Visual Hierarchical Task Analysis Software with Imbedded KLM

Dave Bockus, Ryan Wilson
Brock University
Department of Computer Science
500 Glenridge Ave.
St.Catharines, Ontario Canada
L2S 3A1
Tel: 905-688-5550 x3281
bockusd@brocku.ca

ABSTRACT
Hierarchical Task Analysis (HTA) [1] is used to describe the practice of a software system. When combined with a KeyStroke Level Model (KLM) [2, 5] one can determine comparative task efficiency. Software was developed to allow the graphical representation of systems as a series of tasks which are decomposed into elemental components of operation (HTA). Cognitive descriptions of the task using KLM strings are then imbedded to provide a relative timing for each elemental task. These times are then combined with the associated plans of the HTA to provide an overall task efficiency rating.

General terms: Hierarchical Task Analysis, Keystroke-Level Model, User Interface Design, System Efficiency Analysis

INTRODUCTION
The spectrum of available tools to aid in the development of a HTA when describing efficiency analysis was somewhat lacking. A niche to this exists when applied to interface analysis. Describing the task(s), the relative timing information is imbedded to help describe the cognitive framework for a typical user. This cognitive process is realized by the control and data transfer between user and widget, which is subsequently described by primitive keystrokes and actions. Low level GOMS [2, 4, 5] attempts to describe the operators of this data transfer. This analysis is applied to the widgets giving a more complete view of the task. When combined with a HTA, the resultant task tree describes practice of the task, cognitive framework and relative efficiency.

The term efficiency can be a surrogate for describing simplicity. Often those tasks which are efficient while maintaining basic design principles are also viewed as simple [3, 6, 7]. It then becomes important to include a keystroke level model to describe process and efficiency of subtasks.

VISUAL TA

In most software descriptions the concept of consistency and reuse are paramount to good development. What was needed was a tool which could mirror this design methodology promoting these very concepts.

Vis-TA was designed to allow a user to describe the process of the interface as a HTA with imbedded KLM strings. Each elementary task object is given a relative time based on accepted motor and mental practices which are represented as key press times and mental thought process times. The resultant strings represent the users' relative efficiency when using the interface widget.

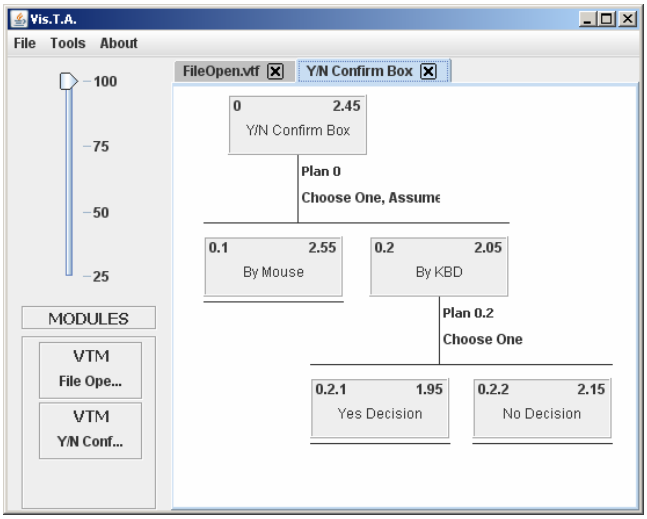

Figure 1. A Y/N Confirm dialog box can be interacted with using mouse controls or the keyboard. A user will choose one of these methods. If empirical studies show 80% of users prefer mouse then the weighting will be 0.8*(by mouse) + 0.2*(by KBD).

Vis-TA is purely graphical, allowing the user to see the hierarchy description of the root task. As KLM strings are defined for the primitive tasks, Vis-TA computes the overall

task time by summing the times of the individual tasks and propagating the result upward though the tree. Plans define how the times are processed. For instance, a Do-In-Sequence implies that all subtasks must be completed, in order, thus requiring the parent task to represent the sum of the times of the child tasks. In cases where a user can choose between 2 or more alternate methods of interaction, a Choose-One-Of plan can assign probabilistic values according to observed interaction. An example would be to use keyboard controls verses a mouse, see Figure 1, to perform an interaction. The sum is a weighed calculation. Do-In-Parallel and Do-In-Any-Order allow for further flexibility when describing tasks.

