N92-22339

DESIGN FOR INTERACTION BETWEEN HUMANS AND INTELLIGENT SYSTEMS DURING REAL-TIME FAULT MANAGEMENT

Jane T. Malin
Johnson Space Center
Houston, TX 77058
malin @ aio. jsc. nasa. gov.

Debra L. Schreckenghost
TheMITRECorporation
1120 NASA Road One
Houston, TX 77058
schreck @ aio. jsc. nasa. gov.

Carroll G.Thronesbery
The MITRE Corporation
1120 NASA Road One
Houston, TX 77058
tbery @ aio. jsc. nasa. gov.

ABSTRACT

Initial results are reported from interdisciplinary projects to provide guidance and assistance for designers of intelligent systems and their human interfaces. The objective is to achieve more effective human-computer interaction (HCI) for real-time fault management support systems. Studies of the development of intelligent fault management systems within NASA have resulted in a new perspective of the user. If the user is viewed as one of the subsystems in a heterogeneous, distributed system, system design becomes the design of a flexible architecture for accomplishing system tasks with both human and computer agents. HCI requirements and design should be distinguished from user interface (displays and controls) requirements and design. Effective HCI design for multi-agent systems requires explicit identification of activities and information that support coordination and communication between agents. We characterize the effects of HCI design on overall system design and identify approaches to addressing HCI requirements in system design. Our results include definition of (1) guidance based on information-level requirements analysis of HCI, (2) high-level requirements for a design methodology that integrates the HCI perspective into system design, and (3) requirements for embedding HCI design tools into intelligent system development environments.

INTRODUCTION

Two multi-year, interdisciplinary projects are currently in progress to develop technology that helps developers of intelligent systems with real-time fault management capabilities achieve more effective human-computer interaction (HCI) design. The objective of the work is to specify requirements for methods and tools and to provide design guidance to assist developers in making intelligent systems better team players. Intelligent systems can be viewed as computer agents that share the task of process monitoring and control with human agents. An important part of designing the intelligent system is to specify how these human and computer agents will interact and what information they will exchange to be an effective fault management team.

Mitchell (1987) has observed that functional requirements for complex systems rarely specify information needed to support tasks of human controllers or operators. Yet, real-time interactive process control software usually requires that the needed information be easily accessible and discernible. This is especially true when this software includes intelligent systems to provide advanced support for human operators. Intelligent systems can provide additional information at a higher level of abstraction (e.g., reasoning about faults in addition to fault signatures in numeric data), but can also increase the need to coordinate and to monitor the software. Lack of integration of HCI information requirements into the functional requirements specification is an important problem in the design of real-time interactive process control software. It is even more critical when such software includes an intelligent system component.

However, user interface (UI) and HCI analysis and design has traditionally been viewed as relevant not to the functional requirements, but rather to the non-functional constraints (Roman, 1985), which are often applied later in software development. As Marshall (1991) phrases it, there is a common perception that human factors specialists "should be brought in to sprinkle magic dust on the interface or workstation once it is

largely developed." But deferring analysis of all types of HCI needs until late in system development contributes to lack of system support for the operator tasks. HCI considerations, however, can affect what the intelligent system does (i.e., its functionality) as well as constrain how the intelligent system is implemented.

The problem is due partly to a confusion between HCI requirements analysis and UI design. Human-computer interaction requirements refer to the information exchange between the user and the computer when performing the task which the software system is intended to support. The user interface design, on the other hand, concerns the display and control software and hardware that are the media for information presentation and dialog with the user. One of the dangers of confusing HCI and UI is the tendency to overlook task-level information needs (e.g., information needed to make a good decision), while focusing on physical interface characteristics (e.g., icon shapes and colors) and low-level interface functions (e.g., radio buttons and pull-down menus). This danger is described in detail by Woods and Eastman (1989) while introducing their "levels" of display design approach.

