SIMULATION OF THE GOMS KEYSTROCK LEVEL MODEL USING DEVS

Ali Mroue[(a)], Jean Caussanel[(b)]

(a)(b)Laboratory of sciences of informations and of systems
LSIS UMR 6168 University of Paul Cezanne, Aix-Marseille III
Av. escadrille de Normandie Niemen 13397 Marseille Cedex 20, France

(a)ali.mroue@lsis.org, (b) jean.caussanel@lsis.org

ABSTRACT
In this paper, we describe our approach for the simulation of GOMS model. The GOMS models characterized human interaction as a goal-directed search, or as general problem solving, in which the shortest route to any goal was considered the most efficient. DEVS is a formalism used to represent Discrete Event System Specifications; it can represent complex systems in an effective way.The DEVS formalism is a powerful tool for model simulation. In order to simulate GOMS models in an effective way, we propose a method to transform GOMS models to DEVS models in order to simulate and improve them. In this paper we will propose the transformation of the Keystroke-Level Model for GOMS into DEVS.

Keywords: DEVS,Goms KeyStrock, HCI

1. INTRODUCTION

In a design at the conceptual level, usability problems are among the most difficult to detect as design proceeds. The designers are unable or may be reluctant to make changes at the conceptual level either for time deadline or because the effort required. Operational prototypes frequently evolve into the final user interface. Evaluating the usability of an interactive system during the early stages of design, e.g. at the conceptual level, is therefore an important task. Predictive usability metrics are an emerging and promising approach to assessing and improving the quality of user interface designs.

One of the important approaches that can be used to predict the usability is GOMS family models, which allows the study of the usability of the interface easly with minimum cost and time, and allow detecting problems before start while still in the design process. There is no existing real simulator for GOMS models, and if it exists it will be in program form, and models will be attached to its application. Another important thing is that GOMS has many limitations, and we are working in improving it, being one of these limits the fact that GOMS models don't take into account the stress, fatigue, etc. of the user.

2. GOMS MODELS

The acronym GOMS stands for Goals, Operators, Methods, and Selection Rules (Card, Newell and Moran 1983). GOMS is a behaviour description model, that lets model the behaviour at different levels of abstraction, from task level to physical actions. GOMS uses as a starting point the Model Human Processor principle of rationality that attempts to model and predict user behavior.

Its essential contribution is a formal structure that allows organizing the design process.
The design method that induces GOMS is done on two axes. In the analysis of task (since determines the behavior). In the predictive evaluation of user behavior in the task.

2.1. GOMS ELEMENTS

Goals, Operators, Methods and Selection Rules (GOMS) are an approach to modeling human computer interaction.

GOMS models consist of descriptions of the methods required to accomplish a specific goal.
Methods are a sequence of operators and sub-goals to achieve a goal. If there exists more than one method to accomplish a goal then selection rules are used to choose which method to use.

- Goals are tasks the system's user wants to accomplish. For example, "Create Folder, Delete Word". A goal can have a hierarchical structure,this means that the achievement of a goal may require to accomplish one or more sub-goals.
- Operators are actions allowed by the software or actions that used are executing. An operator is an atomic level action that can't be composed, and it's characterized by its execution time. The execution of the operators causes change in the mental state of the users or in the environment state. There are two types of operators mental and physical, For example the operators "press enter", "point to the word", etc . . . . Are the physical operators.

The model also includes mental operators, such as "thinking", etc. . . .

- Methods refer to the process that allows one to accomplish a goal. Methods are possible sequences of operators and sub-goals that completely accomplish goals. For example to delete the word just typed, one could press the backspace key repeatedly until all characters have been erased, or i could move one's hand to the mouse, double-click the word and press the delete key. Each of these is a method.
- Selection Rules are used when there exists more than one method that can accomplish the same goal. They are the possible rules by which the user decides which method to choose in order to accomplish a particular goal. A rule has the form:
  If <condition>
  Then use the method M;

Goms has been used in many applications

- Telephone operator (CPM-GOMS)
- CAD systems (NGOMSL)
- Text editing using the mouse (KLM)

2.2. GOMS VARIATION

There are four different models of GOMS: CMN-GOMS, KLM, and NGOMSML CPM-GOMS.
CMN-GOMS stands for Card, Moran and Newell GOMS, is the original version of the GOMS technique in human computer interaction. This technique requires a strict goal-method-operation-selection rules structure.
KLM stands for Keystroke-Level Model based on CMN-GOMS developed by Card, Moran and Newell.