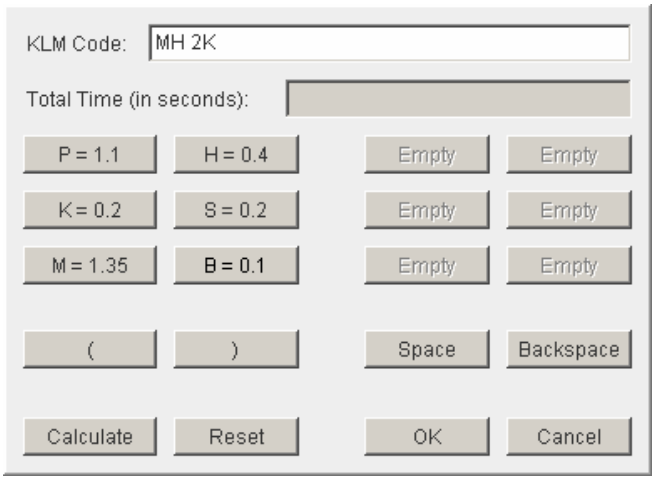

Figure 2, KLM editor. allow for complex strings to be inputted. Strings may use brackets and constant multipliers to enhance readability. The system calculates the time based on the defined symbols. All symbol times can be edited or custom symbols defined in the options menu.

Keystroke Level Model strings define a relative timing of a task for comparative rankings between designs when comparing systems. Vis-TA incorporates a KLM editor allowing the user to define the KLM strings using common infix syntax. This allows for repetition to be defined, example 4(M3K) implies 4 sets of 3 keystrokes each with mental preparation.

MODULES

Large systems are composed of many simple widgets, many of which are reused throughout the system. Vis-TA allows for widgets to be packaged as modules. These modules retain consistency over the system. If the module is edited, then all instances of the module reflect the change. Any HTA can be packaged and include other packaged modules as well. Modules are stored as files making the transfer between projects seemless.

USABILITY FEATURES

Vis-TA was designed to be easy to use. Emphasis was put toward a direct manipulation environment with object action control. Context sensitive menus allow for easy access to each task object.

Features include:

- Few static menus, interaction is object based.
- Direct Manipulation, everything in the environment can be manipulated with the mouse. Drag and drop editing.
- Tabbed windows, allows for many HTAs to be opened concurrently.
- Cut Copy Paste, on all objects, and object groups.
- Scroll wheel zoom, 4 levels of zoom completely controlled by the mouse wheel.
- Canvas can be dragged, replacing scroll bars.
- Ability to package any HTA into a module.
- Integrated KLM editor, allow for complex strings to be formulated for each elemental object.
- Auto update on all KLM calculations.

OPEN SOURCE

This project was pioneered with an open source architecture in mind, and will be released under the GNU GPL. It is expected that others can use this framework to further enhance and release updates, filling in some of the holes which were overlooked.

ACKNOWLEDGMENTS

The authors would like to recognize Corrado Coia for his contribution in pioneering the first version of Vis-TA. Many of his ideas have been incorporated into this version.

REFERENCES

1. Alan Dix, J.F., Gregory D. Abowd, Russell Beale. Human-Computer Interaction. in Hall, P.P. ed., 2004, 510-543.
2. Card, S.K., Moran, T. P., Newell, A. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associate, Publishers, London, 1983.
3. Feldman, A. Web Site Interface Design Theory: A Designer's Primer, 2006.
4. Kieras, D. A Guide to GOMS Model Usability Evaluation using NGOMSL. in *The handbook of human-computer interaction*, North-Holland, Amsterdam, 1997, 733-766.
5. Kieras, D. *Using the Keystroke-Level Model to Estimate Execution Times*, 2001.
6. Norman, D. *The Design of Everyday Things*. Basic Books, New York, 1988.
7. Raskin, J. *The Humane Interface*. Addison-Wesley, 2000.