Specification of HCI requirements is critical in design of real-time, interactive intelligent systems. Focusing on screen display and low-level interaction sequences can lead to ignoring the more global issues of whether the right information is flowing between the software system and the user at the right time. Therefore, it is important to clarify the UI/HCI distinction and tackle the problem of including consideration of HCI information exchange requirements in development methods and tools.

METHODS

In one of the two projects, intelligent fault management systems within NASA have been evaluated for insight into the design of systems with complex HCI (Malin et al., 1991). Fifteen intelligent systems were selected from a variety of aerospace programs, including Space Shuttle, Space Station, unmanned spacecraft, and military and advanced aircraft. Information was collected by interviewing intelligent system developers and users, observing intelligent system demonstrations, and referencing available documentation about applications. In one case, participation in the design of the Space Shuttle Payload Deployment and Retrieval System (PDRS) Decision Support System (DESSY) project permitted first-hand observation of the design process and provided an opportunity to test design recommendations and develop examples of design guidance.

The other project has focused on conceptual design and prototyping of methods and tools for development of intelligent systems and their user interfaces. A prototype software toolkit, User-Intelligent System Interface Toolkit (UISIT), has been developed to support a methodology for developing conceptual designs for interactive intelligent systems and their user interfaces. An evaluation of UISIT prototype tools and methods is now being conducted in an application case. UISIT-based methods and tools are being used to support the PDRS DESSY project, to develop a prototype intelligent fault management support system. The purpose of this effort is to evaluate the underlying HCI development methodology and its support by the tools.

RESULTS

Preliminary results of this work indicate that the concept of information requirements is the productive focus for identifying needed modifications and additions to traditional methods, guidelines and tools. Information requirements identify the information which must be exchanged between the user and the intelligent system in order to support user tasks. Specifically addressing information exchange requirements helps the designer to elaborate what capabilities the intelligent system and the UI will need to have before determining exactly how to implement these capabilities. As such, these are an important subset of the external interface requirements specification. Information requirements also help to coordinate the design of the user interface and the application software.

Our solutions to the problems outlined above can be partitioned into the following areas, which outline what it will take elevate HCI requirements to the level of functional requirements: 1) HCI Design Guidance for Information Requirements, 2) Development Methodology for Information Requirements, 3) User Interface Tools and Methods for Information Requirements.

1. HCI Design Guidance for Information Requirements

Current forms of guidance should be extended to support the development of information exchange requirements.

Traditional guidance focuses primarily on visual appearance and style, offering little assistance in designing an intelligent system and its user interface which provides the right information to support user task performance. The example in Figures 1 and 2 illustrates this new guidance and its emphasis on information exchanged between the human and the computer. This guidance assists designers in determining the information which is required from the intelligent system for the display. It is relevant to decisions made early in the development process that constrain the information available for display. Unlike decisions about how to display information, they can not be deferred until later in the development process. This design guidance can help the HCI designer be more effective in identifying requirements for both the intelligent system and the user interface. Early involvement of the HCI designer in intelligent system development helps to integrate the efforts of the HCI designer and the intelligent system designer.

2. Development Methodology for Information Requirements

Another approach to elevating the consideration of HCI information exchange requirements is to provide an intelligent system development methodology that incorporates HCI considerations as an integral part of design. Such a methodology features the explicit specification of information requirements, including application of the new guidance identified above.

Initial research efforts (Johns, 1990; Malin et al., 1991) have resulted in a new perspective of the user -- the user as another type of agent in a heterogeneous, cooperating, distributed system. System design then becomes the design of an architecture for accomplishing domain tasks with the available human (i.e., users) and computer (i.e., applications) agents. HCI considerations are an important part of such system design, even before user interface design is addressed. We have defined the following stages in developing such a system (illustrated in Figure 3):

- *Description of domain tasks.* Monitoring and fault management tasks, whether performed by humans or software, are described in terms of goals and the actions required to achieve these goals. These goals and actions need not yet be assigned to specific types of agents.

