Estimating Productivity: Composite Operators for Keystroke Level Modeling

Jeff Sauro

Oracle, 1 Technology Way, Denver, CO 80237, jeff@measuringusability.com

Abstract. Task time is a measure of productivity in an interface. Keystroke Level Modeling (KLM) can predict experienced user task time to within 10 to 30% of actual times. One of the biggest constraints to implementing KLM is the tedious aspect of estimating the low-level motor and cognitive actions of the users.

The method proposed here combines common actions in applications into high-level operators (composite operators) that represent the average error-free time (e.g. to click on a button, select from a drop-down, type into a text-box). The combined operators dramatically reduce the amount of time and error in building an estimate of productivity. An empirical test of 26 users across two enterprise web-applications found this method to estimate the mean observed time to within 10%. The composite operators lend themselves to use by designers and product developers early in development without the need for different prototyping environments or tedious calculations.

Introduction

Measuring User Productivity

Measuring productivity with an interface is a key aspect of understanding how changes impact its ease of use. One measure of productivity is the time saved by a more efficient design, that is, a design with a task flow requiring fewer steps. Time saved over repetitions of a task, as a measure of productivity, is a key aspect to calculating return on investment (ROI). Productivity metrics are often needed well before there is a working product or any existing users (esp. when the product is new). Such constraints make gathering empirical measures of productivity from a summative usability test difficult and untimely. The usual process for obtaining time on task data involves recruiting then testing actual users in a lab or remote test setup. This procedure while providing a wealth of informative data can be expensive, time-consuming and requires a working version of the tested product.

As a large software organization, we have dozens of products with hundreds of distinct application areas. There is a large demand for benchmarking and improving

the time to complete tasks for mostly productivity-based software such as expense reports, call center applications, etc. Conducting summative usability tests with the main goal to record benchmark task-time data is a herculean undertaking that takes resources away from formative designs. Our challenge was to derive a more reliable way to estimate time-on-task benchmarks and to inform designers about the productivity of their designs as early as possible during product development.

Cognitive Modeling

Rather than observing and measuring actual users completing tasks, another approach for estimating productivity is cognitive modeling. Cognitive modeling is an analytic technique (as opposed to the empirical technique of usability testing). It estimates the task completion time from generalized estimates of the low-level motor operations. Breaking up the task that a user performs into millisecond level operations permits the estimation of task completion times for experienced users completing error-free trials.

The most familiar of these cognitive modeling techniques is GOMS (Goals, Operators, Methods and Selection Rules), first described in the 1970s in research conducted at Xerox Parc and Carnegie-Mellon and documented in the still highly referenced text *The Psychology of Human Computer Interaction*, by Card, Moran and Newell (1983)[1]. GOMS itself represents a family of techniques, the most familiar of which is Keystroke Level Modeling (KLM).

In its simplest form, a usability analyst can estimate user actions using KLM with only a few operators (pointing, clicking, typing and thinking)—see [2] p.72 for a simple introduction. KLM, probably because of its simplicity, has enjoyed the most usage by practitioners. It has been shown to estimate error free time task completion time to within 10 to 30% of actual times. These estimates can be made from either live working products or prototypes. It has been tested on many applications and domains such as maps, PDAs, and database applications [3], [4],[5], [6],[7],[8] and [9].

One major disadvantage of KLM is the tedious nature of estimating time at the millisecond level. Even tasks which take a user only two to three minutes to complete are composed of several hundred operators. One must remain vigilant in making these estimates. Changes are inevitable and errors arise from forgetting operations (Bonnie John, personal communication, October 12th, 2008). In our experience, two to three minute tasks took around an hour to two hours to create the initial model in Excel, then an additional hour in making changes.

Software to Model KLM Operators: Cog-Tool

A better way of building the estimates comes from a software-tool called Cog-Tool, built and maintained at Carnegie Mellon [10] Cog-Tool itself is the results of dissatisfaction with manual GOMS estimating [7]. Cog-Tool is free to download and after some familiarity can be a powerful and certainly more accurate cognitive modeling tool than hand-tracked estimates. Cog-Tool builds the task time estimates by having the analyst provide screen-shots or graphics from the application and then define each object the users interact with (e.g., a button, a drop-down list, etc.). There is a bit of overhead in defining all the objects and defining the sequence of steps the users take during a task. Once completed, however, Cog-Tool provides an easy way to get updated estimates on the productivity of a task. User-interface designers can actually do the prototyping within Cog-Tool and this in-fact exploits the functionality since changes made within the prototyping environment will immediately lead to a new task-time estimate. If prototyping is done in another environment (which it is in our organization) then the analyst will need to import, define and update the objects and task-flows for each change made.

