MODELING, IDENTIFICATION AND CONTROL, 1986, VOL. 7, NO. 3, 129–144
doi:10.4173/mic.1986.3.2

Man–machine interface design for modeling and simulation software

ARNSTEIN J. BORSTAD†

Keywords: *computer aided design, interactive software, man–machine interaction, software prototyping, modeling, simulation.*

Computer aided design (CAD) systems, or more generally interactive software, are today being developed for various application areas like VLSI-design, mechanical structure design, avionics design, cartographic design, architectual design, office automation, publishing, etc. Such tools are becoming more and more important in order to be productive and to be able to design quality products. One important part of CAD-software development is the man-machine interface (MMI) design.

A more non-typical CAD application, which is currently addressed by Statoil's R&D Division, is modeling and simulation. The use of simulation as a tool for studying complex dynamical systems has become increasingly important during the last decade. Several systems have been developed in the past to accomplish this task. Unfortunately, these systems are not particularly 'user-friendly'. To be able to apply modeling and simulation effectively as a design tool, the quality of the user interface of the simulation systems has to be improved.

In this paper the importance of a careful man–machine interface (MMI) design in CAD software is stressed. A general characterization of the MMI issue is given, along with some general MMI design principles. The prototyping approach to MMI specification, which involves user participation, is advocated. Finally, some details on a prototype developed at Statoil on MMI for advanced simulation systems are described.

1. Introduction

Today's computer systems are not very good at communicating with their users. They often fail to understand what their users want them to do and they are unable to explain the nature of the misunderstanding to the user. In fact, it is the common experience of users of interactive systems, whether novice or experienced, infrequent or regular, that communication with their machine is a time-consuming and frustrating experience. (Foley *et al.* 1984, Kantowitz *et al.* 1983, Sime *et al.* 1983.)

One important contributing factor to this man–machine communication barrier, is that the man–machine interaction problem is often underestimated among computer system developers. Many of the principles applied to MMI design may seem obvious. However, when the time comes to design a user interface, the principles are all too often forgotten. Presentations, procedures, terminology, etc, that are obvious and clear to the computer specialist, can be difficult and even impossible to understand for the user (Fig. 1). The computer expert is easily fascinated by sophistication in programming tools, and the MMI could, from the user's point of view, be negatively influenced by this appreciation. Moreover, there is often pressure to get the system running as soon as possible, which leaves insufficient time for a thoughtful design to be developed. Thus, the user interface is consequently designed for ease of implementation rather than ease of use.

Received 11 February 1986.
† Statoil A/S, Research & Development, N-7000 Trondheim, Norway.

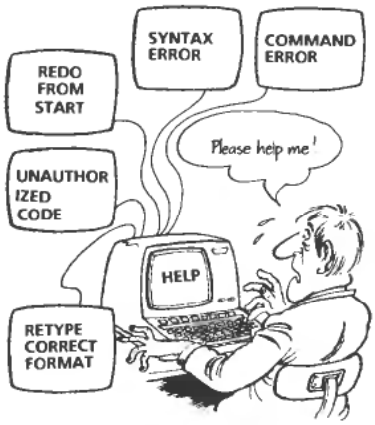


Figure 1. The Man–machine communication barrier. (Gaines *et al.* 1984.)

One way to enforce the development of quality MMI is to pay more systematic attention to MMI in the specification phase of a computer system project, as opposed to adding the MMI after the system has been developed.

2. General characterization of MMI

Traditionally, not only the development of computer systems, but also the use of computer systems, has been restricted to computer systems' specialists. Today, this is not so. Computer systems have become an important tool in various application areas, and it must be possible to interact effectively with such systems, without being a computer expert. Thus, the design of man–machine interfaces has become an important issue in the design of computer systems. (Gaines *et al.* 1984.)

A majority of the computer systems developed in the past tend to have a sequential- or batch-oriented structure. A set of program modules is accessed in a sequential and predetermined order. One easily ends up with this structure when the system development is focused on the computational or internal software itself, and its performance. The inputting and outputting formats are added at the end, because they have to be there. No efficient dialogue with the computer system is possible with this structure, since the information flows go in one direction. We say that the user has only serial access to the system.

In order to achieve a quality MMI, a more dialogue- or interactive-oriented structure is preferable. (Gates 1984.) Here, the user has the initiative, and can interact with the computer system on his own terms. The computer system becomes an effective tool where the user has access to the system in a variety of ways, dependent on what the user needs. Hence, the repertoire of program modules can be accessed in the order chosen by the user. We say that the user has orthogonal access to the system. To be able to successfully design a computer system with orthogonal user access, considerable emphasis has to be put on the MMI design at an early stage of the development process. (Alavi 1984.)

Our objective is to design quality MMI. To be able to do that professionally, some attributes have to be associated with the term 'quality'. There are currently no standards by which to measure the quality of MMI. In the MMI-literature the term 'user-friendliness' is frequently used as a quality attribute. The term entered the language of computer advertising before it entered the language of computer

engineering. User-friendliness certainly gives us a flavour of what quality of MMI means, but the term is too imprecise for our purposes. We need a more formal definition of quality.

