Automatic Evaluation of Assistive Interfaces

Pradipta Biswas
Computer Laboratory
15 JJ Thomson Avenue
Cambridge CB3 0FD
University of Cambridge, UK
E-mail: pb400@cam.ac.uk

Peter Robinson
Computer Laboratory
15 JJ Thomson Avenue
Cambridge CB3 0FD
University of Cambridge, UK
E-mail: pr10@cam.ac.uk

ABSTRACT

Computers offer valuable assistance to people with physical disabilities. However designing human-computer interfaces for these users is complicated. The range of abilities is more diverse than for able-bodied users, which makes analytical modelling harder. Practical user trials are also difficult and time consuming. We are developing a simulator to help with the evaluation of assistive interfaces. It can predict the likely interaction patterns when undertaking a task using a variety of input devices, and estimate the time to complete the task in the presence of different disabilities and for different levels of skill. In this paper we describe the different components of the simulator in detail and present a prototype of its implementation.

Categories and Subject Descriptors

D.2.2 [Software Engineering]: Design Tools and Techniques – *user interfaces;* **K.4.2 [Computers and Society]:** Social Issues – *assistive technologies for persons with disabilities*

General Terms

Algorithms, Experimentation, Human Factors, Measurement

Keywords

Human Computer Interaction, Assistive Technology, Usability Evaluation, Simulator.

INTRODUCTION

Computers offer valuable opportunities to physically challenged people as it help them to engage more fully with the world. However designing and evaluating human-computer interfaces for these users is more complicated than for able-bodied persons, since the range of abilities is more diverse. Their patterns of interaction are also significantly different from those of able-bodied users. So existing HCI models are not easily applicable to assistive interfaces. Assistive interfaces are generally evaluated by analysing log files after a user trial. However it is often difficult to find participants with specific disabilities. Petrie et. al. [10] take the approach of remote evaluation but still need to find disabled participants. As an alternative, a modelling tool that could simulate HCI of users with disabilities would relieve the designer from searching for disabled participants to run a conventional user trial. However, research on assistive interfaces and HCI modelling do not overlap. Very few HCI models have considered users with disability. Researchers on assistive interfaces have concentrated on designing assistive interfaces for a particular application (e.g. Web Browser, Augmentative and Alternative Communication aid etc.), developing new interaction techniques (e.g. different scanning techniques) or developing novel hardware interfaces (head mounted switches, eye-gaze trackers, brain-computer interfaces etc.). They have not looked at designing a systematic modelling tool for assistive interfaces. We have developed a simulator to model HCI of disabled users. It can predict the likely interaction patterns of users when undertaking a task using a variety of input devices, and estimate the time to complete a task in the presence of different disabilities and for different levels of skill. The simulator can be used to compare several existing assistive interfaces and to evaluate new alternatives. We also address the shortcomings of existing HCI models and hope to develop a system that will be easier to use than the existing models and support both able-bodied and disabled users.

RELATED WORKS

The GOMS family of HCI models (e.g. KLM, CMN-GOMS, CPM-GOMS) is mainly suitable for modelling the optimal behaviour (skilled behaviour) of users [5]. On the other hand, models developed using cognitive architectures consider the uncertainty of human behaviour in detail but have not been widely adopted for simulating HCI. For example, developing a sequence of production rules for Soar [3], a semantic network for ACT-R [15] or a set of constraints for CORE [11] is difficult with respect to an interface designer. Usability issues for cognitive architectures are also supported by the X-PRT system [11] for the CORE architecture. Additionally, Kieras has shown that a high fidelity model cannot always outperform a low fidelity one though it is expected to do so [7]. Researchers have already attempted to combine these two forms of model to develop more usable and accurate models. Salvucci and Lee [9] have developed the ACT-Simple model by translating basic GOMS operations into ACT-R production rules [15]. The model works well to predict expert performance but does not work for novices. Blandford et. al. [1] implement the Programmable User Model (PUM) using the Soar architecture. They developed a program, STILE (Soar Translation from Instruction Language made Easy), to convert the PUM Instruction Language into Soar productions. However, this approach also demands good knowledge of Soar from an interface designer. The second problem of existing approaches to modelling comes from specific issues with disability. There is not much reported work on systematic modelling of assistive interfaces. McMillan [34] felt the need to use HCI models to unify different research streams in assistive technology, but his work aimed to model the system rather than the user. The AVANTI project [6] models an assistive interface for a web browser based on some static and dynamic characteristics of users. However, this model does not address the basic perceptual, cognitive and motor behaviour of users and so it is hard to generalize to other applications. Our user model [23] breaks down the task of user modelling into several steps that include clustering users based on their physical and cognitive abilities, customizing interfaces based on user characteristics and logging user interactions to update the model itself. However the objective of this model is to design adaptable interfaces and not to simulate users' performance. Keates et. al. [30] measured the difference between able-bodied and motor-impaired users with respect to the Model Human Processor (MHP) and motor-impaired users were found to have a greater motor action time than their able-bodied counterparts. The finding is obviously important, but the KLM model itself is too primitive to use.

OUR OBJECTIVE

Based on the previous discussion, Figure 1 plots the existing general-purpose HCI models in a space defined by the skill and physical ability of users. To cover most of the blank spaces in the diagram, we set our objectives to develop a model that can:

1. Simulate HCI of both able-bodied and disabled users.
2. Work for users with different levels of skill.
3. Be easy to use and comprehend for an interface designer.

THE SIMULATOR

We are developing a simulator that takes a task definition and locations of different objects in an interface as input. Then it predicts the cursor trace, probable eye movements in screen and task completion time, for different input device configurations (e.g. mouse or single switch scanning systems) and undertaken by persons with different levels of skill and physical disabilities.