Consolidating the Operators

Our organization has a rather complicated infrastructure of prototyping tools for designers so shifting our prototyping efforts into CogTool, while possible, would be a large undertaking surely met with resistance. We wanted a method to create estimates using KLM like Cog-Tool, that automated the tedious estimation process. We also wanted to allow designers to generate prototypes in whatever environment they preferred. Many requests for productivity come from the Marketing and Strategy teams who can use this information to support sales. We also wanted a method by which we could allow product managers and product strategists to generate their own estimates with little involvement from the usability team.

Looking to Industrial Engineering

Some of the inspiration for GOMS (see [1], p. 274) came from work-measurement systems in Industrial Engineering which began in the early 1900s (e.g., Fredrick Taylor) and evolved into systems like MTM (Methods Time Management see [11]). Just like GOMS, these systems decompose work into smaller units and use standardized times based on detailed studies. These estimating systems evolved (MTM-2, MTM-C, MTM-V, etc.) to reflect the different domains of work and more sophisticated estimates. Generating task-times with these systems, while accurate, are often time consuming. A modification was proposed by Zandin [12] called the Maynard Operation Sequence Technique (MOST). MOST, also based on the MTM system, uses larger blocks of fundamental motions. Using MOST, analysts can create estimates five times faster than MTM without loss of accuracy [13].

Similar to the MOST technique, we wanted to describe user-actions at a higher level of work. Instead of building estimates at the level of hand-motions and mouse

clicks, we wanted to estimate at the level of drop-down selections and button clicks. Each of these operations is still composed of the granular Card, Moran, and Newell operators, but the low-level details which caused the errors and were time consuming could be concealed from analysts.

Method

To refine the KLM technique to a higher level of abstraction we first wanted to see if these higher-level composite operators could predict task times as well as the low-level operators. We used the following approach:

1. **KLM Estimation**: Estimate task times using the KLM technique with low level operators for a sequence of tasks.
2. **Generate Composite Operators**: Generate an estimate of the task times for the same tasks using the composite operators by identifying larger operational functions.
3. **Empirically Validate**: Validate the new composite operators by testing users completing the same tasks repeatedly.
4. **Refine Estimates**: Use empirical data to refine composite estimates (such as updating the system response time) and modify the mental operators to account for concurrent processing.

KLM Estimation

Using the method defined in [1] and [5], we estimated the times. For example, the operators for the initial operations of the task "Create an Expense Report" are:

1. M: Mental Operation: User Decides where to click (1.350s)
2. H: Home: User moves hand to Mouse (.350s)
3. P: Point: User locates the create expense report link target (1.1s)
4. K: Key: User clicks on the link (.25s)
5. R: System Response time as New Page Loads (.75s)

The system response time was updated based on taking some samples from the applications.

Generate Composite Operators

Using the granular steps from above, the logical composite operator is clicking on a link, so the five steps above are replaced with: Click on Link/Button. The time to complete this operation is modeled as 1.350 + .350 + 1.1 +.250 +.75 = approximately 3.8 seconds. This process was repeated for all steps in the 10 tasks.

While not a complete list, we found that a small number of composite operators was able to account for almost all user actions in the 10 tasks across the two web applications. The most commonly used actions are listed below:

1. Click a Link/ Button
2. Typing Text in a Text Field
3. Pull-Down List (No Page Load)
4. Pull-Down List (Page Load)
5. Date-Picker
6. Cut & Paste (Keyboard)
7. Scrolling
8. Select a Radio/Button
9. Select a Check-Box

Empirical Validation

We tested 26 users on two enterprise web-based applications (hereafter Product O and Product P). The products were two released versions of a similar travel and expense reporting application allowing users to perform the same five tasks. The participants regularly submit reports for travel and expenses and were experienced computer users. Ten of the participants had never used either of the applications, while 16 of them had used both.

