VIA:

Using GOMS to Improve Authorware for a Virtual Internship Environment

Arthur C. Graesser
University of Memphis, USA

Tristan M. Nixon
University of Memphis, USA

Andrew J. Hampton
University of Memphis, USA

Sam E. Franklin
University of Memphis, USA

Jeneé B. Love
University of Memphis, USA

ABSTRACT

This chapter describes the testing of the computer-human interface of Virtual Internship Authorware (VIA), an authoring tool for creating web-based virtual internships. The authors describe several benchmark tasks that would be performed by authors who create lessons on the subject matter of land science. Performance on each task was measured by task completion times and the likelihood of completing the task. Data were collected from ten novices and three experts familiar with the broader learning environment called Intershipinator. Task completion times and the number of steps to complete the tasks were also modeled by GOMS (Goals, Operators, Methods, and Selection Rules), a theoretical model that predicts these measures of user interaction based on a computational psychological model of computer-human interaction. The output from the GOMS simulations of task completion times and number of steps robustly predicted the performance of both novices and experts. Large deviations between model predictions and human performance are expected to guide modifications of the authoring tool.

Keywords: Authoring Tool, Human-Computer Interaction, Learning Environments, Learning Strategies, Teaching Software, Virtual Learning, Virtual Internship Authorware

INTRODUCTION

One of the major bottlenecks in developing learning environments is designing authoring tools for subject matter experts to create content and recommended learning strategies (Murray, Blessing, & Ainsworth, 2003; Sottilare, Graesser, Hu, & Brawner, 2015). In an ideal world, the *authorware* would be so easy to use that the learning materials, curriculum, and pedagogical

strategies could be created by a wide range of subject matter experts. Moreover, the experts who use the authorware would be able to create lessons even when they have modest expertise in digital literacy, minimal computer programming experience, and no more than intuitive knowledge of the science of learning. They would be able to assemble impressive learning environments that integrate text, diagrams, video, chat facilities, conversational agents, interactive simulations, and other materials that may already exist in different media, modalities, formats, and types of interaction.

Unfortunately, that ideal remains elusive to developers of most advanced learning environments. Instead, subject matter experts typically provide content to computer programmers and pedagogical experts who end up being responsible for transforming the various materials into interactive learning environments. Improvements in authorware may only occur after developing a systematic scientific approach to conducting iterative testing and modification of the content, pedagogical strategies, and human-computer interfaces for the end users.

This chapter describes our approach to improving the human-computer interface of the authorware for a virtual internship project called *Internshipinator* (Shaffer, Ruis, & Graesser, 2015). The authorware is called VIA, which stands for *Virtual Internship Authorware*. We use the term authorware in a general sense here, duly recognizing that a software product from Adobe, called Authorware (http://www.adobe.com/products/authorware/), has existed for decades. VIA is a specific type of authorware that developers of learning environments can use to create virtual internships in a web-based learning environment.

Virtual Internship Authorware (VIA)

In one example, a virtual internship called *Land Science* (Bagley & Shaffer, 2015), students play the role of interns at Regional Design Associates, a fictional urban and regional planning firm. Their problem solving task is to prepare a rezoning plan for the city of Lowell, Massachusetts that addresses the requests of various stakeholder groups (business, environment, industry, or housing). The stakeholders have views on socioeconomic and ecological issues, some of which are incompatible. The students read about the different viewpoints and preferences of stakeholders and eventually prepare individual reports on how to handle competing concerns. While making these decisions, students discuss options with their project teams through online chat. They also use professional tools, such as a geographic information system model of Lowell and a preference survey to model the effects of land-use changes with feedback from mentors. At the end of the internship, students write a proposal in which they present and justify their rezoning plans. During this process, a mentor keeps the small group of three to four students moving forward, but does not encourage any particular solution to the problem solving tasks at hand. The ten-hour simulation environment is divided into fourteen rooms with different goals and objectives, as well as a chat capability associated with each room.