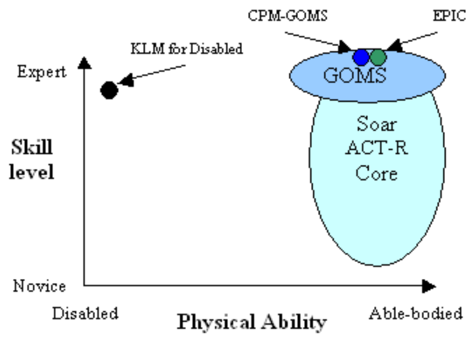


Figure 1. Existing HCI models w.r.t. skill and physical-ability of users

The architecture of the simulator is shown in Figure 2. It consists of the following three components:

The Application model represents the task currently undertaken by the user by breaking it up into a set of simple atomic tasks.

The Interface Model decides the type of input and output devices to be used by a particular user and sets parameters for an interface.

The User Model simulates the interaction patterns of users for undertaking a task analysed by the task model under the configuration set by the interface model. It uses the sequence of phases defined by Model Human Processor. The perception model simulates the visual perception of interface objects. The cognitive model takes the output of the perception model and determines an action to accomplish the current task. The motor-behaviour model predicts the completion time and possible interaction patterns for performing that action. A case study of using the simulator can be found in [22] while an application of the model in evaluating different single-switch scanning techniques is presented in [21].

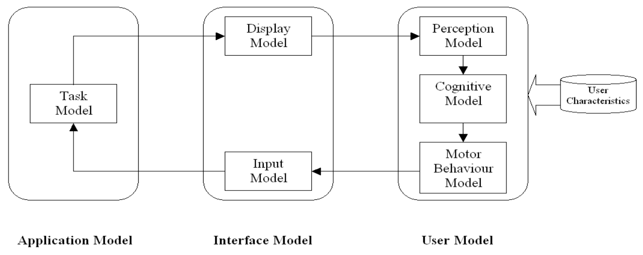


Figure 2. Architecture of the Simulator

The remainder of this paper discusses the design of these three components of the user model. The perception model is designed according to the theories of visual attention. Our cognitive model is more detailed than the GOMS model but not as complex as existing cognitive architectures. The motor-behaviour model is developed by statistical analysis of screen navigation paths of disabled users.

THE PERCEPTION MODEL

Among existing systems, only EPIC [8] and ACT-R/PM [19] have distinct perception models. Currently our perception model considers only vision. It takes a list of keyboard and mouse events and a sequence of bitmap images of an interface as input and produces a sequence of eye-movements and the visual search time as output.

We perceive something in a computer screen by focusing attention at a portion of the screen and then searching for the desired object within that area. If the intended item is not found in that area then attention is shifted to a new location. Our model supports both systematic and random mechanisms of shifting attention [13,27] and also the top down and bottom up theories [8,19] of focusing attention. We model the bottom up theory by analysing a bitmap image of the interface using different computer vision algorithms (e.g. colour-histogram matching, Shape matching etc.). The top down mechanism is modelled in the form of heuristics (e.g. the model never searches in a region of screen which does not contain any controls or it does not undertake a visual search for common operations like minimizing, maximizing or closing a window etc.). The model is controlled by four free parameters: distance of user from the screen, foeveal, parafoveal and periphery angles. The default values of these parameters are set according to the EPIC architecture [8]. By changing resolution or by proper filtering of the bitmap images, the model can also be used to simulate vision of different visually impaired users.

THE COGNITIVE MODEL

We have modelled the optimal (expert) and sub-optimal (non-expert) behaviour separately. We have used the CPM-GOMS [5] model to simulate the optimal behaviour. For sub-optimal behaviour, we have developed a new model. This model takes a task definition as input and produces a sequence of operations needed to accomplish the task as output. It simulates interaction patterns of non-expert users by two interacting Markov processes. One of them models the user's view of the system and the other signifies the designer's view of the system. Users operate in the users' space to achieve their goals. They do it by converting their intended actions into an operation offered by the device. At the same time, they map a state of the device space into a state of the user space to decide the next action. Users behave sub-optimally, when these mappings between the device space and the user space are not done optimally. We can summarize our assumptions as follows:

- o Users and devices operate in two different state spaces [16].
- o Each state space can be modelled as a Markov Decision Process. This is consistent with the fact of finite capacity of short-term memory of humans.
- o Users follow the principle of maximum rationality [3], so if they know an action to achieve their goal, then they will select that action.
- o Users behave sub-optimally by not properly converting their intended action into a device operation and misperception of a device state.
- o A good interface will minimize the mismatch between the user space and the device space.

The performance of the system is illustrated in Figure 3. At any state, users have a fixed policy based on the current task in hand. The policy produces an action, which in turn is converted into a device operation (e.g. clicking on a button, selecting a menu item etc.). After application of the operation, the device moves to a new state. Users have to map this state to one of the state in the user space. Then they again decide a new action until the new state becomes the goal state.