The term man–machine-interaction (MMI) can be defined as information or data exchange between a human and a computer system and its software. The information exchange, or rather the dialogue between man and machine, conceptually consists of two parts. One is the formalization (input) part, and the other the presentation (output) part. The quality of the user interface is reflected by the degree of coherence in the information exchange or the dialogue.

A dialogue is said to be coherent when,

(i) it is possible for the user to formalize his input in a way which is understood and used correctly by the machine

(ii) the presentation gives the user the best possible understanding of the current state of the system

A definition based on psychology and physiology (Foley *et al.* 1984) can also provide some insight into the problem of designing quality user interfaces, as indicated in Fig. 2. Interaction with a computer involves three types of human processes or activities: perception, cognition and interaction. Perception is the process for receiving and transmitting information from the computer to the human brain. Cognition is the process by which the user assigns meanings to the perceived information, and makes decisions based on those and previously accumulated information already stored in his mind. The interaction comes into play when the user, having received, recognized and decided how to respond to the presented information, performs an input action. A quality user interface minimizes the time spent on perception, cognition and interaction. To design quality user interfaces is not a trivial matter. In the design process it could be useful to think of interpersonal communication as an analogy, because there are various desirable attributes of interpersonal conversation and language which should be preserved in a user-computer dialogue. Another useful analogy if you are designing a graphical screen interface, is to characterize how documents and tools, or more generally, how various pieces of information are arranged and manipulated on a person's desk.

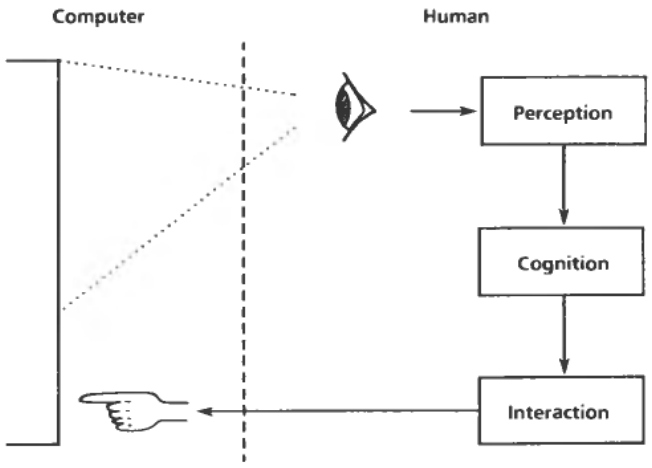


Figure 2. The MMI human processes.

To summarize, a quality user interface is one through which the user carries out the intended work with minimum conscious attention to his 'tools' and maximum task effectiveness.

3. General MMI design principles

General MMI design principles, which are independent of the application, can be extracted from relevant literature on the subject. The literature on MMI is vast, but diverse, fragmented and multidisciplinary. (Burch 1984.) Since computer technology is rapidly changing, literature from recent years is of major relevance. The guidelines embedded in the literature have been derived from a combination of human factors and ergonomics research and experimentation, and on practical MMI-design experience from different projects. However, a considerable part of the MMI designs reported seems to be based on 'ad-hoc' reasoning and 'common sense' thinking, and seems to lack a scientific basis. Nevertheless, a set of MMI design principles should be established for the project. These principles should be kept as a framework for the design. A set of such principles is given below (Foley *et al.* 1984, Vershel 1984):

Consistency

A man–machine interface is consistent when the display and the inputting formats are uniform and lack exceptions and special conditions. The purpose of consistency is to allow the user to generalize knowledge about one aspect of the user interface to other aspects. Consistency reduces the user's need for memorization and the possibilities for misunderstandings.

Context-preserving interface

A man–machine interface is context-preserving when the context in which the user operates is reflected in the interface design, i.e. the state of operation of the system is apparent from the actual information available to the user. A context-preserving dialogue helps to maintain clarity in the user's working session, reducing the need for memorization, and increasing the user's efficiency. As a consequence of this, a computer display should never be blank, but always indicate what is going on.

Application-oriented terminology

The user interface terminology should be based on the user application and not on specialized terminology related to software development disciplines, which is irrelevant to the user. Obviously, a computer system is most easily operated when the computer is able to speak the language of the user.

Revelant information presentation

The dialogue or the information presented to the user should always be relevant to what the user's needs actually are, i.e. the user should not be distracted by occurrences that are not related to the current state of operation, or by internal program changes or whatever it might be that appears irrelevant to the user.

Natural grouping of information

A man–machine interface is conceptually an organization of the exchange of a larger set of more or less related pieces of information between the user and the computer software. Obviously, the different pieces of information should be grouped or ordered in a manner which appears natural and logical to the user.