- *Identification of resources provided to perform tasks and the constraints that affect task performance.* Resources include the capabilities of the operational environment and of the agents, and the availability of information. Constraints result from complexity, dynamics, and deficiencies in these resources.

- *Specification of agent activities and valid agent behaviors in an architecture for multi-tasking and dynamic task allocation.* Actions defined in the task description are assigned as specific agent activities consistent with the available resources and the inherent constraints of the fault management team and the operational environment. The task description is not complete until it includes activities supporting both domain task performance and coordination between agents of the fault management team. Note the implications for modification of task analysis and task description techniques to capture human-computer coordination activities.

- *Evaluation of the activity specification using expected operational scenarios.* The specification of agent activities is evaluated using complex, realistic activity sequences and modes of agent interaction to derive the information and functionality requirements. These scenarios should include tasks shared by multiple agents supporting both agent coordination and control of the monitored process, they should include both nominal and anomalous activity sequences, and they should highlight information used and exchanged to support each task.

- *Analysis of information exchange requirements.* Requirements for information are explicitly identified for both the intelligent system and the user interface, based on results of the evaluation using operational scenarios.

3. User Interface Tools and Methods for Information Requirements

A third way to promote the consideration of HCI information exchange requirements involves user interface tools and methods. User interface tools should support:

- Information requirements development
- Design team communication
- Run-time software module communication

User interface tools should support information requirements development and use as well as user interface design and management. Designing the HCI so that the end user always has the right

EXAMPLE OF HCI DESIGN GUIDANCE

Topic: Intervention Into Intelligent System Reasoning Process

Problem:

An intelligent system can become "disoriented" and require the operator to redirect it onto a more productive path of investigation (e.g., the system investigating an unproductive hypothesis or failing to investigate a likely hypothesis).

Recommendation:

Operator intervention into the reasoning process of the intelligent system can be used to manage errors in the intelligent system. Methods of intervention include modification of the reasoning process or selection of an alternate reasoning mechanism.

Supporting Information:

Intervention into intelligent system processing is one approach for redirecting a "disoriented" intelligent system onto a more productive path of investigation. The intelligent system must be designed, however, to permit such intervention. To effectively intervene in processing, the operator must first have a good understanding of the reasoning strategy used by the intelligent system.

Example Illustrating Problem:

See the following page for an example that illustrates the problem.

Example Illustrating Recommendation:

See the following page for an example that illustrates the recommendation.

Techniques:

Methods of intervening in the method of processing information include modification of the reasoning process or selection of an alternate reasoning mechanism. Examples of modification of the reasoning process are (1) setting processing priorities (e.g., what hypothesis to investigate first), (2) alteration of hypotheses, and (3) specification of alternate solutions.

Cross References:

The user interface used to intervene should reinforce the operator's understanding of the reasoning process to assist in identifying appropriate action. See section 4.1.2 for a discussion of presenting information about intelligent system reasoning. See the discussion of risk in intervening in the intelligent system later in this section.

Research Issues:

Making intelligent systems easily interruptable and redirectable, developing vocabularies for communicating advice about control decisions which reduce the amount of knowledge required by human team members about intelligent system internals.

Figure 1. Example of HCI Design Guidance, Page 1.

EXAMPLE OF HCI GUIDANCE

Topic: Intervention Into Intelligent System Reasoning

AGENT INTERACTION ILLUSTRATING PROBLEM

In this case, the operator is unable to communicate information about a likely fault and its correction procedure to the intelligent system. Unaware of this information, the intelligent system would continue to pursue fault isolation on an incorrect set of suspected faults, even when the actual fault was corrected. The operator would have to do a restart to reset the knowledge base and "fix" the intelligent system. Figure illustrates the agent interaction.