NGOMSML stands for Natural GOMS Language, developed by David Kieras (Kieras 2006). "An NGOMSL model is in program form, and provides predictions of operator sequences, execution time, and time required to learn the methods. Like CMN-GOMS, NGOMSL models explicitly represent the goal structure, and thus can represent high-level goals like collaboratively writing a research paper" (John and Kieras 1996).

CPM-GOMS stands for Cognitive, Perceptual, and Motor and the project planning technique Critical Path Method. CPM-GOMS was developed in 1988 by Bonnie John (John and Kieras 1996). CPM-GOMS does not make the assumption that operators are performed serially, and hence it can model the multitasking behavior that can be exhibited by experienced users. The technique is also based directly on the model human processor a simplified model of human response.

In this paper we will introduce the equivalent of the Keystroke Level Model GOMS in DEVS. Consequently we will only detail the keystroke-level model of GOMS.

2.3. KEYSTROCK-LEVEL MODEL

KLM is a simplified version of CMN-GOMS. It eliminates the goals, methods and rules of selection, leaving only the primitive operators. It contains several claims simplification, it uses operators at the level of primitive keystrokes for prediction. This model seeks to predict efficiency (time taken by expert users doing routine tasks) by breaking down the user's behavior into a sequence consisting of the five primitive operators.
The analyst indicates the method for accomplishing the goal. This method will be composed only of a sequence of strike level primitive operators.

The KLM contains five types of operators:
K - To press a key or button.
P - To point with a mouse to a target on a display device.
H - To home hands on keyboard or other device.
D - To draw a line segment on a grid.
M - To mentally prepare to do an action or closely related series of primitive actions
R - To symbolize the system's response time during which the user has to wait for the system.

Each of these operators has an estimation of their execution time. The time needed to execute a task or to accomplish a goal is the sum of all the time elapsed in the execution of every class of operators.
$Texec = TK + TP + TH + TD + TM + TR$

As we said, every operator is characterized by it's execution time:
TK = (total duration of the tests) / (number of keys hit without error)
20 s worst typist, 0.28 s average typist (40 wpm), 0.08 s best typist (155 wpm)

Pointing is determined by Fitts's Law:
$TP = TP = (K0 - TK) + I \log2(D/L+0.5)$
In general $TP \simeq 1.1$ s for all pointing tasks.
$TD = 0.9 \times n\ 0.16 \times l + n$ (n = number of segments to draw, l = sum of the segments' height)

The homing time is estimated by a simple experiment in which the user moves his/her hand back and forth from the keyboard to the mouse. TH = 0.4s

The M operator represents the time to prepare mentally for the next step in the method primarily to retrieve that step (the thing on has to do) from long-term memory. A step is a chunk of the method, so the M operators divide the method into chunks. The resulting estimate (from Card & Moran's paper of 1978) was 1.35 sec_TM = 1.35. $TR = 0$ if $n \leq t$, $TR = n - t$ if $n \succ t$
n: time required by the system to process the command
t: Is the time required by the user to execute an operator during the processing of the command.

The KLM technique includes a set of five heuristic rules for placing mental operators in order to account

for mental preparation time during a task that requires several physical operators.

0: Insert M

- In front of K's which are not part of an argument string.
- In front of all P's selecting a command.

1: Remove M between fully anticipated operators.

- PMK→PK

2: if a string of MKs belongs to a cognitive unit delete all M but the first.

- 4564.23:MKMKMKMKMKMKMK→MKKK KKKK

3: if K is a redundant terminator then delete M in front of It.

- press Enter, press Enter: MKMK→MKK

4a: if K terminates a constant string (command name) delete the M in front of it.