Flexibility

To be able to operate the system effectively, it should be possible for the user to invoke the various parts of the computer system in a non-sequential and non-predetermined order, i.e. it should not be necessary to go through sequences of operations which are irrelevant to the user's particular choice of operation.

Feedback

The computer system should give an immediate feedback in response to the user inputting action, i.e. if the input has been accepted, by reporting the effect of the action, and indicating the state of operation. Ideally, the user should have the feeling that he is conversing with the machine.

Robustness

The man–machine interface should be robust in the sense that it should not 'crash', i.e. become involved in endless computer loops, fail, produce nonsense, or get wound up in the system software, as a result of an incorrect input. Moreover, the computer should not erase a whole sequence of valid results if the user makes an error.

'User-skill'-adaptive

The man–machine interface should adapt to the user's skill level, allowing more options and choices for the experienced user, and more automatic features for the novice or casual users. In this way the range of users can fully realize its capability within the framework of the system.

User-controlled HELP-facility

The user interface should be equipped with some on-line and user-controlled HELP-facility, which the user can invoke if he needs any assistance. It should be possible to use the HELP-facility concurrently with other operations, without losing data or leaving the current environment or context. It should also be possible for the user to choose the level of detail in the HELP-info.

System developers are likely to claim that the principles listed here are obvious. Nevertheless, when it comes to MMI design they are often forgotten. One way to ensure that they are not forgotten, is to ask continuously during the MMI prototyping if the current design satisfies the principles.

4. MMI prototyping fundamentals

Detailed design guidelines are difficult to develop without actually trying out different concepts of the application involving people like those who will be using the system. Hence, during the specification phase of a computer system project, a software prototype for the user interface should be developed and experimented with.

Prototyping is a well-established method of designing complex hardware systems. It is also becoming more common in software engineering these days, as a supplement to traditional software development. (Adamski 1985, Alavi 1985, Ramamoorthy *et al.* 1984.) Prototyping means building or designing with a new technology or for a new application while the feasibility of the design is questioned. Therefore, prototyping is basically a feasibility study that aims at demonstrating system aspects which are critical to the user. One such important aspect is the user interface. The traditional approach to software development permits little feedback from the users until the coding stage, i.e. very late in the development project. Prototyping, however, is a practical way to cultivate and achieve user participation and commitment to the project from its earliest stage.

MMI prototyping seems to add an extra dimension to the creative process of MMI design, since it appears easier to explore and try out different concepts. MMI prototyping is an iterative and incremental activity where important aspects of the user interface are experimented with. The prototype usually undergoes various experimental stages, with user participation and feedback at each stage, which are taken into consideration in the next and improved version of the prototype. To obtain representative feedback, user participation should include a number of persons and different categories of users, since human reactions and preferences are diverse and not easily predictable. The different stages in the prototyping process are illustrated in Fig. 3.

During the design the prototype should be checked against general MMI design principles established for the project. The idea is that the MMI prototype should evolve into a basis for a quality MMI design of the application at hand. The prototype does not need to be implemented with the same hardware and software as the final system. On the contrary, it could be beneficial to choose some special hardware and/or software to speed up prototyping productivity. The ultimate objective is to be able to extract a detailed and complete MMI specification as a result of the prototyping process.

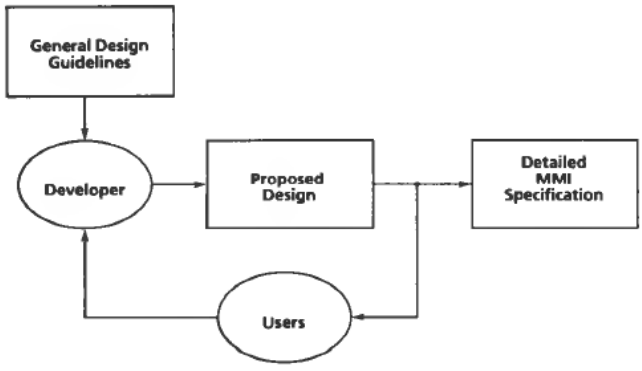


Figure 3. The prototyping process.

5. CAD in modeling and simulation

5.1. Perspective on simulation software

The use of simulation as a tool for studying complex dynamical systems has become increasingly important during the last decade. (Spriet *et al.* 1982, Tyssø 1985.) Several software systems have been designed to accomplish this task. (Rimvall *et al.* 1985.) Unfortunately, these systems are not particularly 'user-friendly'. Most of them tend to have a batch-oriented structure where the user has to access a set of separate program modules in a sequential or predetermined order, in order to complete a simulation study, as illustrated in Fig. 4, i.e. separate programs for modeling, simulation and analysis of results, and where intermediate data are stored on files. Tools which include data-assisted documentation and reporting are scarcely available, which means that the user has to do a lot of manual work to produce a simulation report. Each program module often has its own, specific user interface, which means that the user has to learn how to operate a set of programs, rather than one program.