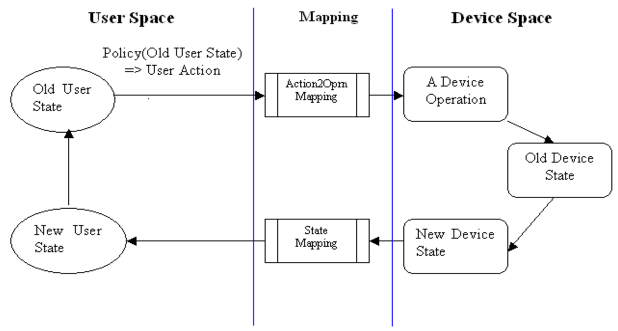


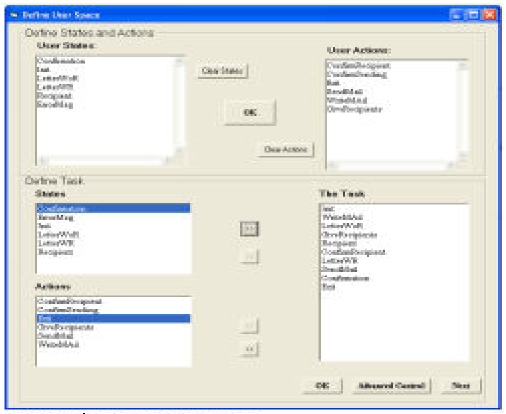
Figure 3. Sequence of events in an interaction

Learning

Besides performance simulation, our model also has the capability of learning new techniques of interactions. Learning can occur either offline or online. The offline learning takes place when the user of the model adds new states or operations to the user space. The model can also learn new state and operations itself. During execution, whenever the model cannot map the intended action of the user into an operation permissible by the device, it tries to learn a new operation. To do so, it first asks for instruction from outside. The interface designer is provided with the information about previous, current and future states and she can choose an operation on behalf of the model. If the model does not get any instruction from outside then it searches the state transition matrix of the device space and selects an operation according to the label-matching principle [16]. If the label matching principle cannot return a prospective operation, it randomly selects an operation that can change the device state in a favourable way. It then adds this new operation to the user space and updates the state transition matrix of the user space accordingly. In the same way, the model can also learn a new device state. Whenever it arrives in a device state unknown to the user space, it adds this new state to the user space. Then it selects or learns an operation that can bring the device into a state desirable to the user. If it cannot reach a desirable state, it just selects or learns an operation that can bring the device into a state known to the user. The model can also simulate the practice effect of users. Initially the mapping between the user space and the device space remains uncertain (i.e. the probabilities for each pair of state/action in the user space and state/operation in the device space is less than 1). After each successful completion of a task the model increases the probabilities of those mappings that leads to the successful completion of the task and after sufficient practice the probability values of certain mappings reach one. At this stage the user can map his space unambiguously to the device space and thus behave optimally.

Usability

One important aspect of a cognitive model is its own usability, which is mostly ignored in the current literature on cognitive models. We developed user interfaces for developing and running the model (Figures 4 and 5 respectively). The model should be developed in three steps. In the first step, the designer has to specify some possible user states and actions. Then she has to define a state transition diagram for the current task by selecting a state and an action alternatively. This can be done with the help of a physical DFD (for structured design) or a state-transition diagram (for object-oriented design) developed as part of the system design document. Individual entries of the state transition diagram can be modified by clicking on the 'Advanced Control' button. In step 2, all of the previous operations have to be repeated for developing the device space. Finally in step 3, the states and actions of the user space and the device space have to be mapped with each other. The mapping can be done by defining a joint probability distribution matrix using the interface shown in Figure 4d. The interface designer is also free to choose any advanced modelling techniques (like rule-based system or a decision network) to model the mapping between the user space and the device space. Once developed, the model can be run using the interface shown in Figure 5a. At this stage, the system also permits to define and simulate a new task (Figure 5b). We have demonstrated the use of the model for a simple but non-trivial example in the next section.



User States: Confirmation, Init, LetterWoR, LetterWR, Recipient, ReadMsg

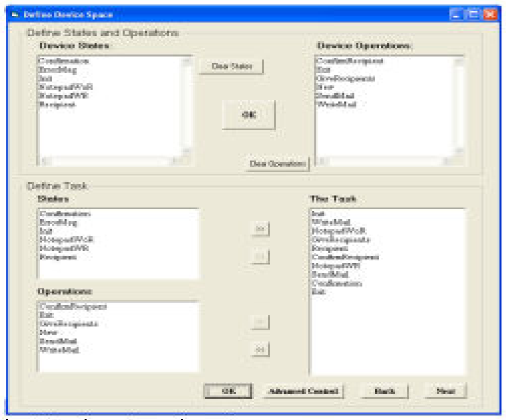
User Actions: ConfirmRecipient, ConfirmSending, Exit, SendMail, WriteMail, GiveRecipients

States: Confirmation, ReadMsg, Init, LetterWoR, LetterWR, Recipient

Actions: ConfirmRecipient, ConfirmSending, Exit, GiveRecipients, SendMail, WriteMail

The Task: Init, WriteMail, LetterWoR, GiveRecipients, Recipient, ConfirmRecipient, LetterWR, SendMail, Confirmation, Exit

a. Design User Space



Device States: Confirmation, ReadMsg, Init, NotepadWoR, NotepadWR, Recipient

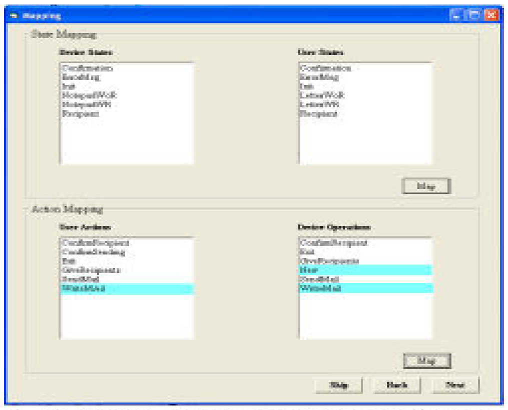
Device Operations: ConfirmRecipient, Exit, GiveRecipients, New, SendMail, WriteMail