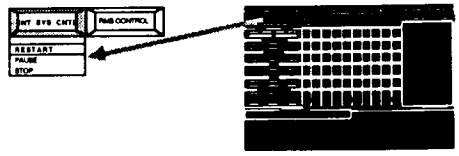

Illustration: Operator Restarting Intelligent System

AGENT INTERACTION ILLUSTRATING RECOMMENDATION

In this case, the ability to affect reasoning by altering hypotheses used by the intelligent system has been provided to the operator. The operator "informs" the intelligent system about the fault, performs a corrective procedure, and normal processing to continues. Figure illustrates the agent interaction.

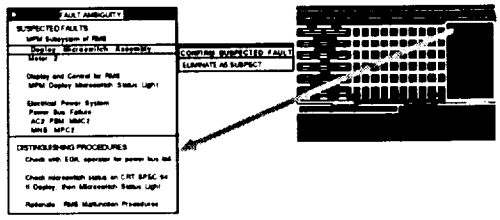

Illustration: Operator Informing Intelligent System of a Fault

Figure 2. Example of HCI Design Guidance, Page 2.

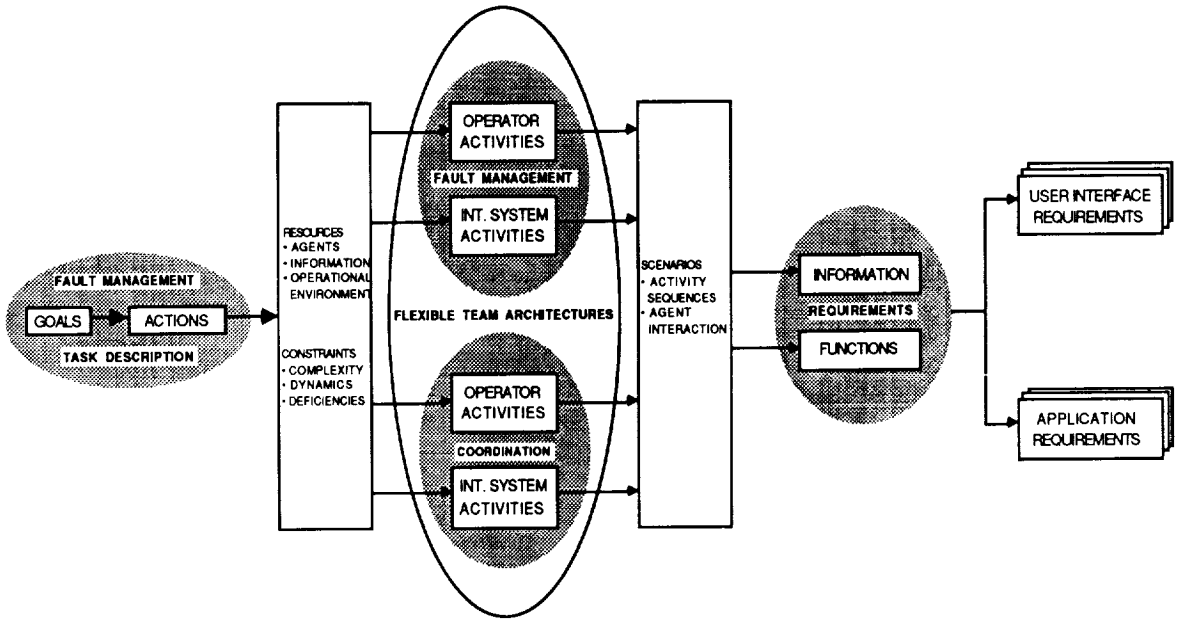

Figure 3. Recommended Requirements Development Methodology.

information at the right time during task performance is difficult, and it requires different tools from those which support user interface design. Tools should support the representation of information exchange items and the interrelationships among these items.

User interface tools should also use information exchange requirements for coordination and communication among design team members. This allows early and continued consideration of the information exchanged between the user interface and the applications components.