To reduce the learning time and to provide a more stable estimate of each operator, each participant was shown a slide slow demonstration of how to perform each task. This also dictated the path the user should take through the software. They then attempted the task.

The participants were not asked to think out loud. They were told that we would be recording their task times, but that they should not hurry – rather to work at a steady pace as they would creating reports at work. If they made an error on a task, we asked them to repeat the task immediately. To minimize carry-over effects we counter-balanced the application and task order. We had each participant attempt the five tasks three times on both systems. The training was only showed to them prior to their first attempt. From the 30 task attempts (5*2*3=30) we had hundreds of opportunities to measure the time users took to complete the dozens of buttons, links, drop-downs and typing in text-boxes. These applications were selected because they appeared to provide a range of usable and unusable tasks and exposed the user to most of the interface objects they'd likely encounter in a web-application.

The goal of this test setup was to mimic the verification methods Card, Moran, and Newell did in generating their granular estimates. They, however, had users perform actions hundreds of times. Comparatively, our estimates were more crudely defined. We intended to test the feasibility of this concept and were most interested in the final estimate of the task-time as a metric for the accuracy of the model.

Concurrent Validation

When estimating with KLM one typically does not have access to user data on the tasks being estimated. It is necessary to make assumptions about the system response time and the amount of parallel processing a user does while executing a sequence of actions. System response time understandably will vary by system and is affected by many factors. Substituting a reasonable estimate is usually sufficient for estimating productivity.

In estimating parallel processing, there are some general heuristics ([2], p. 77) but these will also vary with the system. For example, as a user becomes more proficient with a task they are able to decide where to click and move the mouse simultaneously. The result is the time spent on mental operators are reduced or removed entirely from estimate. In the absence of data, one uses the best estimate or the heuristics.

Because our goal was to match the time of users and we had access to the system, we needed to refine the operators with better estimates of actual system response time and of the parallel processing. To do so, we measured to the hundred of a second the time it took users to complete the composite operations (e.g., clicking a button, selecting from a pull-down list) as well as waiting for the system to respond. We adjusted the composite operators' total time by reducing the time spent on mental operation; in some cases eliminating them entirely (see also [14], for a discussion of this approach). The final empirically refined estimates appear in Table 1 below.

Composite Operator	Refined Time (seconds)
Click a Link/ Button	3.73
Pull-Down List (No Page Load)	3.04
Pull-Down List (Page Load)	3.96
Date-Picker	6.81
Cut & Paste (Keyboard)	4.51
Typing Text in a Text Field	2.32
Scrolling	3.96

Table 1. Composite Operators and the refined time from user times.

Some of the operators need explanation. The Date-Picker operator will vary depending on the way the dates are presented. The Cut & Paste Keyboard option includes the time for a user to highlight the text, select CTRL-C, home-in on the new location and paste (CTRL-V). The estimate would be different if using context menus or the web-browser menu. Typing Text in a Text Field only represents the overhead of homing in on a text-field, placing the curser in the text-field and moving the hands to the key-board. The total time is based on the length and type of characters entered (230msec each). Finally, the refined times above contain a system response time which will vary with each system. That is, it is unlikely that clicking of a button and waiting for the next page to display will always take 3.73 seconds. Future research will address the universality of these estimates across more applications.

Results & Discussion

Table 2 below shows the results of the KLM estimates using the "classic" Card Moran and Newell operators and the new composite operators for all 10 tasks. Both the number of operators used and the total task times are shown.

		Classic KLM		Composite KLM	
Product	**Task**	**# of Operators**	**Time (sec)**	**# of Operators**	**Time (sec)**
O	Create Meeting Rprt	81	62	23	98
O	Update a Saved Rprt	51	52	21	46
O	Edit User Preference	43	26	15	35
O	Find an Approved Rprt.	32	18	6	26
O	Create Customer Visit Rprt	149	88	32	55
P	Create Meeting Rprt	169	134	36	156
P	Update a Saved Rprt	93	74	21	82
P	Edit User Preference	65	46	13	60
P	Find an Approved Rprt	48	31	11	43
P	Create Customer Visit Rprt	131	118	23	111
	Mean	86.2	64.9	20.1	71.2
	SD	48.1	38.9	9.3	40.5

Table 2. Comparison between Classic KLM Composite KLM Time & Operators

The data in Table 2 show there to be a difference of six seconds between the composite and classic KLM estimates of the mean task completion time but this difference is not significant [t (17) = .727 p >.7]. The correlation in task time estimates between the two systems is strong and significant (r =.891 p <.01). The average number of operators used per task differed substantially—66 (86.2 vs 20.1) representing a 75% reduction. This difference was significant [t (9) = 4.27 p <.01]. This reduction in the number of operators per task suggests estimates can be made 4 times faster using composites operators.