- cd enter: MKKMK→MKKK

4b: if K terminates a variable string (parameter) keep theMin front of it.

- cd class enter: MKKKMKKKKKMK→MKKKMKKKKKM K

2.3.1. USE OF KLM

To use KLM we must perform the three following steps

1. Encode using all physical operators (K, P, H, D(n,l), R(t)).
2. Apply KLM rule number "0" and introduce the M.
3. Apply KLM rule number "1, 2, 3 and 4" to remove the M.

Example:
Goal: Selection of a text
Method:
Grab the mouse
Point the mouse to the desired location
Select

First Step:
H[mouse] P[mouse] K[mouse-button] H[keyboard]

Second Step (Rule 0):
H[mouse] M P[mouse] M K[mouse-button] H[keyboard]

Third Step (Rules 1): H[mouse] M P[mouse] K[mouse-button] H[keyboard]

As a result we obtain: Tmethod = 2TH + TP + TK + TM.

3. DEVS FORMALISM

DEVS (Zeigler 1984) is a formalism used to represent Discrete Event System Specifications; it can represent complex system in an effective way. DEVS model is a powerful simulation model. It is a modular formalism for deterministic and causal systems. It allows for component-based design of complex systems. Several specific platforms for DEVS models simulation can be found.

A DEVS model may contain two kinds of DEVS components: Atomic DEVS and Coupled DEVS. An Atomic DEVS does not contain any component in it. It only has a mathematical specification of its behavior. A Coupled DEVS is a modular composition of one or more Atomic and Coupled DEVS.

An atomic DEVS model has the following structure:
$D = <XD, YD, SD, \delta extD, \delta intD , \lambda D, taD>$
Where:
XD: is the set of the input ports and values
YD: is the set of the output ports and values
SD: the sequential state set
$\delta extD$: $Q \times X \rightarrow S$, is the external transition function where $Q = (s, e) \rightarrow s \in S, 0 \leq e \leq ta(s)$ is the total state set.
$\delta intD$: $S \rightarrow S$, is the internal transition function
λD: $S \rightarrow Y$, is the output function
taD: $S \rightarrow R+0, \infty$ (non-negative real), is the time advance function

A coupled model, also called network of models, has the following structure:
$N = <X, Y, D, \{Md / d \in D\}, EIC, EOC, IC>$
X and Y definitions are identical to XD and YD of an atomic model. The inputs and outputs are made up of ports. Each port can take values and has its own field of values.
$X = \{ (\rho,\mu) / \rho \in IPorts, \mu \in X\rho \}$
$Y = \{ (\rho,\mu) / \rho \in OPorts, \mu \in Y\rho \}$
D is the set of the model names involved in the coupled model.

Md is a DEVS model. The variables representing the inputs and the outputs of the model will be indexed by the model identifier. Hence the following notation:
$Md = <Xd, Yd, Sd, \delta extd, \delta intd, \lambda d, tad>$
The inputs and the outputs of the coupled model are connected to the inputs and outputs of the included models.
$EIC = \{((N, a), (d, b))/a \in IPorts, b \in IPortsd\}$

The set of the coupled model input ports ipN associated with the input ports ipd of the models D are the components of the coupled model.

There is the same situation for the output ports.
$EOC = \{((N, a), (d, b))/a \in OPorts, b \in OPortsd\}$

Inside the coupled model, the outputs of a model can be coupled with the inputs of the other models. An output of a Model cannot be coupled with one of its inputs.
IC = {((i, a), (j, b))/i,j ∈ D, i, j,a ∈ OPortsi, b ∈ IPorts j }.

4. GOMS TO DEVS

We chose to transform GOMS models to the DEVS formalism for many reasons some of these reason are:

1. They have many common properties. GOMS Goal has hierarchical structure; DEVS have a hierarchical structure "Coupled Model". GOMS has an atomic element named operator that can't be decomposed and has an execution time property, the same for DEVS models, which have an atomic element named state that has its time life period assigned etc.
2. DEVS is very powerful in simulation, and we have a powerful simulator for DEVS models named LSISDME.