To be able to apply modeling and simulation effectively as a design tool, the quality of the user interface of simulation systems has to be improved. (King *et al.* 1984, Symons 1985, Tyssø 1985.) In an advanced simulation system all the typical simulation activities: modeling, data specification, presentation, analysis and reporting, should be integrated via one common user interface, as indicated in Fig. 5. The user interface should be user-driven, interactive and flexible, in such a way that the user can carry out the intended work with minimum conscious attention to the 'tools' and maximum task effectiveness.

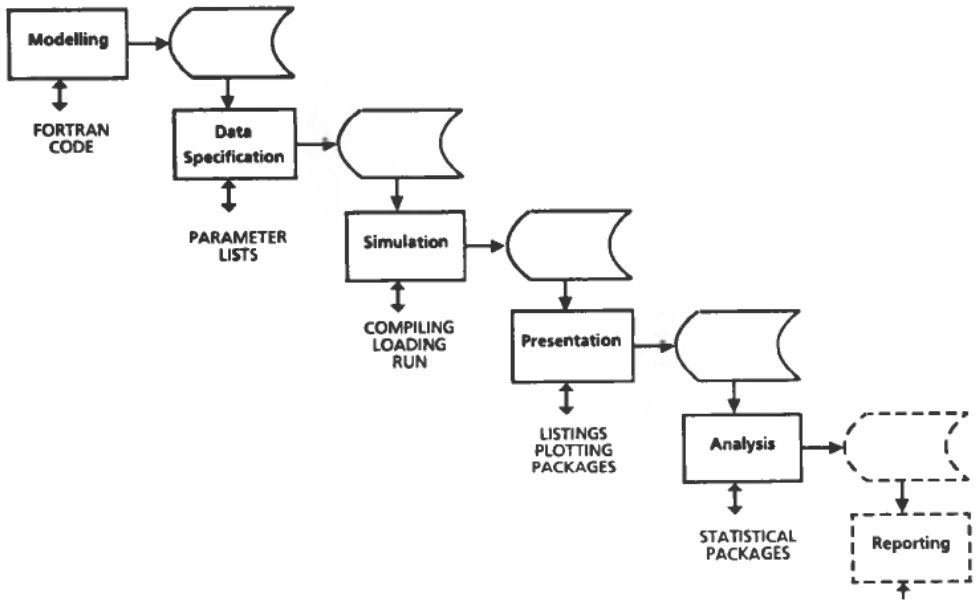


Figure 4. Typical structure of existing simulation software.

5.2. MMI-prototype developed at Statoil

Statoil's R&D division in cooperation with A/S Veritec and A/S Computas Expert Systems, has developed an MMI prototype for advanced simulation systems, where the MMI-design philosophy advocated in this paper has been adopted.

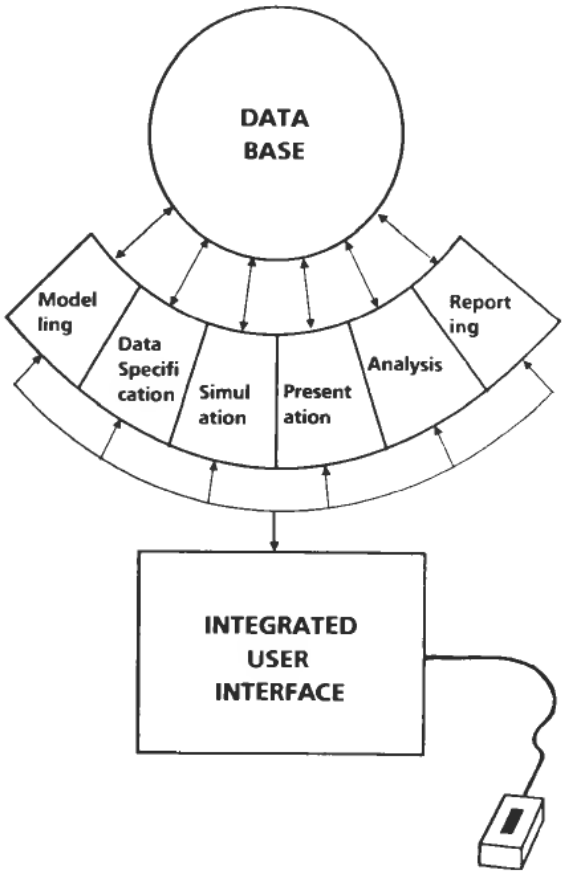


Figure 5. The integrated user interface of an advanced simulation system.

Emphasis has been put on the user interface itself, and minor effort has been put in to the background software. The purpose of developing the prototype has been twofold:

(*a*) To demonstrate today's possibilities with repect to quality user interface design

(*b*) An experimental basis for specifying a user interface of a simulation system to be developed.