Land Science is clearly a complex internship-based learning environment with many different learning resources, communication media, recommended agenda, and instructions. The author of the environment has to create all of this material for the virtual internship, which is quite a challenge. Authors with high expertise can readily meet the challenge, but most authors need more scaffolding and assistance from VIA. Authors with modest abilities can go a long way by taking an existing internship environment and modifying it to create new content and capabilities. At the very least, the human-computer interface and navigation facilities need to be

sufficient for a broad range of users. The design of VIA was created to support authors of varying abilities.

This chapter describes the testing of the human-computer interface of VIA. These tests consisted of benchmark tasks that are performed by authors who modified content on the subject matter of land science. Performance on each task was measured by task completion times and the likelihood of completing the task. Data were collected from ten novices and three experts who were familiar with Intershipinator. We also developed a model of ideal task completion times and the number of steps to complete the tasks with *GOMS (Goals, Operators, Methods, and Selection Rules)*. GOMS is model that predicts these measures of user interaction based on a computational psychological model of human-computer interaction (Card, Moran, & Newell, 1980, 1983; John & Kieras, 2006). The GOMS simulations of task completion times and number of steps were found to predict the performance of both novices and expert authors. The implications of this successful application of GOMS are discussed at the end of the chapter.

Virtual Internships

Internships are a valuable commodity for college students looking for practical experience in a difficult job market. Interns can gain valuable work experience by contributing to companies in ways that go beyond theory taught in classrooms, including managing disparate viewpoints and competing interests. A 2010 study concluded that college students who participated in internships were given more options of full time employment post-graduation (Gault, Leach, & Duey 2010). The Internshipinator project simulates these inevitable challenges that reside outside the purview of the textbook. It does this by providing the internship's author a structured but flexible *frameboard* to populate the environment with content.

The Intershipinator project specifically focuses on internships in STEM fields (Science, Technology, Engineering, and Mathematics). It embraces several key practices in conventional STEM internships. For example, good internships give interns authentic work assignments that represent the actual work in the field. The Land Science internship gives interns tasks and problems meant to closely follow real world land science and urban planning projects. Internships also commonly use intern managers to focus efforts and give foundational information. Internshipinator affords this engagement by including a mentor role. Moreover, collaborative problem solving is known to be a highly useful skill in workplace environments (NRC, 2011; OECD, 2013) so many internships purposely place interns together to test their collaborative skills. The Land Science internship echoes this by providing the interns a series of tasks that require collaborative learning and problem solving to properly find a solution to their assignment. High quality internships also evaluate the progress of interns empirically through intern interviews and assessments of the interns' products. Internshipinator encourages all of these practices with entrance and exit interview options, computer-mediated communication, and products to be evaluated. These capabilities that are present in Internshipinator also need to be incorporated in the VIA tool. As already discussed, VIA was designed to allow a broad range of experts in a particular STEM field to author their own internships with minimal prior computer programming knowledge.

Internships created through VIA need to consider three types of users: (1) the interns who use Internshipinator to learn, (2) the mentors who use Internshipinator to help the interns learn, and (3) the authors who use VIA to create Internshipinator. The author is the subject matter expert who designs the internship. Authors determine how internships progress, what materials to

provide for both mentors and interns, and the environment in which interns operate. Although the authors create this material to facilitate the interns' learning with the assistance of mentors, they often have little or no direct interaction with interns and mentors. Authors need to make sure that the internships they design follow a logical progression and have instructions that direct the users to achieve their respective end goals. For example, authors create helpful pre-generated prompts (i.e., verbal messages in emails or chat) for mentors to use during the process of facilitating interns in their learning. Authors create reference material to suit interns' needs to receive relevant subject matter information in accomplishing tasks. Authors typically construct their internships in VIA by dividing the content structure into a series of modules called *rooms*; interns receive a single topic in each room and usually a task that is evaluated. That is, the material required for the task is attached to a room, including any specific pre-generated prompts that the authors believe would be useful to the mentor in guiding progress in the task. After the author has finished designing Internshipinator with VIA, the mentor administers and monitors Internshipinator through synchronous computer-mediate communication during the process of the interns interacting with Internshipinator.