Do the applications differ in their Composite KLM times?

Next we used the composite operators to estimate which product had better productivity (allowed users to complete the tasks faster) as this would be one of the primary-aims of estimating productivity. Table 3 shows the average of the KLM times for the sum of the operations for the five tasks for both applications.

Task	Product P (Secs.)	Product 0 (Secs.)	Diff. (Secs)	% Diff.
Create Meeting Report	156	98	58	37
Update a Saved Report	82	46	36	44
Edit User Preference	60	35	25	42
Find an Approved Report	43	26	17	40
Create Customer Visit Report	111	55	56	50
Average	**90**	**52**	**38**	**42**

Table 3: KLM Composite Estimates between applications.

Table 3 above shows the KLM composite estimates to predict Product O to be approximately 42% more productive (90-52)/90 than Product P. To validate these estimates we used the 3rd error-free completed task from each user for the empirical estimates. Table 4 below shows the mean and standard deviations for both products.

#	Task	Prod. P (*SD*)	Prod. O (*SD*)	Diff.	n	% Diff.	t	p-value
1	Create Meeting Rpt	157 (24)	105 (14)	52	16	33	9.5	<.001
2	Update a Saved Rpt	81 (19)	54 (9)	26	13	32	6.1	<.001
3	Edit User Preference	52 (13)	34 (6)	18	15	35	5.0	<.001
4	Find an Approved Rpt	38 (10)	33 (11)	5	18	13	1.6	>.12
5	Create Cust. Visit Rpt	123 (19)	61 (14)	62	15	50	11.9	<.001
	Ave	**89 (49)**	**57 (29)**	**32**	**15**	**36**		

Table 4: Mean Task Times in Seconds for All Participants for Their Last Trial (Completed & Error Free Attempts Only)

The third error free trial data shows the Product O application to be approximately 36% more productive (89-57)/89 than Product P. This difference represents an error of 14% (.42-.36)/.42. The estimates of 89 seconds and 57 seconds represent errors of 1% and 9% respectively.

In assessing the accuracy of these estimates we are using the mean time from a set of users, which is in itself an estimate of an unknown population mean time of all users. There is therefore error around our estimate, which varies depending on the standard deviation and sample size of the task (just as in any usability test which uses a sample of users to estimate the unknown mean time). Some tasks have fewer users since not all of the 26 users were able to complete the third task on both systems

without error. The means and 95% confidence intervals around the empirical estimates are shown in Figure 1 below. Also on the graph are the predicted KLM estimates using both the classic and composite methods.