The preliminary design methodology developed in conjunction with the UISIT prototype toolkit is illustrated in Figure 4. It defines information requirements early in the system design process and supports coordination among design team members. Analysis and preliminary design occur first to derive high-level descriptions of intelligent system functions, tasks, architecture, and HCI. From these descriptions, requirements are derived for the major software component functions and for HCI information exchange. Using information requirements, the design team can be partitioned into smaller groups, with each group performing a separate but coordinated and concurrent design of a software component. As it is discovered that requirements must change to match constraints, the information requirements can serve as a point of coordination among the design team members. For instance, if the intelligent system designer discovers that additional information from the user is required to complete an activity, the information exchange specification serves as a single point of coordination with the user interface designer. The intelligent system designer and user interface designer can prototype their software modules separately while using the information exchange specification to coordinate with one another.

User interface tools and methods should support user interface prototyping. An especially effective form of prototyping support is to represent information requirements in the same module which supports run-time information exchange. This allows the separate prototyping of the user interface and the applications modules. It also requires the information exchange specification to be stated unambiguously.

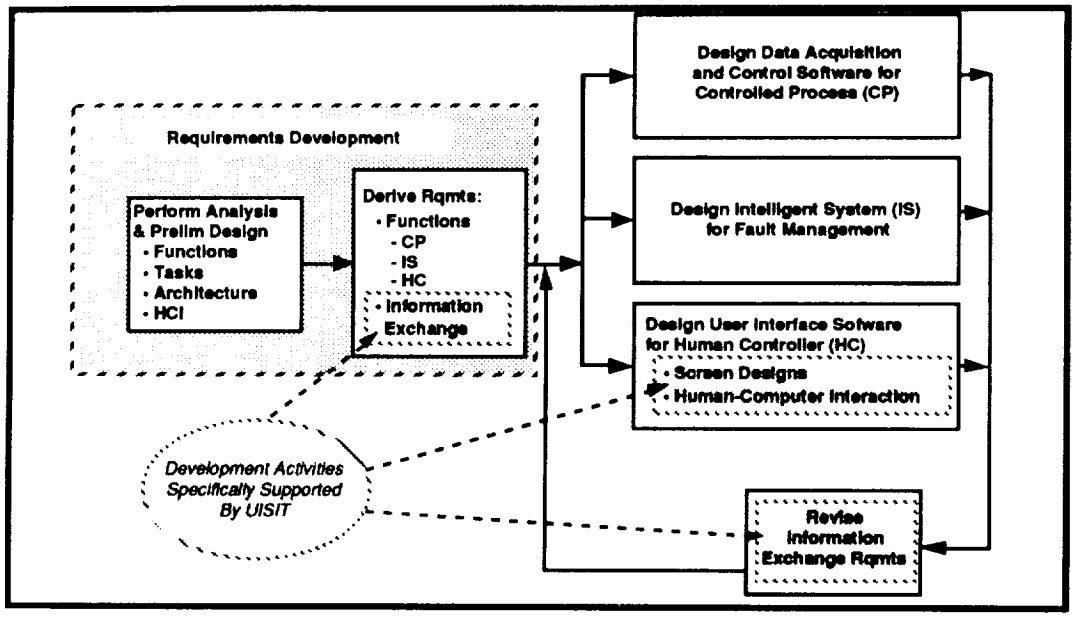

Figure 4. Recommended UISIT Methodology.

Finally, it presents a design team member with an external cue to coordinate changes with the remainder of the design team.

Figure 5 illustrates how a software system built with the aid of UISIT segregates the information exchange specification from the specification of the intelligent system and the data acquisition software. This segregation allows each specialist to apply unique expertise to a single software component, while providing a means for coordinating design team efforts through the information specification module. When it is discovered that a change needs to be made to the information specification module, it is clear to all involved that the change needs to be coordinated with the other members of the development team. In this way, the information specification module serves as an abstraction of what each module designer needs to know about the design of the other modules.