As mentioned, the mentors manage the internships and directly oversee their assigned interns. A mentor provides guidance and assistance as interns progress through tasks and they evaluate interns based on author specifications. The mentor may assume the role of a project consultant. Alternatively, the mentor may take on the role of a fellow intern and thereby appear to be a peer from the perspective of other interns. These options in roles permits flexibility in the interaction and allow both a cooperative and an authoritative role to direct progress, sometimes in tandem. In Land Science, for example, the mentor portrayed both a project consultant directly working with interns and also the project manager.

The intern is the participant for whom the Internshipinator simulation was built. Interns are guided and assisted by the mentor and complete the tasks set up by the author. Interns are the "players" of this simulation environment. They collaborate with one another and follow the suggestions of the mentor to achieve the proficiency in the subject matter created by the author.

It is apparent that the virtual internships like Land Science are complex learning environments with many facilities and a significant amount of technical content. It would be impractical and restrictively slow for new design teams to build the new internships from scratch. Therefore, a more practical approach is for the authors to modify an existing internship, such as Land Science, by substituting and adding content with the new subject matter. That requires authoring tools such as VIA in addition to the other components of Internshipinator.

The GOMS Model

The GOMS (Goals, Operators, Methods, and Selection Rules) model was developed in the early 1980's for measuring and predicting the performance of skilled, successful human-computer interactions with technology (Card, Moran, & Newell, 1980, 1983). *The Psychology of Human-Computer Interaction* (Card et al., 1983) played a signature role in launching the field of human-computer interaction because it was grounded in (a) a psychological theory of problem solving, namely Newell and Simon's (1972) *General Problem Solver*, (b) a computational model of perception, action, and cognition, and (c) a large body of empirical research in human factors and human performance that had accumulated for decades. Just as important, the model could generate predictions on task completion times and errors for tasks on the basis of the device characteristics together with the model.

Digital system designers frequently use GOMS modeling to predict task execution times and the usability of products (Gray, John, & Atwood, 1993; John & Kieras, 2006; Williams, Hultman, & Graesser, 1998). For example, Drury, Scholtz, and Kieras (2007) used GOMS to analyze human-robot interfaces, Oyewole and Haight (2011) used GOMS to help novice users of website interfaces perform better, and Saitwal et al (2010) used GOMS to analyze the usability of interfaces for electronic health records.

The GOMS model assumes that tasks are organized by a hierarchically organized set of *goals*. An *operator* is a specific action that is triggered by a particular perceptual pattern, performed in a particular way, and accompanied by cognitive processes. More specifically, each operator has one or more perception-cognition-action cycles with time parameters that can be estimated *a priori* based on research in cognitive engineering and human factors. A *method* is an organized set of operators that have achieved particular goals in the past. *Selection rules* specify which method is applied under particular conditions when multiple methods are possible.

GOMS decomposes tasks and complex actions into their component parts and generates predictions on the completion times and steps for each task or action. The decomposition can have different grain sizes. At the most micro-level, for example, the keystroke modeling predicts the execution times of individual operators (Card, Moran, & Newell, 1980). At a mid-level, times can be predicted for performing methods or tasks at an intermediate level of abstraction (Gray, John, & Atwood, 1993). At the macro-level, times can be predicted for an entire complex task.

Cogtool v1.2.2

John (2013) developed a program (*Cogtool v1.2.2*) that allows researchers to generate predictions for the completion of tasks and actions on digital interfaces based on the GOMS model. This program prompts the researcher to specify interface screenshots, embedded widgets to mark operators and their functions, the operators to be performed, and other information that is beyond the scope of this chapter to describe. The sequence of operators chosen for a task are automatically compiled into a script that combines that operator's individual times to compute a predicted time for the execution of the task. It should be noted that GOMS models an experienced user of the interface, not novices. Therefore, the sequence of operators that GOMS models is the smallest critical path to achieve the main goal of the task.

METHOD

Participants

The novice participants were recruited from the University of Memphis pool enrolled in psychology courses, N = 10. Most of the students were freshman taking introductory psychology. The experts were three members of the research team who had experience using Internshipinator for Land Science.

