Introduction

Recently, due to improvements in workstation technology and in other related fields such as computer graphics and artificial intelligence, many studies and experiments have been performed on the subject of the man-to-machine interface. These studies have demonstrated that it is extremely important to provide users with intelligent and friendly front-ends to all classes of programs, including system programs or shells, in order to develop an optimally productive computing environment. In fact, a new field or science has emerged which spans a wide variety of related fields, including computer graphics, psychology and others. This field aims towards the study of providing specifications and concrete solutions for the creation of the most appropriate environments and interfaces for users of computers.

XWIB is an application in the field of man-to-machine interaction. It makes use of computer graphics and language manipulation techniques in order to build window-based interfaces for the generation of input to a wide variety of applications. The applications under consideration span a variety of numerical simulation techniques which require complex input data specification. In particular XWIB has been designed to support the applications within MOTECC, which in general require complex data input.

The input specification of a typical application in the class of numerical simulation programs requires an in-depth knowledge of a complex set of formats, variable types, and data structures. The various input parameters are usually logically grouped and organized in some hierarchical fashion, often with some dependencies between different parameters in different hierarchical levels. Usually these parameters are specified via either a line or screen editor by typing the necessary input data from a computer keyboard. This requires complete knowledge of the names and possible options for all the input parameters of interest, of the format of the input file, and of the the editor being used. In other words users are requested to be “experts” of each particular “system”; in this case system refers not only to the particular application or simulation program that is being used, but also to the operating system and file editor of the computer being used.

The initial acquisition of this type of knowledge can often prove to be a deterrent to new users trying to learn a new program, or to users trying to use a different computer system. One solution to alleviate these
difficulties is to develop a user-friendly interface based on a windowing environment consisting of hierarchical cascade menus and dialog windows, with each level in the cascade containing finer and finer levels of information. This hierarchical representation scheme, together with format checking and an automatic dependency handling mechanism would allow the user to concentrate entirely on the specification of the input data rather than on the details of the way to enter them, or the format of the file. This would also avoid the possibilities of making conceptual or syntactic mistakes in the compilation of the input file.

XWIB has been designed as a general purpose application that can be used to generate special purpose window-based interfaces for specific input representation and/or generation in different domains. In order to be generally applicable to a large body of domains, the program provides:

- a graphical representation of the input parameters that naturally reflects their characteristics and structure;
- simplicity of use and user-friendliness: all the relevant information for creating input can be available at the touch of a button;
- mechanisms for simple syntax checking of the input and to automatically ensure that the formats of the input data are in accordance with a format definition provided to the program;
- input generation guidance: the user is driven in the steps to be taken;
- dependency-handling mechanism among input parameters;
- help messages;
- on-line documentation;
- general utility functions: editing and system facilities.

The program has been developed on an IBM RT/PC workstation under the AIX operating system. It has been implemented in the C-language and it makes use of three C-language subroutine libraries. The first one, known as Xlib, is provided with the X Window System, and contains the X Window functions for handling the windowing part of the program. The second library contains the functions private to XWIB for handling the windowing representation scheme of the program. The third library contains functions for handling the input and output flows. In particular it includes a parser and an interpreter for a Dependency Language developed within XWIB, and a calculator for basic arithmetic operations. The program runs on Unix-based workstations supporting a C compiler and the X Window System.

1. Input Representation and Generation with XWIB

The main purpose of a user interface is to provide the users with friendly and easy-to-use instruments to interact with systems, databases or, more generally, programs. The purpose of XWIB is to provide an instrument for the creation of user interfaces that can be used as front-ends to programs and/or databases that require input representation, generation and modification. In the section below we describe the characteristics of an important class of applications in the domain of chemistry that have been studied for the purposes of XWIB.

1.1. An Example in the Domain of Numerical Simulation Programs

As an example of a problem that can be solved with XWIB we will describe the characteristics of numerical simulation programs with respect to the problem of input data generation.

The first characteristic of the input required by numerical simulation programs concerns the amount and complexity of input parameters. These programs usually require a huge amount of input parameters and values which can have many different types of formats and data structures. These include varied formats of type integer, floating, alphanumeric and logical and different types of unidimensional and multidimensional data structures.
The second important characteristic that is observed is that most of the input data for simulation packages have an intrinsic hierarchical structure. An example is the "NAMLIST" data type that is one of the most common methods for data input, mainly because it belongs to the IBM Fortran language, which is in fact one of the most common programming languages in scientific computing. Thinking in terms of a graph-like notation, the namelist represents the highest node in the graph, the different variables within the namelist the second level nodes, and the values of the namelist variables the third level nodes. Each of the third level nodes are connected to a second level node by a single arc in the graph, while the second level nodes are subsequently connected to the first level node again by a single arc. A similar organization can be seen in a more complex data type called "datagroups" or groups of formatted Fortran cards. In this case the first node in the graph represents the datagroup, while the second level nodes are represented by the formatted data cards, which may contain multiple data elements. The third level in the graph is then the individual data elements, while the fourth level is represented by the values of these data elements. Figure 1 shows an example of this type of hierarchical data representation.