States: Confirmation, ReadMsg, Init, NotepadWoR, NotepadWR, Recipient

Operations: ConfirmRecipient, Exit, GiveRecipients, New, SendMail, WriteMail

The Task: Init, WriteMail, NotepadWoR, GiveRecipients, Recipient, ConfirmRecipient, NotepadWR, SendMail, Confirmation, Exit

b. Design Device Space



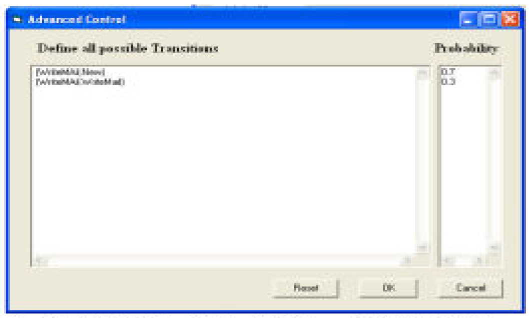
Device States: Confirmation, ReadMsg, Init, NotepadWoR, NotepadWR, Recipient

User States: Confirmation, ReadMsg, Init, LetterWoR, LetterWR, Recipient

User Actions: ConfirmRecipient, ConfirmSending, Exit, GiveRecipients, SendMail, WriteMail

Device Operations: ConfirmRecipient, Exit, GiveRecipients, New, SendMail, WriteMail

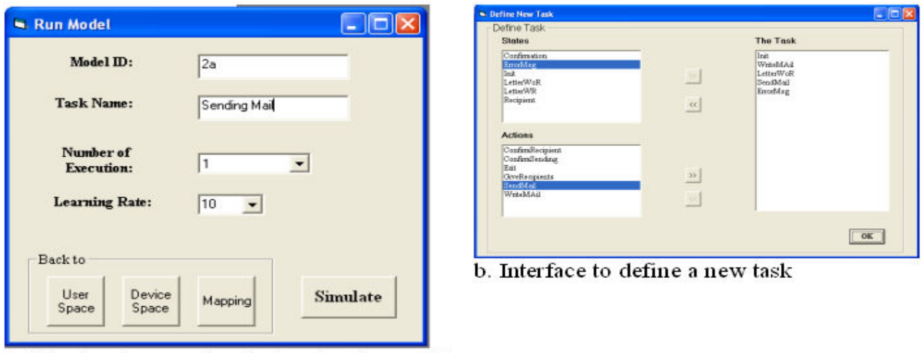
c. Mapping between the User Space and the Device Space



Define all possible Transitions	Probability
[WriteMAil,New]	0.7
[WriteMAil,WriteMail]	0.3

d. Mapping by Joint Probability Distribution Matrix

Figure 4. Interfaces to develop the model



Model ID: 2a

Task Name: Sending Mail

Number of Execution: 1

Learning Rate: 10

States: Confirmation, ErrorMsg, Init, LetterWoR, LetterWR, Recipient

Actions: ConfirmRecipient, ConfirmSending, Exit, GiveRecipients, SendMail, WriteMAil

The Task: Init, WriteMAil, LetterWoR, SendMail, ErrorMsg

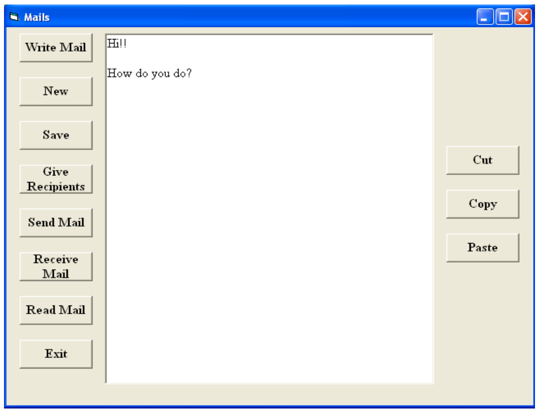
a. Interface to execute a task using the model

b. Interface to define a new task

Figure 5. Interfaces to run the model

Demonstration

We have modelled a situation of sending e-mails using our system. Initially we developed a very simple interface (Figure 6) for sending and receiving e-mails. The interface did not impose or indicate any particular order of operations and allowed the user to do any operation at any time. So it helped us to observe the natural interaction patterns of users while sending or receiving e-mails. The device model was developed from the interface itself. The state transition diagram of the device space is shown in Figure 7. We developed the user space by collecting interaction patterns from 5 participants on the interface. The participants were expert computer users but none used the interface before. They were aged between 25 to 35 years. The state transition diagram of the user space is shown in figure 8. The mapping between the user space and the device space is presented in table 1. We ran the model for two iterations to simulate the task of sending an e-mail using this particular interface. The output of the model is shown in table 2.



Hi!!

How do you do?

Figure 6. An interface to send and receive e-mails

In this particular example, the difference between the user space and the device space lies in the interpretation of the 'Send Mail' operator. Users expected after clicking on the 'Send Mail' button, they would automatically be asked to specify recipient, which was not supported by the device. So during executing the task for the first time, the model encountered the error message and learned the operation 'Give Recipient'. After specifying the recipient, the user wanted to confirm the sending operation. The 'ConfirmSending' action did not have any matching operation in the device space. At this stage the model applied the label matching principle, which successfully returned the 'Send Mail' operation in the device space. At the next iteration, the model performed the task optimally by using its learned knowledge. Thus this simple example demonstrates how the model can simulate the performance and learning of first-time users of an interface.