The prototype has been developed on a Xerox 1108 workstation, a 16 bit computer with 1·5 Mb memory, 42 Mb rigid disk, a bit-mapped high resolution screen (1000 × 800 pixels), and a 'mouse' as the main interaction device. The basic programming language is Interlisp-D, a major dialect of the LISP language. In addition, a high level extension to Interlisp-D named LOOPS is used. LOOPS offers object-oriented, data-oriented and rulebased programming in addition to procedure-oriented programming which is available through Interlisp-D. Interlisp-D/LOOPS is integrated in an interactive development environment. The equipment is suited to experimental programming, i.e. rapid prototyping, since frequent structural changes of programs can be undertaken with minimum effort.

The MMI prototype is highly graphical-oriented with various objects coded as iconic symbols. The screen is organized with multiple windows, representing differ-

ent parts of the user interface, and this makes it possible for the user to interact with different aspects of the user interface simultaneously. The user interface is 'mouse'-driven, i.e. operations are selected and activated by pointing at icons or at menu selections.

5.3. Elements of the user interface

User activities with functions. The user interface is operated via a finite set of User Activities, which are accessed via a menu, Fig. 6(*a*). Each User Activity consists of a set of User Functions, which is implemented as a submenu, see Fig. 6(*b*) for the Modeling Functions menu. Both User Activities and their Functions are selected with the mouse. The menu selection currently being active is displayed on a black background. The User Activities are defined as follows.

Modeling is defined as the User Activity for building a model of linked components. The user can fetch a relevant set of components from a Component Library. Each component is symbolized with an icon, and the components are linked by drawing connections between the terminals of the component icons. The model is denoted the Active Model.

Data Specification is defined as the User Activity for specifying data to the component variables. The data are entered in a component data window. The complete set of data for all components in the Active Model, is denoted the Active Dataset.

Simulation is defined as the User Activity for simulating an Active Model, with an Active Dataset, over a specified time horizon, resulting in a set of Active Results. A user-selected subset of the Active Results can be displayed during simulation.

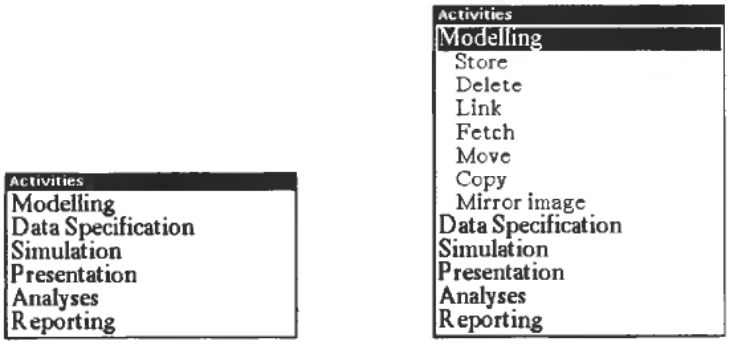


Figure 6. (*a*) User Activity menu. (*b*) Modeling Functions menu.

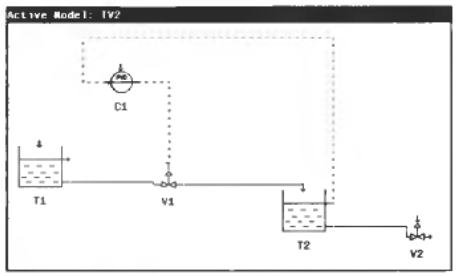
Presentation is defined as the User Activity for selecting a subset of Active Results to be displayed during simulation. Active Results can also be replayed here after simulation.

Analysis is defined as the User Activity for analysing Active Results. (Not implemented in the current version of the prototype).

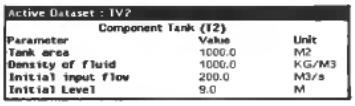
Reporting is defined as the User Activity for documentation and reporting of a simulation experiment. (Not implemented in the current version of the prototype).

Information objects. The User Activities with Functions operate on various Information Objects. The Information Objects are: Active Components, Active Model, Active Dataset, and Active Results, Fig. 7.

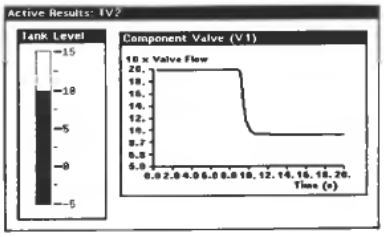
Active Components
A relevant set of components currently in use for modeling. The components are fetched from the Component Library by the user and displayed in a separate window labeled Active Components.



Active Model
The model currently under consideration. The model is represented as a flowsheet of linked components and is displayed in a separate window labeled Active Model. The visual part of the flowsheet can be changed by scrolling the window. Models can be stored and fetched from a Model Library.



Active Dataset
The dataset currently under consideration. Data is displayed in a separate window labeled Active Dataset for one component at a time. Datasets can be stored and fetched from a Dataset Library.



Active Results
The results generated by simulating the Active Model with the Active Dataset over a specified time horizon. A user-selected subset of the results is displayed in a separate window labeled Active Results. Results can be stored and fetched from the Results Library.

Figure 7. Information Objects.