Materials

Participants completed testing in a computer lab. They scrolled through a PowerPoint orientation with audio and text. Testing proceeded on the VIA tool. Cogtool was used to simulate the actions taken by testers of the VIA interface as accurately as possible.

Procedure

As discussed in the next section, the testers were asked to perform a series of benchmark tasks that manipulate various aspects of the interface in order to achieve specific goals. The times collected from the participant testers were correlated with the ideal items for each task, based on Cogtool. The ideal time is the quickest time possible with the fewest number of mouse clicks. That is, the ideal time was computed as Cogtool's estimation of the quickest possible path to complete the task with the interface. The screen shots of the VIA interface were uploaded to Cogtool. Next, "widgets" were attached to the interface objects in a given task to allow for Cogtool to simulate their use. Lastly, we input to Cogtool the correct operations list: the interface objects to manipulate in order to correctly complete the task in the shortest amount of time. Following Saital et al. (2010), we used the minimum number of clicks and execution times as our theoretical predictor variable.

Table 1 shows an example predicted task completion time and a sequence of operators when we used Cogtool to model a task in VIA. The predicted task completion was 28.9 seconds for the task. Each row is a step in the procedure that contains a visual interface frame, the action performed (e.g., move mouse, click, think), and the widget device on the interface. A script step list (as in Table 1) was prepared for each benchmark task in the tests of VIA.

Table 1. Script of Frames, Actions, and Widget Devices in a GOMS Model for VIA

	Script Step List	**Visualization**
Frame	**Action**	**Widget/Device**
Frame 2 [2]	*Move Mouse*	*Upload (Widget 3)*
Frame 2 [2]	Left Click	Upload (Widget 3)
Frame 4	*Think for 1.200 s*	
Frame 4	*Move Mouse*	*ddm (Widget 1)*
Frame 4	Left Click	ddm (Widget 1)
Frame 4	*Think for 1.200 s*	
Frame 4	*Move Mouse*	*Final p (Widget 2)*
Frame 4	Left Click	Final p (Widget 2)
Frame 5	*Move Mouse*	*D:7 (Widget 1)*
Frame 5	Left Click	D:7 (Widget 1)
Frame 6	*Think for 1.200 s*	
Frame 6	*Move Mouse*	*Resource (Widget 1)*
Frame 6	Left Click	Resource (Widget 1)
Frame 7	*Think for 1.200 s*	

Frame 7	*Move Mouse*	*Create resource (Widget 1)*
Frame 7	Left Click	Create resource (Widget 1)
Frame 7	*Home Keyboard*	
Frame 7	Type 'title'	Title (Widget 2)
Frame 7	Type 'link name'	Link name (Widget 3)
Frame 7	Look at	Map (Widget 4)

Benchmark Tasks in test of VIA

Table 2 lists 20 benchmark tasks that the participants performed in our testing of VIA. The tasks were approximately ordered by complexity, with initial tasks orienting the participant to the system. The later modification tasks required changes to the Land Science internship structure that mimic the steps and actions an author would go through during the process of editing an internship. Tasks 3 and 4 are not included in Table 2 because those tasks simply involve reading about information to orient them to the system.

A deep understanding of these tasks would of course require visual depictions of the interface. Unfortunately, the project was in the early phases of iterative development so the quality of the visual depictions was not sufficient to be reproduced in this chapter. It is the later stages of iterative development when attention is typically given to the aesthetics of the font, color, resolution, menu layouts, window borders, and so on.

Some of the tasks had two phases. Phase 1 involved the testers completing tasks without assistance. Phase 2 followed only if they failed phase 1. That is, if they failed to complete phase 1 within 20 seconds, they received phase 2 where they were given hints (including texts, visuals, or screen shots). However, the final task 20 had to be completed without any assistance. The tasks were selected to capture tasks with actions that an author would perform while creating or modifying an internship using the VIA software. The tasks did not require much time to perform individually, with most requiring a relatively low number of clicks and mouse movements.