Table 1. Mapping between the user space and device space

User Space	Device Space
States	
Ready to write mail	Welcome Screen
Letter without recipient	Notepad without recipient
Specify recipient	Specify recipient
Letter with recipient	Notepad with recipient
Confirmation Message	Confirmation Message
Actions	
Write Mail	Write Mail
Send Mail	Send Mail
Confirm Recipient(s)	Confirm Recipient(s)

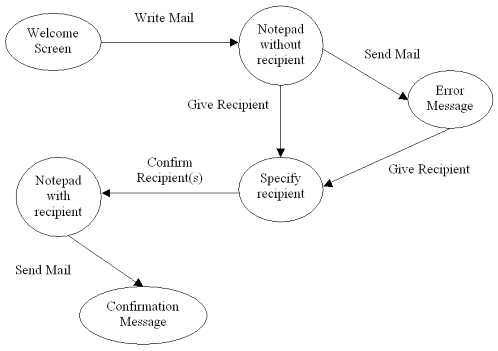


Figure 7. State transition diagram of the device space

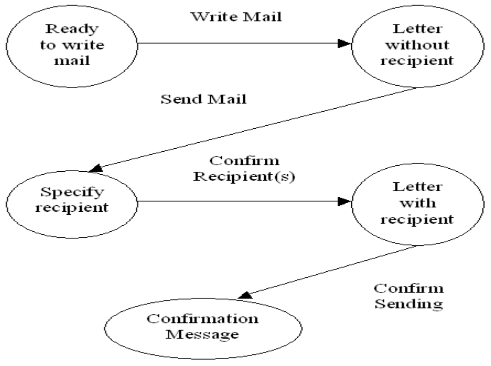


Figure 8. State transition diagram of the user space

Table 2. Output of the cognitive model

	Device Space	**User Space**
Iteration 1		
State	Welcome Screen	Ready to write mail
Action	Write Mail	WriteMail
State	Notepad without recipient	Letter without recipient
Action	SendMail	SendMail
State	ErrorMsg	
New Action Learned	GiveRecipients	
State	Specify recipient	Specify recipient
Action	ConfirmRecipient	ConfirmRecipient
State	Notepad with recipient	Letter with recipient
New Action Learned	SendMail	
Action	SendMail	Confirm Sending
State	Confirmation	Confirmation
Iteration 2		
State	Welcome Screen	Ready to write mail
Action	Write Mail	WriteMail
State	Notepad without recipient	Letter without recipient
Action	GiveRecipients	GiveRecipients
State	Recipient	Recipient
Action	ConfirmRecipient	ConfirmRecipient
State	Notepad with recipient	Letter with recipient
Action	SendMail	Confirm Sending
State	Confirmation	Confirmation

THE MOTOR-BEHAVIOUR MODEL

A motor behaviour model simulates movement limits and capabilities of users for different input devices and interaction techniques [12]. For able-bodied users, most motor-behaviour models are based on Fitts' Law [26] and its variations [12]. For disabled users, there is growing evidence that their interaction patterns are significantly different from those of their able-bodied counterparts [29-32]. However the applicability of Fitts' law for motor-impaired users is a debatable issue. Smits-Engelsman et. al. [4], Wobbrock and Gajos [14] found it to be applicable for children with congential spastic hemiplegia and motor-impaired people respectively, but Bravo et. al. [24] and Gump et. al. [2] obtained a different result. In general for real life pointing tasks, motor-impaired persons are not always governed by the visual feedback. Their movements seem to be more ballistic (rapid and discrete movement without visual feedback, [2]). This may be a result of their poor coordination between perception and motor-action. This poor coordination causes more neuro-motor noise than the permissible limit of Fitts' law [25]. They obey Fitts' law when the task is very simple and thus requires less coordination between vision and motor-action [4] or there are other cues (e.g. auditory) besides vision [14]. There has been some works to develop an alternative to Fitts' law for motor-impaired people. Gump et. al. [2] found significant correlation between the movement time and the root of movement amplitude (Ballistic Movement Factor [17]). Gajos, Wobbrock and Weld [18] estimated the movement time by selecting a set of features from a pool of seven functions of movement amplitude and target width, and then using the selected features in a linear regression model. We have developed the motor-behaviour model by statistical analysis of cursor traces of a previous experiment [32]. We did a more detailed analysis of different phases of movement for several pointing tasks undertaken by motor-impaired users and developed a model to predict the movement time for a pointing task. We investigated the cursor traces for each individual pointing task. The main difference between the mouse movement of the motor-impaired and able-bodied users lie in the characteristics of the sub-movements [29,31]. Able-bodied users move the mouse towards the target by a single long sub-movement followed by some smaller sub-movements to home on the target. In the case of motor-impaired users, the number of sub-movements is greater than that of able-bodied users and the main movement towards the target is often composed of two or more sub-movements. The time spent between two sub-movements (described as pause) also significantly affects the total task completion time. So our model estimates the total task completion time by calculating the average number of sub-movements in a single pointing task, their average duration, and the average duration of pauses. In the present study, we define a pause as the event when the mouse stops movement for more than 100 msec and a sub-movement is defined as a movement occurring between two pauses. To reveal the characteristics of the sub-movements and the pauses, we clustered the points where the pauses occurred (i.e. a new sub-movement started). We found that about 90% of the sub-movements took place when the mouse pointer was very near the source (the pointer had not moved more than 20% of the total distance) or near the target (the pointer had moved more than 85% of the total distance). The sub-movements near the source and target are rather ballistic and the remaining 10% of the sub-movements actually constituted the main movement. So our model divided the sub-movements and pauses during a pointing task into three classes based on their position with respect to the source and the target. The model operates based on the following equation.

$$\text{Movement Time} = p_1(d_1+s_1) + p_2*d_2 + f\,(\text{Dist}/v_2) + p_3(d_3+s_3) - (s_1+s_3)$$

Where,

Dist	Distance from source to target
p_1	No. of pauses near source
d_1	Average duration of a pause near source
s_1	Average duration of a sub-movement near source
p_2	No. of pauses in main movement
d_2	Average duration of a pause in main movement
v_2	Speed of movement in main movement
f	Fraction of the total distance covered by the main movement
p_3	No. of pauses near target
d_3	Average duration of a pause near target
s_3	Average duration of a sub-movement near target