General Functions. A set of General Functions has been included, which can be invoked independently of the context in which the user is working. These functions are, Help, Print, and Text, and are implemented as icons, which can be activated by the mouse, Fig. 8. The function currently active will be displayed on a black background.

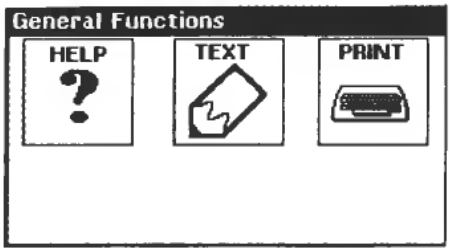


Figure 8. General Functions.

Help

By pointing at this icon the user will get a textual description of how to operate the system, 'detailed' or 'summary', displayed in a separate window labeled Help. This Help function comes in addition to the context-dependent Help-information presented in the Messages window.

Print

By pointing at this icon and a window on the screen, a hardcopy print of the chosen window will be generated. If the user points at the background of the screen, a hardcopy print of the total screen is generated.

Text

By pointing at this icon the user can write a text (comment or annotation), and place it anywhere in the Active Model window or the Active Results window.

Library Icons. Access to the four Libraries, Component Library, Model Library, Dataset Library and Results Library, are made via a set of Library Icons (Fig. 9), which are activated by using the mouse. A list of the content of a Library is shown in a window which opens when pointing at a particular Library Icon with the mouse. The user functions fetch, store and delete are applicable on all Libraries apart from the Component Library, which has read-only access via the fetch function. The content of the Libraries is organized in a hierarchical way, Fig. 10. A model can have several Datasets, and several Results can have been generated from a Model and a Dataset.

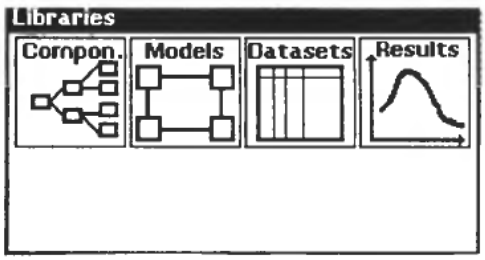


Figure 9. Library Icons.

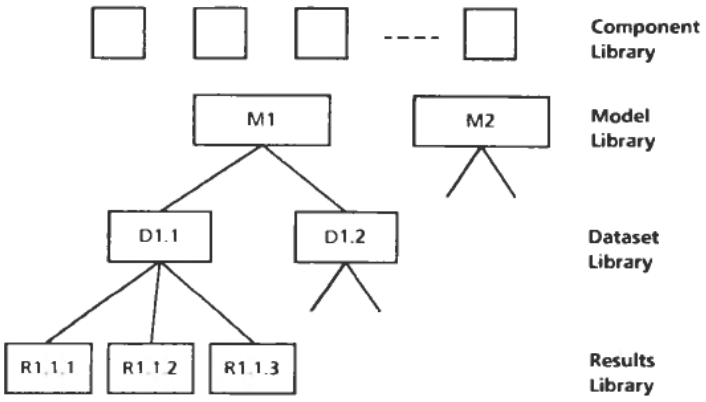
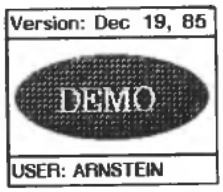


Figure 10. Organization of libraries.

Status Information

The user interface also includes various kinds of Status Information, which monitors the current use of the system, and hence has a context-preserving function. The Status Information includes: Login/Logoff-icons, System- and User Identification Icon, Clock Icon, Calendar Icon, Flowsheet Overview Window, and Messages Window, Fig. 11.

Login/Logoff Icons
The user logs in and out of the system by pointing at these arrows with the mouse.

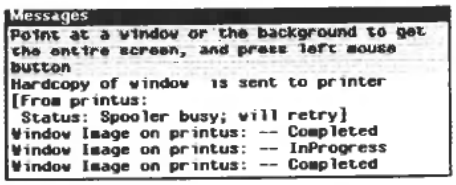


System-and User Identification Icon
Displays the system's name and the current version, and the user currently being logged on.

Clock Icon
Displays current time.

Calendar Icon
Displays current month and day.

Flowsheet Overview Window
Shows which part of the Active Model that is currently displayed in the Active Model window. The window is updated on invoking the scroll icons.



Messages Window
Displays mouse button help texts, and status messages from the system. Old messages are displayed on shaded background.

Figure 11. Status Information.

5.4 MMI conceptual model

The various user interface elements can be summarized in a conceptual model, as depicted in Fig. 12. The user interface is operated via a set of User Activities.