![Diagram](image)

Figure 1. Example of a hierarchical data representation in a graph-like notation. The connection between different levels is always with a single arc.

In addition to the hierarchical structure present in the namelist and datagroups, another higher level appears may sometimes be present on top of the previous ones. This level is represented by the nature of the data input, and is a result of the fact that particular computations require specific groupings of data (usually particular combinations of namelists and datagroups). This is a much more subtle hierarchy, depending for example on the type of computation to be performed. For example, in using KGMOI for computations of interaction energies between pairs of molecules it is advisable to compute these energies including a correction for the Basis Set Superposition Error (BSSE). This is one of the options that can automatically be invoked by the KGMOI program. However, when this option is used, a particular combination of
namelist data and datagroups inputs is needed in the input file, and they must be ordered in such a way that they follow the path of the computation. This "data ordering" is actually another level in the data hierarchy, imposed by the particular computation being performed. If another type of computation is to be used, then this organizational hierarchy may not be present, or may be replaced by another organization.

In essence, what this means is that depending upon particular values of input parameters it is often necessary to combine blocks of data in different forms; the different forms depending upon the namelists and datagroups that are needed by the computation to be performed. This type of structure can be viewed simply as another level in the data hierarchy, and can be easily represented graphically, even if it is true that this is not always easily recognizable by the application developers, or users themselves. Figure 2 shows an example of one hierarchical data organization for a type of computation that can be performed with the KGNMOI program.

This last aspect is one strong advantage to have a graphically oriented interface for input specification to large numerical application programs. For example, even though traditional program manuals are well written and programs are well documented, it is often the case that particular relationships between input parameters are not explicitly cited in the manuals, or may not be easily recognizable from the description of the input. It may also be the case that different and varied input parameters are so numerous that the users are not aware of all the possible ways to combine them, or that they are not aware of all the necessary data structures needed for particular types of computations.

Therefore, an interface for handling these types of input specification should ideally be designed to provide an environment that graphically depicts the representation of the data structure, and also drives the user in the choice of the steps to be taken each time. This type of information can be clearly represented using a window-based input scheme, through the use of hierarchical cascade menus and dialog windows, with each level in the cascade containing a successively finer level of information.

---

2. A Windows-Based Environment

In this section, after a brief introduction to the X Window system, we will describe the windowing capabilities offered by XWIB.

2.1. The X Window System

X Window is a "network-transparent" and "vendor-independent" operating environment which supports multiple overlapping windows for color and monochrome displays on a variety of vendor workstation platforms. Using a metaphor, we may say that a typical X Window environment appears to the user as a desk with some sheets of paper on top of it. Each sheet of paper contains a certain amount of information, and as one turns the pages on the desk the information on the pages can be manipulated and modified as desired.

However, instead of the desk and sheets of paper, X Window provides a graphics screen and "windows". The windows can be viewed as small screens that are displayed on the graphics display, and which can overlap, become invisible and visible again, can accept input and display output, can be popped up and down by means of a mouse (typically menu windows) and finally can display icons or images. Moreover, different applications or programs may run at the same time, making use of any number of windows, so that a user can perform several actions at once by simply moving the cursor from one window to another.
Figure 2. In this example, a MOLSCF computation from KGNMOL has been selected. This requires the specification of the data for INT1&2 and SCF, as indicated in the menu entitled MOLSCF in the lower left corner. By selecting INT1&2, the user is informed that data for &INTIMP, COORDINATES and BASIS are needed. &INTIMP is a namelist, containing parameters specifying some of the characteristics for the integrals computation. In this example the Z matrix format for inputting the coordinates is being set to false.
Figure 3. An example of the windowing environment for the quantum mechanical program HONDO is shown. The user has selected the option to specify the RUN TYPE computation. The RUN TYPE in HONDO is specified by a primary namelist called &CNTRL and another set of namelists which varies depending upon the value chosen for the parameter RUNFLG in &CNTRL. The figure shows the menus in which the user will be allowed to enter in the first step of the input specification (all the ones that open from &CNTRL menu). Once a value for the parameter RUNFLG has been selected the user will be allowed to proceed in the specification of the related namelists. As the parameter RUNFLG can take just one value at the time, the access to the entries of the menu RUN TYPE will be in mutual exclusion too.