

UniVista

Interfaces for Scientific Modeling Codes

Edward Peterlin

April 7 1999

The Problem

- **scientific modeling codes are beginning to be shared much more extensively than before**
 - budgetary constraints
 - inter-lab research cooperation
- **most scientific modeling codes are Fortran namelist based**
 - input is a long text file of names and values
 - no documentation
- **frequently no one aside from the author of the program can help explain how to use it and what it does**
- **difficult to properly model the research usage patterns of modeling codes with a naive design**

Previous Work

- **Amber (PPPL 1996)**
 - modeling codes are essentially a data management problem
- **UniVista I (PPPL 1997)**
 - same objectives as UniVista
 - partial demo, then project leader quit and project was abandoned and code lost
- **PEST (Manickam 1998-9)**
 - GUI for a modeling code written in IDL
 - similar graphical layout as UniVista: variables are associated with UI input elements on the screen
 - problems
 - interface does not work for any modeling code aside from PEST
 - user can't adapt the interface to match their research patterns

The UniVista Approach

- **create a flexible interface creation tool for any environment and modeling code**
- **provide tools for three different types/needs of users**
 - **modeling code author**
 - **should be easy to use to give incentive to create a GUI**
 - **research designer**
 - **provide a way to customize the interface to fit the style of research they want to do with a modeling code**
 - **research executor**
 - **provide an easy way to run the modeling code varying elements of a chosen research pattern**
- **allow the interface to mimic research patterns**
 - **research builds off of previous research**

The UniVista Design

- **flexibility**
 - achieved through proper data management with a database
 - organizes and collects interfaces
 - archives results for future reference and extensions
- **portability through Java and Swing**
 - scientific community is quite diverse utilizing a variety of machines
 - by using Java 1.1 and Swing 1.0 as a foundation, UniVista will run on all initial machines at the PPPL (WinNT, Solaris, MacOS) as well as for other labs
 - Swing offers the possibility of having an identical look on each platform at the same time as flexibility to use a native 'look and feel'

The UniVista Design

- **distributed architecture**
 - there can be any number of researchers on a project
 - researchers may not be in the same laboratory
 - all of UniVista information is stored in standard relational databases, including all of the information to recreate a graphical user interface, interfacing through JDBC
 - built in database security provides robust methods for sharing and protecting sensitive data

The UniVista Design

- **support for research methodology**
 - interesting results frequently lead to further research on what caused those results
 - UniVista studies (the user interfaces to viewing and changing values of the variables for a modeling code) are flexible
 - created off of descriptions of the modeling code
 - can have unique code to check whether variables satisfy parameters for a research area
 - can be cloned off of a past run so researchers can take a snapshot of the conditions that led to specific results and quickly build an interface to allow them to do further directed study

Results

- **the design aspect of UniVista is complete, and the idea of GUIs being accepted for modeling codes has been shown through the PEST project**
- **development is along the path to having a complete working model of the design**
- **the final UniVista working model will be tested by creating a user interface for Degas, an ion flow modeling code from the PPPL**

Conclusions

- **the proper user interface design tool for scientific modeling codes needs to be adaptive enough to respond to the particular work patterns of researchers**
- **the most powerful user interfaces are those which can be reconfigured to match the way the user wishes to use them**
- **the UniVista design provides UI flexibility and (through Java) flexibility to function in most any scientific computing environment**