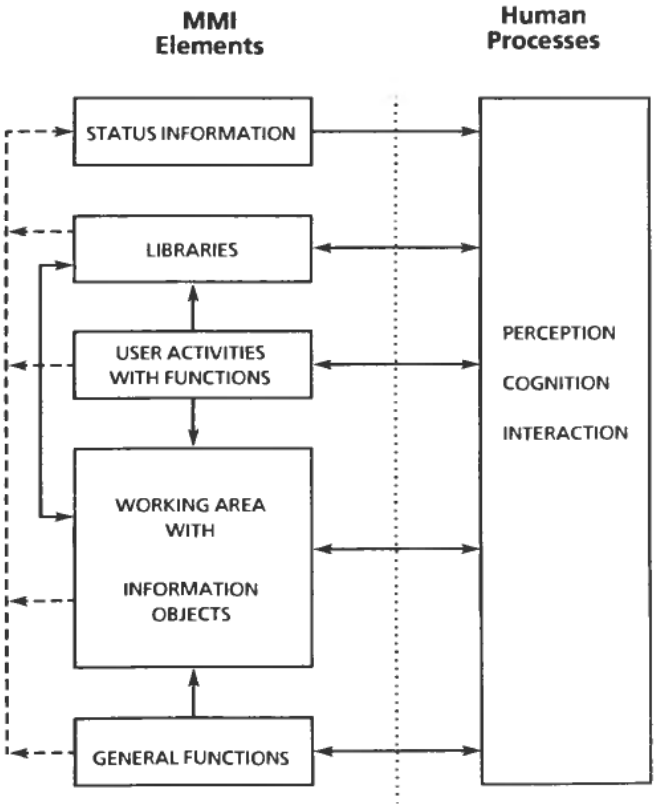


Figure 12. MMI conceptual model.

Each User Activity consists of a set of Functions. Various Information Objects can be inspected and interacted with by these Functions. The Information Objects can be stored in and retrieved from a set of Libraries. In addition to the User Activities with Functions, a set of General Functions, which are generally available, has been included, i.e. they do not belong to a particular user activity. To preserve the context of the operation of the user interface, a set of Status Information objects, which monitors the current use of the system, has been included. The user interface has been organized in a way which tends to minimize the human processes involved, i.e. perception, cognition and interaction.

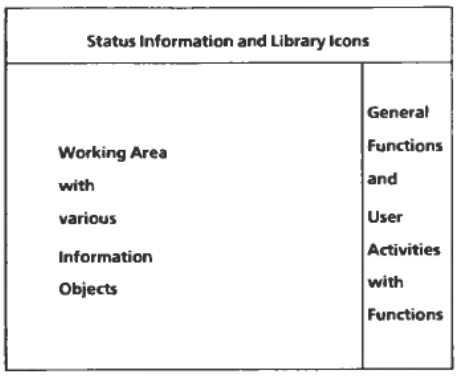


Figure 13. Principal screen organization.