GOMS has some limitations, like for example that it doesn't take into account the fatigue of the user. So in the work presented in (Seck, Frydman and Giambiasi 2004).

3. a model for fatigue and stress of users is defined under DEVS. So the transformation of GOMS models to DEVS will help us to extend GOMS in order to add the model of fatigue and stress to GOMS.

5. TRANSFORMATION

In this paper we present only the transformation of KLM into DEVS models.
After studying the two approaches "DEVS-KLMGOMS" we propose the following rules for transformation:

- A KLM-GOMS model Corresponds to an atomic DEVS model that has 1 input and 1 output.
- The Goal is the name of the DEVS model
- The number of phases in this model is equivalent to the number of operators + two phases, one for the initial phase and the other for the end phase. The Life time of each phase is equivalent to the execution time of the corresponding operator.
- The phases are connected together by internal transitions.

Applying these rules to a KLM-GOMS example will give:

KLM GOMS:
Goal:
Select Text
Method:
H[mouse]MP[mouse]K[mouse button]H[keyboard]

The above KLM example mentioned is equivalent in DEVS to (Figure 1). Which is an atomic DEVS model having 2 initial phases "init and End phases" plus "n" other phases. The n represent the number of operators used in the model etc…

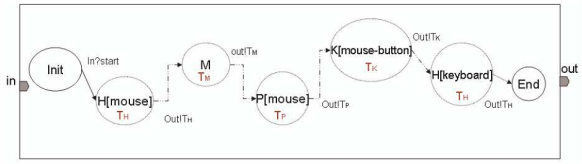

Figure 1: DEVS "Select Text" KLM Goal

6. RESULTS

By giving the equivalence for GOMS keystrock level model in DEVS, we have added many features to its initial representation.

First of all the proposed keystrock DEVS representation gives an effective way for simulating keystrock model. This simulation can be done automatically in an easy way using our laboratory simulator "LSISDME".

Secondly, any change or add of new operators can be done easily without any modification or recompilation of the simulation tool "LSISDME". Other simulators have to be modified and recompiled in order to take into account new operators or new functions.

Another important point for the transformation to DEVS is that we have now the capability to add existing models such as the stress model defined by (Seck, Frydman and Giambiasi 2004) and take into account this human factor. Note that coupling the Keystrock and the stress models will extend the original one (Keystrock Level GOMS) and solve some of the disadvantages of GOMS models.

7. CONCLUSION

In this paper we introduced GOMS models and the DEVS formalism; we explained the reasons of our choice for the DEVS formalism for simulation GOMS Model. We have detailed the KeyStrock Level Model, and we defined all its' elements in the DEVS formalism. In this work we succeeded in finding a new simulation tool for the Keystrock Level Model that is effective and gives good results. This transformation into DEVS has many advantages, first we have obtained an effective simulation tool for Keystrock level models, and secondly we can easily interpret these models and extend them in order to add many other modules. We can also do the simulation in interactive mode, and implement the core of the DEVS simulator in a user training applications. The current prospects relate to complete transformation of all GOMS model family to the DEVS formalism in order to simulate it, and then extend it so as to take in consideration the user stress and fatigue, Also we will model the task of search of information on the web and simulate it.

REFERENCES

Card, S. K.; Newell, A. and Moran, T. P. 1983. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA.

Kieras, D.E. 2006. *A Guide to GOMS Model Usability Evaluation using GOMSL and GLEAN4*. University of Michigan

John B. E. and Kieras D. E. 1996. *The GOMS family of user interface analysis techniques: comparison and contrast.* ACM Transactions on Computer-Human Interaction, 3(4):320–351.

Seck M., Frydman C., and Giambiasi N. 2004. *Using DEVS for modeling and simulation of human behavior.* In Lecture Notes in Computer Science special issue DEVS modelling and simulation, volume 3397, page 692, février 2005. Pub. Springer Verlag - Ed. Tag Gon Kim - ISSN : 0302–9743 - ISBN : 3–540-24476-X

Zeigler B. P. 1984. *Theory of Modelling and Simulation.* Krieger Publishing Co., Inc. Melbourne, FL, USA.