Figure 6 shows an object hierarchy comprising the information specification component of a sample system built for UISIT evaluation.

The contents of the object hierarchy represent the information requirements and their relationships for PDRS DESSY. The object layer was constructed within the UISIT framework, in accordance to the suggested UISIT hierarchy. Because it is encoded as an object layer, supporting run-time communications between the user interface software and the applications software, the information specification module is unlikely to be misinterpreted by members of the development team

RELATED WORK

In 1983, Norman outlined four strategies for improving to human-computer interaction design. Our recommendations for enhancing these strategies are summarized below:

- *Help the designer to maintain an awareness of the user's needs.* Rather than simply trying "to impress upon the designer the seriousness of the matter," we propose that it would be more effective to assist in identifying user needs by elevating analysis of information exchange

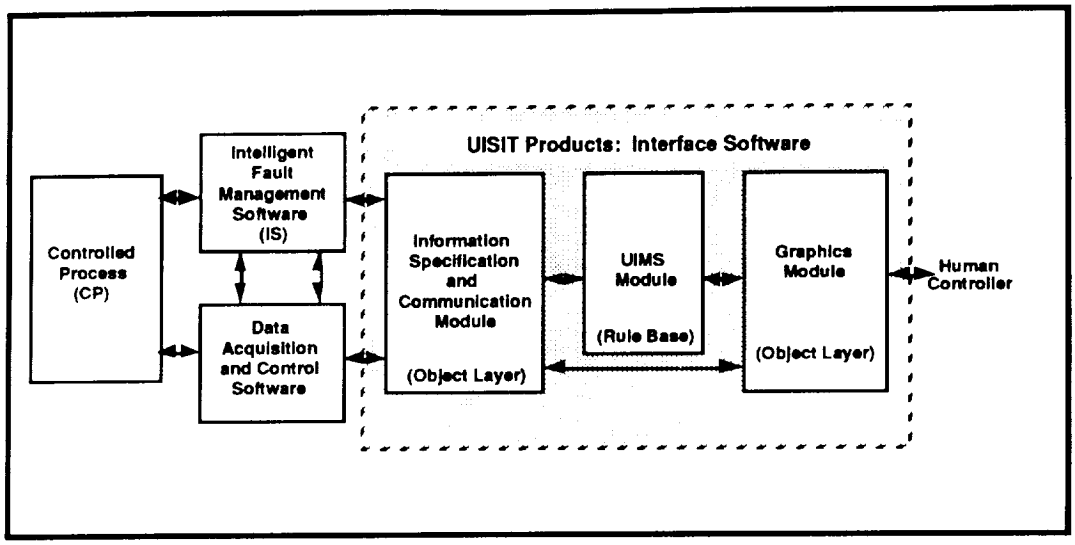

Figure 5. UISIT-Supported Architecture.

between the software and the user to a functional requirements level.

- *Provide the designer with methods and guidelines.* Most assistance has concentrated on support for evaluation of user interface designs. We shift the focus to support for analysis of information needed to support user tasks.

- *Provide software tools for interface design.* As Norman suggests, this is a natural follow-on to providing methods and guidelines. The software tools can provide guidance in two ways: (1) explicitly in the form of on-line guidelines, templates, and examples; and (2) implicitly by the design of the toolkit, which requires specific architectural components and a specific sequence of developmental steps. In either case, such tools make it easier for the designer to follow the prescribed methods and guidelines.

- *Separate the interface design from other programming tasks.* While Norman's original suggestion was based on such principles as software reusability and interface modifiability, we have found that this practice can also support design team communication.

