A Flexible GUI Design System

S. D. Mullerworth
The Meteorological Office
Bracknell, UK

A Flexible GUI Design System

S.D.Mullerworth
The Meteorological Office
Bracknell, RG12 2SZ
UK
sdmullerworth@meto.gov.uk

The Generic Hierarchical User Interface (GHUI) is a design package for creating forms-based user interfaces and was written at the UK Meteorological Office. The intention is that the package is simple enough, that a basic, working interface can be written quickly by developers with little experience of the package, but that later, the interface can be enhanced by incorporating application specific code.

Generic functions provided by the GHUI include a client/server database system that allows users to save and copy work, a hierarchical tree widget for navigating with ease among a large number of windows, a flexible function for processing users' input into a text format that is suitable for controlling the related application, a function for creating uncomplicated input windows containing standard text, entry box, button and table widgets, and an error checking facility to prevent input of invalid responses. Application specific functions can be added to provide, for example, additional input validation checking or non-standard input windows.

A GHUI-based user-interface is specified by a set of text control files with a simple syntax. For example, each input panel, such as the one shown in the figure overleaf, is specified by a list of instructions in a single control file. Amongst other control files is a register that contains declarations for each of the data items set by the user, and a set of template files that define how the input to the application is to be converted to a format suitable for running the related application.

An important design aim was to ensure that each of the types of control file was easy to understand and therefore easy to write. The constraints that this design aim placed on the format of control files had an additional advantage in that it was then possible to write additional functions that reread the same control files for different purposes. For example, rather than using TclTk commands, input panel control files are written in a higher level GHUI language. Such an approach has enabled a number of useful generic functions to be written that base their output on these files; for example, a function that creates a text description of a user's settings. Such generic functions would be more difficult to implement if the use of TclTk commands to enhance the appearance of input panels was allowed.

The GHUI was written to create the Unified Model User Interface (UMUI). The Unified Model (UM) to which the UMUI interfaces is a very large, flexible modelling package used by the Met. Office both for its wide-ranging forecast products and for its climate prediction programmes. To offer the full flexibility of the UM to users, the UMUI requires some two hundred separate input windows. While professional looking applications have been developed using only the generic functions provided by the GHUI system, the UMUI takes full advantage of the opportunities for incorporating application specific code.

Much of the application specific code in the UMUI relates to the validation of user input. Each item in the database can have one of a standard set of checks applied to it, for example, to constrain the range of a numerical input. Alternatively, a specially written validation routine can be specified which enables complex conditions and cross-checks with other input to be applied. Tcl is an effective language for creating the required short scripts.

Another more substantial piece of application specific TclTk code provides the UMUI with a non-standard input window design in an area where the basic nature of the standard GHUI input panel was unsuitable. Although all the application specific code in the UMUI has been written in TclTk, clearly it would be possible to use other languages or incorporate input panels created by other GUI application builders.

All extensions are incorporated into the application in almost the same way as the standard GHUI functions; generally by listing the name of the function in the appropriate control file. Thus, the fact that

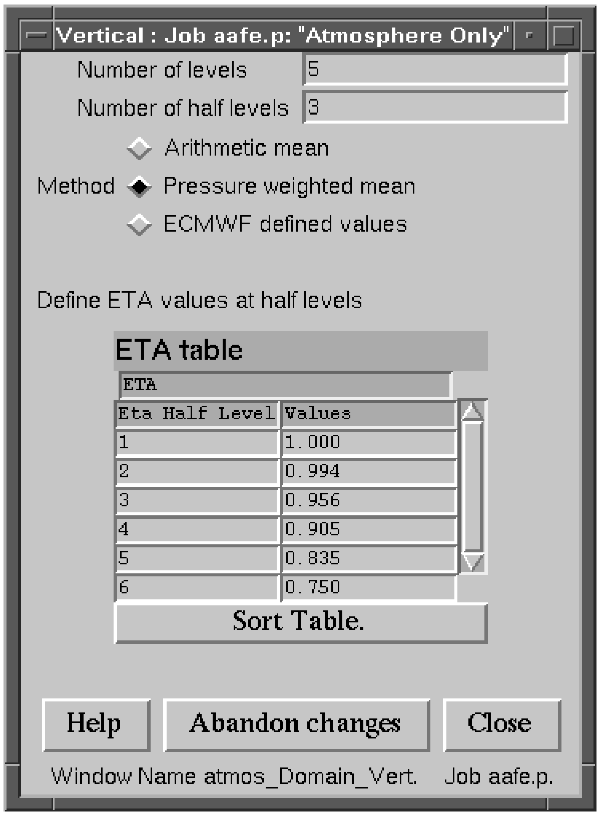

Figure 1: A simple example input panel from the UMUI showing entry boxes, a set of radiobuttons and a table widget. The three buttons at the bottom of the window are common to all panels. 'Help' brings up local help for the window, 'Close' and 'Abandon' close the window with or without saving changes.

a function is application specific is transparent to the user, yet such functions are relatively uncoupled from the GHUI system and they are thus unlikely to be affected by GHUI system upgrades.

The decision to design the GHUI rather than use an existing GUI application builder was made in part because of the need to use public domain software as it was intended that the UMUI was to be distributed free. Use of text control files has brought some other significant advantages when compared with many GUI application builders. Firstly, alterations can be done quickly with a text editor and no compilation or rebuilding process is required. Secondly, the files are readable by humans which is useful, for example, when searching for a particular question on one of 200 panels; simply search the input window control files for a particular string. Thirdly, again with regard to the number of windows required, the text format is much less cumbersome than the C code or resource files generated by many GUI packages.

Our experience demonstrates the feasibility of maintaining a suite of GHUI-based applications. Incorporation of application specific code to overcome the restrictions of the generic functions has not caused significant problems. The GHUI can provide the basis for user interfaces to a whole class of scientific packages. The GHUI approach, of designing an application to provide basic core functionality which can be enhanced with application specific code, could be applied to other classes of application where common requirements can be identified.