

Project Proposal—KLM Generator

Trenton Schulz

August 16, 2009

1 Introduction

1.1 Description

One of the models for predicting user performance is the Goals, Operators, Methods, Selection (GOMS) model. Since its introduction by Card, Newell, and Moran (1983), it has been shown to predict performance close to the actual time for expert users of a system. Several different versions of the GOMS model have been developed to cover various different aspects of human-computer interaction.

One of the simpler models to approach is the Keystroke-Level Model (KLM) that was introduced before back in 1980 by Card, Moran, and Newell (1980) before the other GOMS models. It has just operators for dealing with keystrokes, using the mouse, moving between the devices, and heuristics for applying “mental operations.” While the model is simplified in that you do not need to worry about goals, methods, or selections, it still is a long process to create a model. Most of the work is done “by hand” with the aid of a spreadsheet. Even then, mistakes are likely to happen with missed operators or misapplying a mental operator. All of this work can be very time consuming and this may also be a problem to fit into the lifetime of a product’s creation.

Since the computer receives the majority of the operators (key presses and mouse movements), can detect others (changing from the mouse to the keyboard) and can apply the mental operator heuristics easily, it seems like it would be a good idea to try and automate as much of the KLM generation as possible. The hope is that creating a tool that will allow for quick KLM generation will empower developers and designers to evaluate their applications’ KLM models, experiment, and build more usable interfaces. This should in turn benefit users of these applications.

1.2 Review of Literature

As mentioned above, the original model is presented by Card et al. (1980). They introduce the entire model and test it against several types of text editors, painting programs, and command-line programs available at Xerox PARC. After pouring through the data, they find that the root mean square is 21% for each individual task, but around 5–6%

when comparing them as a group. This work is again reviewed in their book *The Psychology of Human-Computer Interaction* (Card et al., 1983, Chapter 8). These works form the basis for all further work in GOMS and the KLM.

Since the initial model has been introduced, many articles have been done validating GOMS models. For example, Gong and Kieras (1994), use GOMS in the redesign of a CAD program to gain increased efficiency and less learning time. The article also points out that some corrections to the model needed to be made to make the model more accurate. These changes and others have resulted in several other GOMS models being developed since the original model. There are now extra models that deal with concurrency issues, and another works with learning a new system. Bonnie John (1995) along with Kieras (John and Kieras, 1996b,a) review many of the articles that had been written at the time. They also review the various GOMS models that had emerged at that time. All the studies reviewed show that GOMS in all its flavors work to predict performance and improve the design of a product.

Even after that time GOMS has continued to be useful and has even been able to branch out into other areas. For example, Santoro et al. (2004) use GOMS to simulate a naval control room. Kieras and Santoro (2004) presents some interesting lessons learned from this experience and suggestions to modifications to the GOMS model. Luo and John (2005), use the keystroke level model on handheld devices like a Palm Pilot. This work is further expanded by Teo and John (2006). Hinckley et al. (2006) use a keystroke level model to prove that a new interface, the springboard, is more efficient and usable than current interface paradigms. By using a modified version of GOMS, Tonn-Eichstädt (2006), is able to evaluate web pages for blind users. Though he does point out that, “Finally, the verified model has to be built into an automatic tool to facilitate analysis for designers and developers.” (Tonn-Eichstädt, 2006, page 62)

That is not to say that there have not been attempts to build software that can automatically generate GOMS model. The first is QGOMS, presented by Beard et al. (1996), allows “back-of-the-envelope” models in a “quick and dirty” style. In QGOMS the user builds a tree and fills in all the information for each leaf node. The main idea is that this should speed up the process of creating models and allow for comparisons. Kieras et al. (1995) present a formalized way for specifying GOMS models with a tool called GLEAN. Using GLEAN, they are able to reproduce the results done by Gong and Kieras (1994). Baumeister et al. (2000) review these two tools and another called CATCHI and find that while each provides some interesting ways of automating the process, they all have some shortcomings. Hudson et al. (1999) examine CRITIQUE which can generate keystroke-level models from the subArctic Java toolkit. There are fairly promising results, but as pointed out by John et al. (2004), the tool is not in much use or generally available. This article goes on to put forth several ideas about how one should create a tool for this task. They are (John et al., 2004, page 455):

1. Exploit tools already in widespread use by the UI design and cognitive modeling communities.
2. Connect interface mock-ups to cognitive models so changes in the mock-ups are automatically reflected in the models’ predictions.