We have estimated each of these model parameters from statistical analyses. One challenging task in developing the model was to categorize users based on their extent of disabilities. Several clinical scales have been used to measure disability (e.g. Ashworth scale [20], the weighted disability score [28], Tardieu Scale, Spasticity Grading [35] etc.), but they are hardly applicable in modelling HCI. In the present set of data, the experimenters categorized the users in several ways based on their experience, difficulty in clicking, pointing, dragging etc. Among these we found that a scale based on the difficulty in dragging, is significantly correlated ($p<0.05$) with three model parameters (No. of pauses near source, No. of pauses near target [Figure 9] and average speed of main movement [Figure 10]). We drew histograms of other parameters (Figure 11) and then they were approximated by the inverse transform method [33]. However in developing the model we assumed a fixed boundary among the three regions (near source, main movement, near target). To make the model more realistic, we blurred these boundaries. We calculated the probability of a pause from the function shown in Figure 12. As can be seen from Figure 12, the probability of a pause gradually increases to 1 near the source and the target. We estimated the pause durations by multiplying it with the probability of occurrence of a pause.

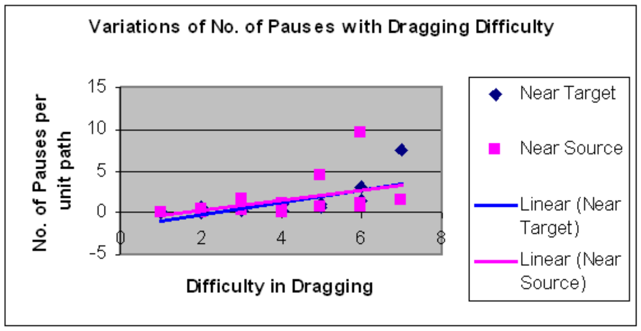


Figure 9.Variation of number of pauses w.r.t. a scale based on difficulty in dragging

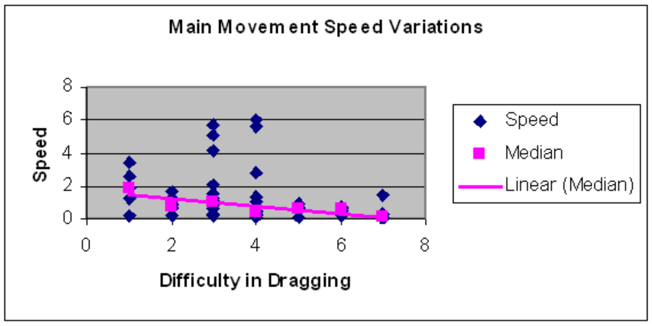


Figure 10. Variation of speed of main movement w.r.t. a scale based on difficulty in dragging

To estimate the accuracy of our model, we tested the model on 62 pointing tasks undertaken by 15 participants. The predictions are obtained by running Monte-Carlo simulation 500 times for each pointing task. The actual and predicted average task completion times and a Z-score distribution of the actual and predictions are shown in table 3 and Figure 14 respectively. Figure 13 presents a scatter diagram of actual and average predicted time. The median of the z-scores has come at –0.27 instead of 0, however the predicted average task completion time is found to be significantly correlated ($p<0.002$) with the actual.

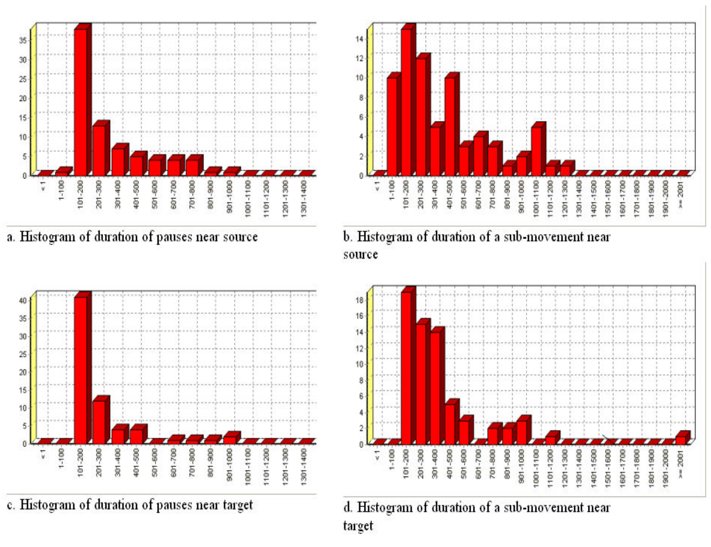


Figure 11. Histograms of model parameters

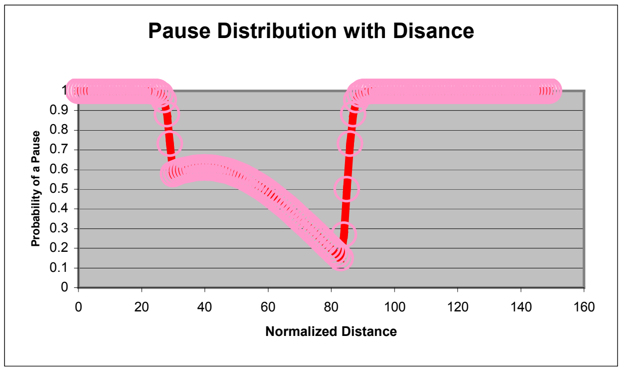


Figure 12. Probability of occurrence of a pause

Table 3. Actual and Predicted Task Completion Time

Participants	Average Predicted Time (msec.)	Actual Time (msec.)
P1	3566	1880
P2	4138	2176
P3	3418	2400
P4	4018	2500
P5	3920	2907
P7	14632	10309
P9	7389	2796
P11	687	1293
P12	14512	9349
P14	14974	22833
P15	4134	10478
P16	3584	1629
P17	7895	15888
P19	4018	2335
P20	3188	8771
Pearson r	0.71	
t	3.64	
p	0.0015	

IMPLEMENTATION

We have developed the simulator in a modular fashion – all of its components can be run independently of each other as well as together. The sequence of operations during execution of the simulator is shown in Figure 15. The cognitive model takes a task description from the task model and produces a list of low-level device operations. The interface designer has to execute these operations manually while our mouse hooking program runs as a daemon.