While we have been emphasizing HCI development in our own work, we do not intend to imply that user interface issues are unimportant. We emphasize HCI development because it is too often neglected. Both HCI and UI challenges are important, and both are necessary for effective system design. Without effective HCI design, the UI designer may not have a clear idea of the information exchange goals he is trying to optimize, and the resulting software may not support the user's task performance. Without effective UI design, the user interface may not convey the information effectively, thereby detracting from the support of the user's task performance. Since both types of issues are necessary for effective design and since HCI issues are often overlooked, we have concentrated our work on HCI issues. There are significant UI design challenges in monitor and control of complex processes, due to the large quantities of real-time information, accompanied by deficiencies of the quality and availability of this information.

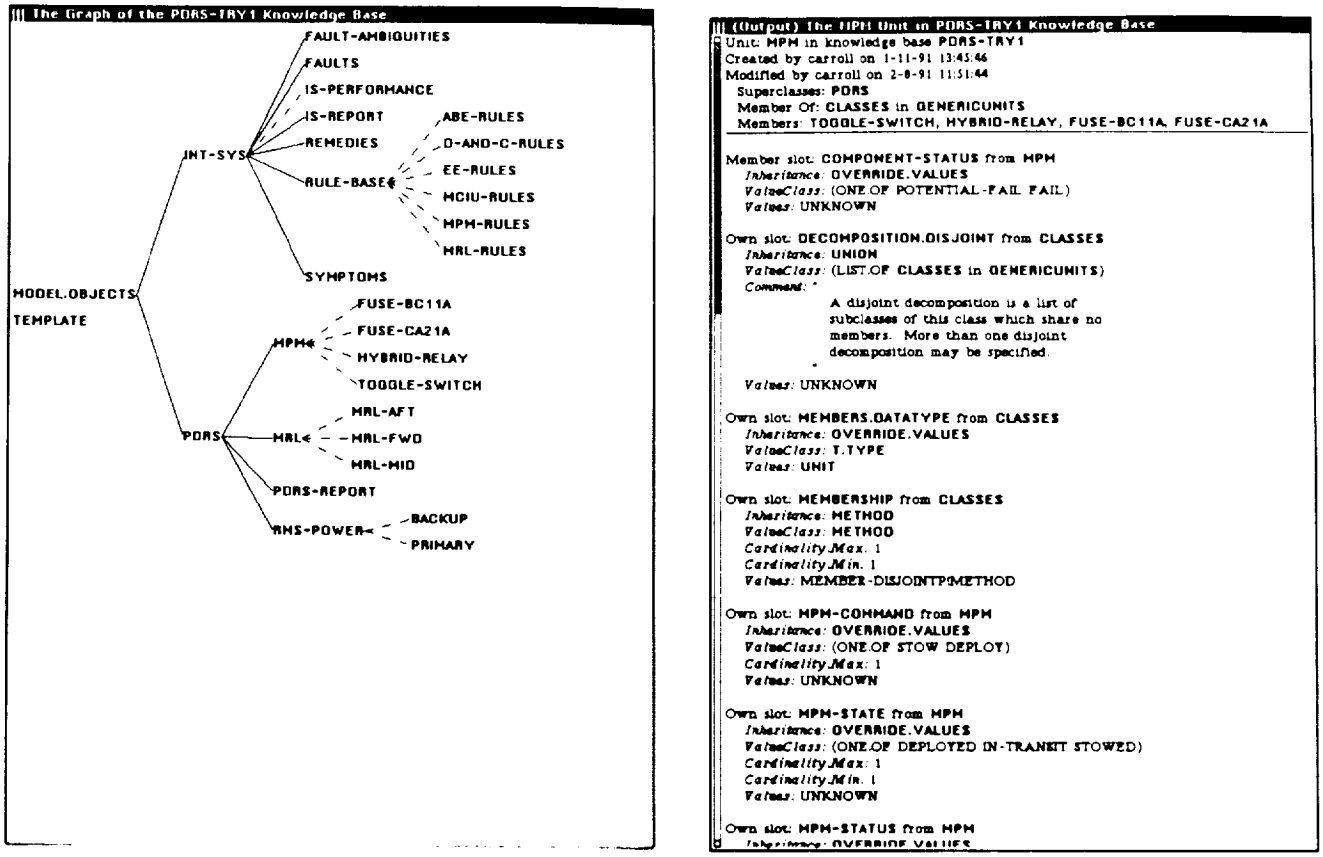

Figure 6. Object Hierarchy and Sample Definition for UISIT Communication Layer.