3. Avoid the need for learning new programming languages by using WYSIWYG drag-and-drop to construct mock-ups and demonstration to construct models.

The article shows such a tool by linking together several tools: a plug-in for Dreamweaver, Netscape LiveConnect, and ACT-R. Though they do admit this set up is a bit sub-optimal since you must have several applications open at the same time.

This is just a beginning, I still need to take a further look at how things like ACT-R work figure into the whole situation. Along with other advances in modelling, but I think this shows a good beginning.

1.3 Purpose of the Study

The main purpose of this study is to create a KLM model generator for the Qt library. Qt is a framework for working on a variety of cross-platform challenges including GUI, network, SQL, etc. Qt is available in both a commercial and open source license and is the underlying library of KDE, a very large open source desktop environment for Linux and other Unix-like operating systems.

Part of the reason that GOMS and the KLM appeal to me is the fact that they provide a way of getting a number that can be used in comparing various attributes of an interface. This may be from coming a math and engineering background where quantitative data is typically being used for most things, such as the number of unit test failures, how long a function, etc.. Of course, as pointed out by John (John, 1995), you can also use several of the GOMS models for qualitative data as well, but I will focus primarily on the KLM model, which is pretty much only quantitative data.

Despite the fact that being able to generate these numbers and that this can aid in user interface design, I've found that it is not known by many people outside of the academic community. I first encountered this when reading Raskin (2000) and really was amazed that, combined with efficiency, one could determine how good their interface was. I mentioned GOMS to several of my fellow engineers at work, but they had not heard of it, but were interested in finding out more about it. I considered showing them how to do it on some of our older programs, but it seemed like there were many manual steps that would have make it unappealing to most of the engineers. In some ways it reminded me of profiling code. This seemed to be a lot of work some years ago, though I knew it could provide big pay-offs. Then, I discovered Shark, a tool for profiling on Mac OS X. As detailed on the Apple Developer Website (Apple Developer Connection, 2004), this made profiling as simple as pushing a button. I know that I was much more likely to try profiling things when I found out how easy it was to do. I feel that by making a tool to easily generate a KLM model, it will encourage more people to use GOMS and evaluate interfaces. While I know that there are interaction designers out there, I do know from experience that a fair amount of GUI development still is done by engineers. The other advantage is that combining this with Qt will allow many open source developers to use the tool, tools like this are missing in the open source community or even commercial development. Also, since Qt has its own window and dialog designer that allows one to create forms without any programming knowledge, people other than developers can test

out interfaces as well. It should empower everyone. Qt's cross-platform nature should ensure that any platforms would benefit. It may even work out of the box for embedded devices such as PDAs or mobile phones.

1.4 Research Questions

Some of the questions that could be researched is whether or not a tool is helpful in increasing awareness of GOMS? Can people who know and can do GOMS use the tool? Also, can people who do not have training in GOMS can successfully use the data presented to inform their decisions in design? Will it cause people to use the method more often? And finally, does a tool actually help or must everything be done "by hand?"

2 Research Design and Methods

The first part of this project would be in creating a generator for Qt. The generator should be able to be added to existing applications or applications under development with little or modification to them. It should also be easy to use so that it encourages its use. Ideally, it should be as easy to use as profiling with Shark. The generator should present the data in a couple of formats and also the data to be exported to be used in other situations such as reports or printing.

Once the generator has been written and tested. It would interesting to see how well it would work in the toolbox of the engineers here at work in Trolltech. There are several ongoing projects that involve making GUI applications that are used by others and it may be helpful to try the tool. This would involve informing developers how to use the tool and later checking up on how the tool worked with their designs and their general impressions of it. This check up could be done in the form of interviews or a survey/questionnaire. Since, I have taken a qualitative methods course and had some fair experience interviewing people now and much less designing questionnaires, I may decide on the former. On the other hand, it may be a lot more quantitative data that I'm looking at the end of the period.

3 Work with Analysis

For writing the generator, not much needs to be done in "normal" analysis, other than getting the thing written and working.