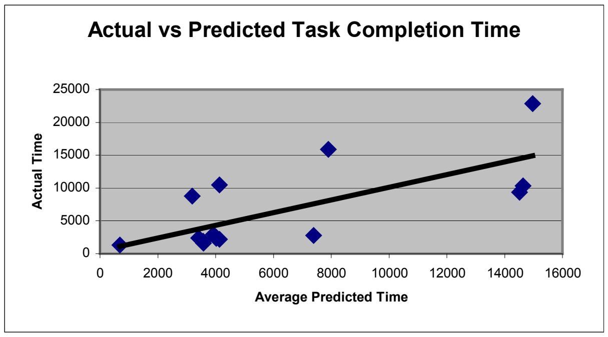


Figure 13. Scatter Diagram of Actual vs. Predicted Task Completion Time (in msec.)

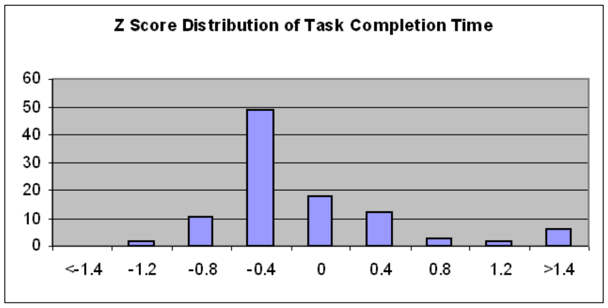


Figure 14. Prediction from our model for mouse interface

The mouse-hooking program generates a list of events (a list of key presses and mouse clicks), a low-level snapshot (a sequence of bitmap images) and a high-level snapshot (locations of windows, icons, buttons and other controls in the screen) of the whole interaction. The perception model operates on the event list and the sequence of bitmaps while the motor-behaviour model takes the event list and the high-level snapshot as input. An interface designer is free to use any one or more than one modules of the system. For example, one can run a GOMS analysis on the output of the cognitive model instead of using our perception or motor-behaviour model. Similarly the mouse-hooking program can be run for any interaction that is not produced by our cognitive model and the perception and (or) the motor-behaviour model can be used on the output of the mouse-hooking program.

CONCLUSIONS

In this paper we have presented a simulator that can predict the likely interaction patterns when undertaking a task using a variety of input devices, and estimate the time to complete the task in the presence of different disabilities and for different levels of skill. We have developed the simulator using the concept of Model Human Processor and described each modules of the system in details. We are now working to increase the accuracy of the model and to validate it by some experiments with people with disabilities.

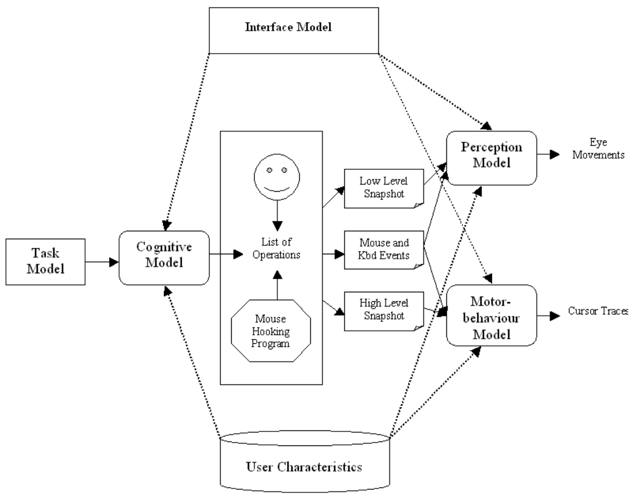


Figure 15. Sequence of operations in the simulator

ACKNOWLEDGEMENT

We would like to thank the Gates Cambridge Trust for funding this work. We like to thank the students of Computer Laboratory and Trinity College, Cambridge to take part in our experiments. We are also grateful to Dr. Shari Trewin of IBM TJ Watson Research Centre and Dr. Helen Pain of University of Edinburgh for sharing their data with us.

REFERENCES

[1] A. Blandford, R. Butterworthb and P. Curzonb, Models of interactive systems: a case study on programmable user modelling, International Journal of Human-Computer Studies, vol. 60 (2004), 149–200

[2] A. Gump et. al., Application of Fitts' Law to individuals with cerebral palsy, Perceptual and Motor Skills (2002), 94, 883-895

[3] A. Newell, Unified Theories of Cognition. Harvard University Press, Cambridge, MA, 1990

[4] B. C. M. Smits-Engelsman et. al., Children with congential spastic hemiplegia obey Fitts' Law in a visually guided tapping task, Journal of Experimental Brain Research (2007), 177, 431-439

[5] B. E. John and B. E Kieras., The GOMS family of user interface analysis techniques: Comparison and Contrast. ACM Transactions on Computer Human Interaction, Vol. 3 (1996), 320-351