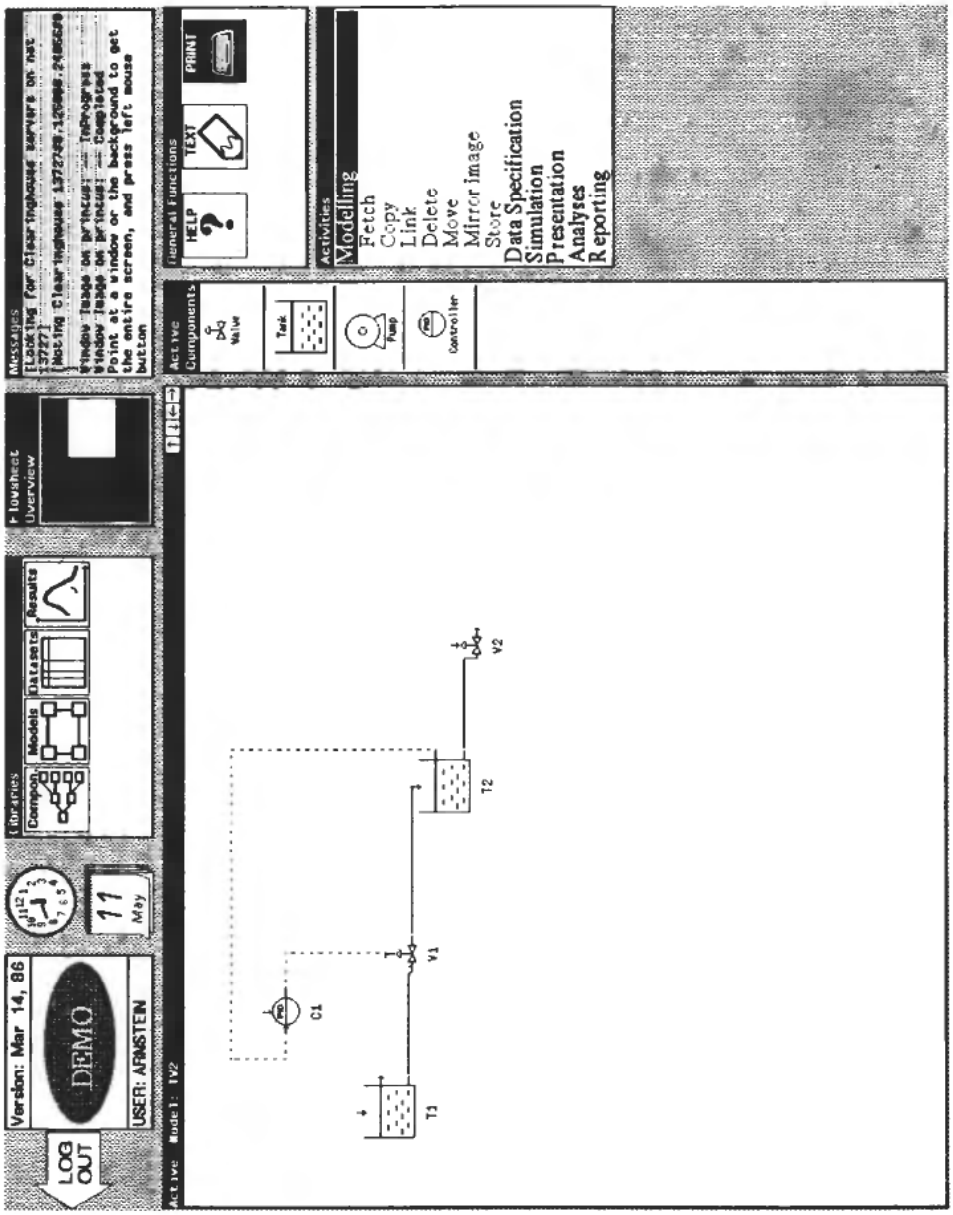


Figure 14. Detailed screen layout, Modeling.

The various information elements have been organized on the screen as indicated in Fig. 13. The Status Information and Library Icons are located at the upper part of the screen, General Functions and User Activities with Functions are at the rightmost part of the screen, whereas the various Information Objects cover the largest part of the screen which we call the working area. The content of the working area is dependent on the user activity currently being active. As an example, the detailed screen layout during Modeling is shown in Fig. 14.

6. Concluding remarks

An outcome of the current expansion in the use of computers, is the growing number of users without formal training in programming or computer technology. Such users wish simply to apply the computer as a useful tool in their daily work, and are not interested in becoming computer professionals or in understanding details of the application system. Today, poor usability in a myriad of forms is one of the main complaints of computer users. In order to design software systems which permit an effective relationship between people and computers, it is necessary to focus on the MMI design. Modelling and Simulation is a typical application where a lot can be achieved by enhancing the quality of the user interface. The main elements of the user interface for this application have been elaborated in this paper.

Acknowledgment

The preparation of the paper was supported by Statoil. The author is thankful to Magne Fjeld at Statoil, who suggested publication of the paper, and who contributed with valuable comments on the organization of the paper.

References

Adamski, L. (1985). The prototyping process, *Systems International*, June issue, Vol. **13,** No. 6, 91–92.

Alavi, M. (1984). An assessment of the prototyping approach to information systems development, *Communications of ACM*, June issue, Vol. **27,** No. 6, 556–563.

Burch, J. L. (1984). Computers: The Non-Technological (Human) Factors. A recommended reading list on computer ergonomics and user friendly design (The report Store).

Foley, J. D., and van Dam, A. (1984). *Fundamentals of Interactive Computer Graphics* (Addison-Wesley).

Gaines, B. R., and Shaw, M. L. G. (1984). The art of computer conversation (Prentice Hall).

Gates, B. (1984). The efficient user interface. *Systems International*, Nov issue, Vol. **12,** No. 11, 131–132.

Kantowitz, B. H., and Sorkin, R. D. (1983). *Human Factors: Understanding People–system Relationships* (John Wiley & Sons).

King, R. A., and Gray, J. O. (1984). A graphical man–machine interface for CAD and simulation of dynamic systems, Presented at the 6th European Conference on Electrotechniques EUROCON 84—Computers in Communications and Control.

Ramamoorthy, C. V., Prakash, A., Tsai, W.-T., and Usuda, Y. (1984). Software engineering, problems and perspectives, *IEEE Computer Magazine*, Oct issue, Vol. **17,** No. 10, 191–209.

Rimvall, m., and Cellier, F. (1985). Evolution and perspectives of simulto nuae oloigteCSSLsadad, *Modeig, Idenfcato otrol*, Vol. **6,** No. 4, 181–199

Sime, M. E., and Coombs, M. J. (1983). *Designing for Human–computer Communication* (Academic Press Inc.).

SPRIET, J. A., and VANSTEENKISTE, G. C. (1982). Computer-aided modelling and simulation (Academic Press Inc.

SYMONS, A. (1985). What is user friendliness in modelling and simulation? Proceedings of the Summer Computer Simulation Conference 1985. pp. 69–73.

TYSSØ, A. (1985). Simulation as a tool in operational safety, reliability and control, *Modelling, Identification and Control*, Vol. **17,** No. 13, 127–140.

VERSHEL, M. (1984). Designing user interfaces for engineering design systems, *Proc. Int. Conference on Computer-Aided Design, ICCAD-84*, pp. 218–220.