As for the second phase, it really depends on what is chosen as the method for gathering data. If I go a qualitative route, I may need to go through the interviews several times to get all the themes and ideas out. If I am going for quantitative data, I can easily put that together in a table or chart as part of the final part of the thesis.

4 Ethical Considerations

Since I will eventually be working with other people and getting their feelings and opinions. I will need to have some sort of informed consent form for them. I already am an employee of Trolltech, so I probably do not need to sign another non-disclosure agreement with them, but depending on what is presented, I may need to check with the legal department.

Another question is the software itself. Ideally, I would like to make the software as open source, so that others can benefit from it or build on it. My contract with Trolltech gives me quite some leeway on this as long as it is not done on company time. I do know that the University of Oslo may also have feelings about what should be done with the software. I do not anticipate this to be a big problem, but it is probably worthwhile to mention it.

References

- Apple Developer Connection. Optimizing with shark: Big payoff, small effort [online]. (2004) [cited 2006/12/12]. Available from World Wide Web: http://developer.apple.com/tools/shark_optimize.html.
- Baumeister, L. K., John, B. E., and Byrne, M. D. (2000). A comparison of tools for building GOMS models. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 502–509, New York, NY, USA. ACM Press.
- Beard, D. V., Smith, D. K., and Denelsbeck, K. M. (1996). QGOMS: A direct-manipulation tool for simple GOMS models. In *CHI '96: Conference companion on Human factors in computing systems*, pages 25–26, New York, NY, USA. ACM Press.
- Card, S. K., Moran, T. P., and Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. *Commun. ACM*, 23(7):396–410.
- Card, S. K., Newell, A., and Moran, T. P. (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.
- Gong, R. and Kieras, D. (1994). A validation of the GOMS model methodology in the development of a specialized, commercial software application. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 351–357, New York, NY, USA. ACM Press.
- Hinckley, K., Guimbretiere, F., Baudisch, P., Sarin, R., Agrawala, M., and Cutrell, E. (2006). The springboard: multiple modes in one spring-loaded control. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 181–190, New York, NY, USA. ACM Press.
- Hudson, S. E., John, B. E., Knudsen, K., and Byrne, M. D. (1999). A tool for creating predictive performance models from user interface demonstrations. In *UIST '99: Pro-*

- ceedings of the 12th annual ACM symposium on User interface software and technology*, pages 93–102, New York, NY, USA. ACM Press.
- John, B. (1995). Why GOMS? *interactions*, 2(4):80–89.
- John, B. E. and Kieras, D. E. (1996a). The GOMS family of user interface analysis techniques: comparison and contrast. *ACM Trans. Comput.-Hum. Interact.*, 3(4):320–351.
- John, B. E. and Kieras, D. E. (1996b). Using GOMS for user interface design and evaluation: Which technique? *ACM Trans. Comput.-Hum. Interact.*, 3(4):287–319.
- John, B. E., Prevas, K., Salvucci, D. D., and Koedinger, K. (2004). Predictive human performance modeling made easy. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 455–462, New York, NY, USA. ACM Press.
- Kieras, D. E. and Santoro, T. P. (2004). Computational GOMS modeling of a complex team task: Lessons learned. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 97–104, New York, NY, USA. ACM Press.
- Kieras, D. E., Wood, S. D., Abotel, K., and Hornof, A. (1995). GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs. In *UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology*, pages 91–100, New York, NY, USA. ACM Press.
- Luo, L. and John, B. E. (2005). Predicting task execution time on handheld devices using the Keystroke-Level Model. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1605–1608, New York, NY, USA. ACM Press.
- Raskin, J. (2000). *The Humane Interface: New Directions for Designing Interactive Systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.
- Santoro, T. P., Kieras, D. E., and Pharmer, J. A. (2004). Verification and validation of latency and workload predictions for a team of humans by a team of computational models. *U.S. Navy Journal of Underwater Acoustics, Special Issue on Modeling and Simulation*.
- Teo, L. and John, B. E. (2006). Comparisons of keystroke-level model predictions to observed data. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 1421–1426, New York, NY, USA. ACM Press.
- Tonn-Eichstädt, H. (2006). Measuring website usability for visually impaired people—a modified GOMS analysis. In *Assets '06: Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, pages 55–62, New York, NY, USA. ACM Press.