concentration: Management Systems Engineering Sep. '86 - Jan. '91
M.S.	*Seoul National University*, Seoul, Korea. Department of Industrial Engineering Concentration: Operations Research Mar. '81 - Feb. '83
B.S.	*Seoul National University*, Seoul, Korea. Department of Industrial Engineering Mar. '77 - Feb. '81

PROFESSIONAL EXPERIENCE

Mar. '87 - present
Management Systems Laboratories, Virginia Tech.
Graduate Project Assistant

Mar. '83 - Aug. '86
Korean Institute for Defense Analyses, Korea.
Researcher, Department of Management Systems Design and Development.
(Aug. '83 - Jan. '84 Army Officer Training Course)

PUBLICATIONS

Lee, Kwang S., Kurstedt, Harold A., & Middleman, Lou (1989). A Framework Indicates Direction in Future Intelligent System Development. *Proceedings of the 25th Annual Meeting of the Southeastern Chapter of TIMS*, 303-306.

Kurstedt, H. A., Lee, K. S., Mendes, P. M., & Jones, M. R. (1988). The Responsive System Operationalizes the Management System Model Concept Using Distributed AI Techniques. *Proceedings of the Conference on the Impact of Artificial Intelligence on Business and Industry*, 63-73.

Kurstedt, H. A., Lee, K. S., & Mendes, P. M. (1988). The Responsive System: A New Challenge for AI. *Proceedings of the First International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, 177-184.

Kurstedt, H. A., Mendes, P. M., & Lee, K. S. (1988). Engineering Analog in Management. *Proceedings of the Ninth Annual Conference of the American Society for Engineering Management*, 197-202.

PERSONAL DATA

1. Date of Birth:	Jan. 21, 1959
2. Gender:	Male
3. Family Status:	Married, Two Sons.
4. Country of Citizenship:	Korea