Because the conclusions of this report are based on the study of intelligent systems for real-time fault management, the reader may be concerned about how well they generalize to other types of software systems, especially in view of Leveson's (1990) warning that applying common data processing approaches to process control systems can lead to disaster. We believe that our results for HCI design guidance are applicable to *complex* software systems. Complex software systems are characterized by (1) large amounts of information from multiple sources, (2) sophisticated software capability, often with multiple tasks performed concurrently or jointly by human and computer, (3) time-constrained processing with deficiencies in information quality and availability, and (4) active information exchange between human and computer. Our results on development methodology and tools appear to be generalizable to a broader set of systems, in particular systems where information exchange between human and computer represents a significant aspect of using the software.

SUMMARY

This paper has introduced an on-going research project to improve human-computer interaction for fault management intelligent systems. This issue is being investigated by means of a case study, by participation in intelligent system design, and by prototyping methods and tools. Preliminary results indicate that elevating HCI information exchange requirements to the level of software functional requirements is critical to designing software which supports user task performance. The following is a summary of the findings from this research effort:

1. HCI Design Guidance for Information Requirements

- User interface design guidance, which focuses on visual appearance and style of information presentation, should be extended HCI, to also assist designers in developing information requirements.
- HCI design guidance should be integrated within a development methodology that supports use of these guidelines.

2. Development Methodology for Information Requirements

- Task analysis and task description techniques should be modified to identify human-intelligent system coordination activities and to support identification of information requirements.
- Developers should adopt a methodology that makes guidance easier to use and integral to the development process from the early stages of analysis and design.
- Mechanisms for communication of information requirements and coordination of design activities among members of the development team should be provided with the methodology.

3. User Interface Tools and Methods for Information Requirements

- User interface tools should support information requirements development and should include explicit representation of information items and the interrelationships among them.
- User interface tools should provide information requirements as a point of coordination and communication among members of development team.
- User interface tools should represent information requirements in the same module that supports run-time information exchange.

REFERENCES

Johns, G. (1990, September). *Graphic Interfaces to Intelligent Fault Management Systems: Issues and Guidelines* (Report No. MTR-90W00103). McLean, VA: The MITRE Corporation.

Leveson, N. G. (1990, November). "The Challenge of Building Process-Control Software." *IEEE Software, 7*(6), 55-62.

Marshall, C. (1991, March). "Ergonomics Is Dead: Long Live Ergonomics." *Human Factors Society Bulletin, 34* (3), 4-6.

Malin, J. T., Schreckenghost, D. L., Woods, D. D., Potter, S. S., Johannesen, L., Holloway, M., & Forbus, K. D. (1991, July). *Making Intelligent Systems Team Players: Case Studies and Design Issues, Volume 1. Human-Computer Interaction Design* (NASA Technical Memorandum). Houston, TX: NASA-Johnson Space Center.

Mitchell, C. M. (1987). "GT-MSOCC: A Domain for Research on Human-Computer Interaction and Decision Aiding in Supervisory Control Systems." *IEEE Transactions on Systems, Man, and Cybernetics, 17*(4), 553-572.

Norman, D. A. (1983, December). "Design Principles for Human-Computer Interfaces." *CHI '83 Proceedings*. New York: ACM, 1-10.

Roman, G. C. (1985, April). "A Taxonomy of Current Issues in Requirements Engineering," *IEEE Computer, 18*(4), 14-21.

Woods, D. D., & Eastman, M. C. (1989). "Integrating Principles for Human-Computer Interaction into the Design Process: Heterarchically Organized HCI Principles." In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, IEEE.