Carnegie Mellon University
Software Engineering Institute

An Application Developer's View of AUIS

June 1994

Software Engineering Institute
Carnegie Mellon University
Pittsburgh PA 15213

Sponsored by the U.S. Department of Defense

Notes:

- **The talk discusses use of AUIS from the point of view of a long-time tool developer currently trying to build tools to support the development of a large application.**
- **AUIS is the developer's second foray into the world of graphic interfaces and close to being the first foray into the world of object oriented programming.**

Overview

Fundamental concepts of our tool set

Reasons for choosing AUIS

Features of AUIS

Hindsight

Notes:

- **The main purpose of the talk is to discuss our experience with AUIS. We use AUIS as the user interface portion of a tool set and to understand our use some knowledge of the tool set is necessary.**
- **The features of AUIS that are discussed are those that have either helped or hindered us in our development.**
- **The big question: Would we use AUIS again?**

Outline of Tool Set

Constructing an environment to support structural modeling.

Notes:

- **We are applying structural modeling to the development of a real-time flight simulator.**
- **One characteristic is that there are many instances of very few types of software element.**
- **Each software element has a fixed structure (it may change from simulator to simulator) and may be represented by a database schema.**
- **The user interface is the sole source of interaction with the data either by editing it directly or by causing other programs to manipulate the database.**
- **We have many windows running from a single application.**

Editable Objects

Structured documents

- **Text**
- **Equations**
- **Tables**
- **Pictures?**
- **Figures?**

Representations of simulator structure

Ada code

Notes:

- **The editable objects are stored in pieces in the database and are constructed for purposes of display. After the user has edited the object it is parsed into appropriate pieces and stored back in the database.**
- **Essentially, documents are text with some insets for tables and equations. We do need capability for read only text.**
- **The simulator structural representation is based on org.**

Reasons for Choosing AUIS?

Local support is available

Appeared to have the right features for our needs

Looked better than other systems

Curiosity

Believed that it could be linked with our existing database system

Notes:

- **Fred is a good advocate.**
- **Good to support locally produced systems.**
- **Source availability means that modifications can be made if necessary.**
- **Looked at other free systems such as InterViews and a whole bunch of others. We had not looked at tcl/tk.**
- **As well as a query language, the database has a C language function call interface.**

Useful Features

Data streams

Insets

Interesting classes

Smart styles

Notes:

- **Streams allow us to separate cleanly the database from the interface. They provide a simple data representation without worrying about how to display the object.**
- **Insets give a really good, seamless way to display objects and allow the user to use appropriate editors for each object rather than one universal set of editors.**
- **Smart styles are dynamite! They make a great deal possible when editing text.**

Not So Useful Features

Dynamic loading

Menus

Documentation

Implementation

Notes:

- **Dynamic loading makes debugging near impossible. We use AUIS 6.2 on a Sparc2 with SunOS 4.1.2 and gdb 4.12 and cannot debug dynamically loaded objects.**
- **Menu manipulation is very hard. A lot of the time we need to change the menus to suit our own needs and it has been hard!**
- **There is not enough helpful documentation — of varying quality, not all complete, documentation of data streams would have been nice.**
- **Not all classes are implemented as well as they could be! There are some really good ideas that aren't carried through (read only styles). It is hard to find where the source is!**

Comments

Dynamic loading:
- we don't need it
- use statically linked objects
- shared libraries?

Menus:
- would like an implementation that is easier to control
- lots of new views

Documentation:
- standards
- automatic generation from class header files?

Implementation:
- need more solidly engineered product

Notes:

- Technically, dynamic loading is a nice idea, but from our viewpoint brings on problems that outweigh any gains. Perhaps the concept should be investigated to see just how much use is made of it.
- Menus proved to be an example of a class that is clearly necessary, but very hard to use. Thought into use of the classes would be helpful. Menus might be more convenient if a portion were under the control of the application and another portion from the data object.
- The documentation isn't always complete. The implementation doesn't always come up to the promises of the documentation.
- Concepts need to be fully implemented — consider the ReadOnly style.

Hindsight

AUIS has caused a number of headaches

Hasn't always lived up to its promise

We would use it again

It wouldn't link with the database

AUIS is a great idea, just not the best implemented

You need a good programmer to develop new applications

Notes:

- **AUIS is probably still the best interface out there for our needs.**
- **It would be nicer if it were easier to construct a control program with better menu support.**
- **We intend to continue using it for our current application and would use it again for similar applications.**
- **The database problem may be as simple as collision between function names — time hasn't provided the opportunity to investigate.**