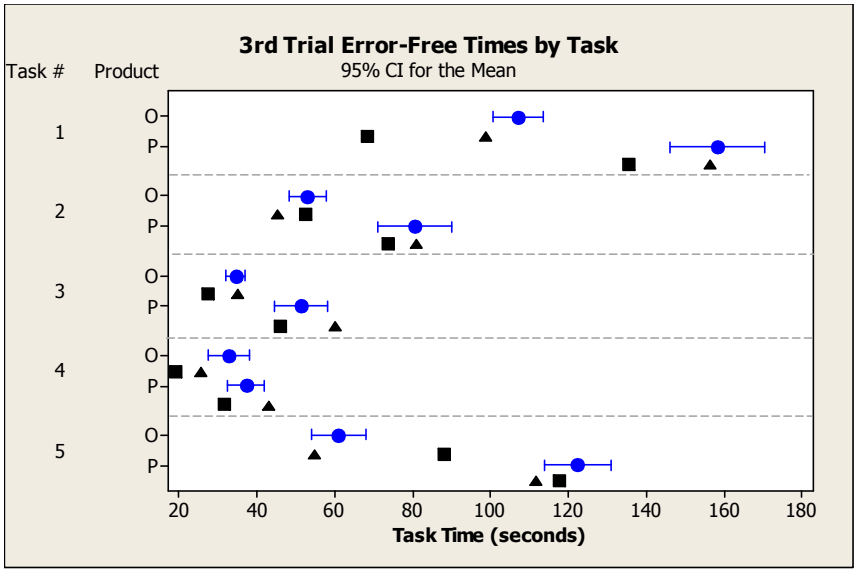

Fig. 1. Means and 95% Confidence Intervals for 3rd Error-Free Trial by Task and Product (blue circles and error bars). The black triangles are the Composite estimates and the black squares are the granular "classic" estimates.

Figure 1 shows visually the variability in the users' mean times. When the KLM estimates are within the range of the blue error bars, there is not a significant difference between the KLM estimate and the likely population mean time. For example, both KLM estimates are not as accurate on Task 1 (especially the Classic KLM estimate) as both estimates are outside the range of the error-bars. On task 2 the estimates are more accurate as three out of the four KLM estimates are within the likely range of the actual user time.

Limitations

While the refined times of the operators displayed in Table 1 above estimated our total task time well, actual composite times will vary with each system. A major factor in each composite operator is the system response time. For desktop applications there might be little if any latency compared to the typical network delays one gets with a web-application. For each system, an analyst should define the composite operators, which would likely include many of the ones defined here.

Conclusion

The data from this initial exploration into combining the granular operators into composite operators shows KLM estimates can be made four-times faster with no loss in accuracy. The estimates made with the composite KLM operators are within 10% of the observed mean time of error free tasks. Composite task times were not significantly different than those from classic KLM estimates ($p > .7$) and task level times correlated strongly ($r=.89$). While the composite operators and their times will vary based on the interface, the method of combining low-level operators into a higher-grain of analysis shows promise. When productivity measures need to be taken and cognitive modeling is used as a more efficient alternative, using composite operators similar to those defined here show promise for faster and more approachable than millisecond level KLM estimates.

References

1. Card, S., Moran, T., and Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.
2. Raskin, Jef *The Humane Interface* Addison Wesley Pres. 2000
3. Baskin, J. D. and John, B. E. 1998. Comparison of GOMS analysis methods. In *CHI 98 Conference Summary on Human Factors in Computing Systems* (Los Angeles, California, United States, April 18 - 23, 1998). CHI '98. ACM, New York, NY, 261-262
4. John, B. (1995). Why GOMS?. *Interactions* 2, 4, 80-89.
5. Olson, J. R. and Olson, G. M. 1990. The growth of cognitive modeling in human-computer interaction since GOMS. *Hum.-Comput. Interact.* 5, 2 (Jun. 1990), 221-265.
6. Gray, W.D., John, B.E., & Atwood, M.E. (1993). Project Ernestine: A validation of GOMS for prediction and explanation of real world task performance.'' *Human–Computer Interaction*, 8, 3, 207-209.
7. John, B., Prevas, K., Salvucci, D., & Koedinger, K. (2004) Predictive Human Performance Modeling Made Easy. Proceedings of CHI, 2004 (Vienna, Austria, April 24-29, 2004) ACM, New York.
8. Gong, R. and Kieras, D. (1994). A validation of the GOMS model methodology in the development of a specialized, commercial software application. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* Boston, Massachusetts. CHII '94, ACM, New York, NY, 351-357
9. Haunold, P. and Kuhn, W. (1994). A keystroke level analysis of a graphics application: manual map digitizing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* Boston, Massachusetts. CHII '94, ACM, New York, NY, 337-343.
10. John, B (2009) The Cog-Tool Project http://www.cs.cmu.edu/~bej/cogtool/ accessed Jan. 2009.
11. Maynard, H, G. Stegemerten and Schwab, J *Methods Time Measurement* New York McGraw-Hill, 1948
12. Zandin, Kjell MOST Work Measurement Systems, New York: Marcel Dekker 1980.
13. Niebel and Freibalds *Methods, Standards, and Work Design* McGraw-Hill 2004
14. Mayhew, D. (2004), Keystroke Level Modeling as a Cost-Justification Tool. Chapter appearing in Bias & Mayhew (Eds.) *Cost-Justifying Usability* , 2nd Ed. 465-488