[6] C. Stephanidis, et. al., Adaptable and Adaptive User Interfaces for Disabled Users in the AVANTI Project, Intelligence in Services and Networks, LNCS-1430, Springer-Verlag 1998, 153-166

[7] D. E Kieras. Fidelity Issues In Cognitive Architectures For HCI Modelling: Be Careful What You Wish For. In Proceedings of 11th International Conference On Human Computer Interaction (HCII 2005). Las Vegas, July, 2005

[8] D. Kieras and D.E. Meyer, An Overview of The EPIC Architecture For Cognition And Performance With Application To Human-Computer Interaction, Human-Computer Interaction (1990), vol. 12, 391-438

[9] D.D. Salvucci and F.J. Lee, Simple cognitive Modelling in a complex cognitive architecture, In Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing Systems, Fort Lauderdale, FL, 2003, 265–272

[10] H. Petrie et. al., Remote usability Evaluations with disabled people. . In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '06) , Montreal, Canada, April 22-27, 2006. ACM Press, New York, NY, 2006, 1133- 1141

[11] I Tollinger. et. al., Supporting Efficient Development of Cognitive Models At Multiple Skill Levels: Exploring Recent Advances In Constraint-Based Modeling, In Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing Systems, Portland, Oregon, USA,2005, 411 – 420

[12] I. S. MacKenzie, motor-behaviour models for human-computer interaction. In J. M. Carroll (Ed.) HCI models, theories, and frameworks: Toward a multidisciplinary science. 27-54. (2003) San Francisco: Morgan Kaufmann

[13] J. H. Reynolds and R. Desimone, The Role of Neural Mechanisms of Attention In Solving The Binding Problem, Neuron 24 (1999), 111-125

[14] J. O. Wobbrock and K. Z. Gajos, A Comparison of area pointing and goal crossing for people with and without motor impairments, . In Proceedings of 9th International ACM/SIGACCESS Conference on Computers and Accessibility (ASSETS 2007) (To appear)

[15] J. R. Anderson and C. Lebiere, The Atomic Components of Thought. Hillsdale, NJ: Erlbaum, 1998

[16] J. Rieman and R. M. Young, A dual-space model of iteratively deepening exploratory learning, International Journal of Human-Computer Studies (1996) 44, 743-775

[17] K. C. Gan and E. R. Hoffmann, Geometrical conditions for ballistic and visually controlled movements, Ergonomics (1988), 31, 829-839

[18] K. Z. Gajos, J. O. Wobbrock and D. S. Weld, Automatically generating user interfaces adapted to users' motor and vision capabilities, In proceedings of UIST 2007.

[19] M. D. Byrne, ACT-R/PM And Menu Selection: Applying A Cognitive Architecture To HCI, International Journal of Human Computer Studies (2001), vol. 55

[20] M. P. Barnes and G.P. Johnson, Upper Motor Neurone Syndrome And Spasticity, Cambridge University Press, U.K., 2001

[21] P. Biswas and P. Robinson, Performance Comparison of Different Scanning System using a Simulator, Proceedings of the 9th European Conference of Advancement of Assistive Technology in Europe (AAATE 07) (To appear)

[22] P. Biswas and P. Robinson, Simulation to Predict Performance of Assistive Interfaces, Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS'07) (To appear)

[23] P. Biswas et. al., User Model To Design Adaptable Interfaces For Motor-Impaired Users, In Proceedings of the Tencon '05 – IEEE Region 10 Conferences, Melbourne, Australia, 2005, 1801-1806

[24] P. E. Bravo et. al., A study of the application of Fitts' Law to selected cerebral palsy adults, Perceptual and Motor Skills (1993), 77, 1107-1117

[25] P. H. McCrea and J. J. Eng, Consequences of increased neuromotor noise for reaching movements in persons with stroke, Journal of Experimental Brain Research (2005), 162, 70-77

[26] P.M. Fitts, The information capacity of the human motor system in controlling the amplitude of movement, Journal of Experimental Psychology (1954), 47, 381-391

[27] S. J. Luck et. al., Neural Mechanisms of Spatial Selective Attention In Areas V1, V2, And V4 of Macaque Visual Cortex, Journal of Neurophysiology (1997), vol. 77, 24-42

[28] S. Keates and J. Clarkson, Countering Design Exclusion An Introduction To Inclusive Design, Springer-Verlag London Ltd., UK., 2004

[29] S. Keates, and S. Trewin, Effect of age and Parkinson's disease on cursor positioning using a mouse. In Proceedings of 7th International ACM/SIGACCESS Conference on Computers and Accessibility, Baltimore, MD, USA, October 2005.

[30] S. Keates, J. Clarkson and P. Robinson, Investigating the Applicability of User Models for Motion Impaired Users, In Proceedings ASSETS 2000, ACM/SIGACCESS Conference on Computers and Accessibility, November 13-15, 2000

[31] S. Keates, S. Trewin, and J. Paradise, Using pointing devices: Quantifying differences across user groups. In Proceedings of UAHCI 2005: 3rd International Conference on Universal Access in Human-Computer Interaction, Las Vegas, USA, July 2005.

[32] S. Trewin and H. Pain, Keyboard and mouse errors due to motor disabilities. International Journal of Human-Computer Studies 50(2), (1999), 109-144.

[33] S.M. Ross, Probability Models For Computer Science, Elsevier, 2002

[34] W. W. Mcmillan , Computing For Users With Special Needs And Models of Computer-Human Interaction, In Proceedings of the ACM/SIGCHI Conference On Human Factors In Computing Systems (1992), 143-148

[35] V.A. B. Scholtes et. al. , Clinical assessment of spasticity in children with cerebral palsy: a critical review of available instruments, Developmental Medicine and Child Neurology (2006), 48, 64-73