Table 2. Benchmark Tasks in Tests of VIA

Task Number	**Task Description**
Task 1	Log in to the VIA using provided password and username.
Task 2	Select correct Land Science frameboard for Editing.
Task 5	Select any tab in the Global Panel to reinforce the knowledge of the location of the global panel.
Task 6	Close the tab in the global panel. (This is very important to know because any tab open in the Global Panel hides the Compiled View)

Task 7	Use drop down menu to select RFP (Request for Proposal) room for editing.
Task 8	View the deliverables in the RFP by locating the deliverables tab in the Compiled View.
Task 9	Select the rubric text for editing in the Compiled View. (This task begins the process of emphasizing the connection between the Compiled View and the Dependency View).
Task 10	Edit selected text from last task in the text box in the Dependency View.
Task 11	Locate the list of resources which can be found in the Global Panel.
Task 12	Select the invite email in the final proposal room. (This task requires the tester to once again change rooms and remember that the invite email is located in the Compiled View).
Task 13	View and close feedback in the Global Panel.
Task 14	Change the notebook section type. (This task requires the tester to look deeper in the VIA as they search for the notebook tab in the dependency tree and change the section type).
Task 15	Modify paragraph in invite email. (Like tasks 9 and 10, the tester must click the text in the Compiled View and edit in the Dependency View. However, now the tester is expected to be able to do so without much guidance).
Task 16	Edit notebook section in deliverable 2. (The tester must again dive into the VIA by changing the deliverable that is visible in the dependency tree, locate the notebook section and edit some part of it such as the notebook title).
Task 17	Upload a pdf. (The testers must remember that resources contain all pdf files and are located in the Global Panel. They must then locate the upload tab and select the correct pdf to upload).
Task 18	Add pdf to invite email in deliverable 2. (This requires the tester to drag and drop the pdf into the resource box of deliverable 2 in the dependency tree).
Task 19	Edit necessary text. (The tester has knowledge of how to properly edit text, but must find where the information for resources is mentioned in the rubric and add additional text referring to the newly added pdf).
Task 20	Add a map to deliverable 7 and edit the necessary text in the final proposal room. (This task has no hints because the testers must now use all of their acquired knowledge to complete it).

An overview of the series of tasks conveys the scope and sequence of the benchmark tasks. Tasks 1 and 2 prompt the tester to log into the system proper and access the correct frameboard to edit. Tasks 3 and 4 give details about the user interface and familiarize testers with the names and locations of important functions. Task 5 begins the VIA interface interaction in earnest. It shows the tester how to open and close the Global Panel, a vital part of VIA. From there the tester learns how to navigate between rooms. Tasks 7-16 focus on the association of a "Deliverables Tree/View" in the center of the screen with a "Compiled View" on the right of the screen, which shows a more polished view of content in the Deliverables View (more so what an end user would see). After a quick stop back in the Global Panel to learn about the resources subtab, the testers' attention is drawn toward the Compiled View. These tasks include actions such as locating a particular text in the Invite Email, modifying texts in the rubric, and changing notebook sections in the Dependency Tree. In tasks 17-19, the tester explores the interface widely, from the Global Panel, to the Dependency View, and to the Compiled View. The tester must use their knowledge of navigating the VIA to load files and edit sections appropriately to incorporate instructions for the added information. These tasks emphasize the intimate relationship between the Compiled View and Dependency View areas of the VIA. The final task 20 then attempts to utilize as much of the system as possible, while approximating a realistic task an end user would perform. This task required the most clicks (11) to complete.

RESULTS

We collected data from two groups of testers. The expert group included three members of the research team who were experienced users of Land Science. These three experts completed all 20 tasks successfully. The novice group included ten undergraduate college students who were taking an introductory psychology course and completing the experiment for course credit. Testers in the novice group did not always successfully complete each of the 20 tasks. For these students in the novice group, we computed the number of students out of ten who successfully completed the task as one measure of performance. For all testers, both experts and novices, we computed task completion times as a measure of performance. We compared these performance measures to two theoretical predictions of the GOMS model: GOMS task completion time and GOMS clicks to complete the tasks (i.e., functionally similar to number of steps on the critical path to the solution).

Table 3 shows the above measures for 18 of the 20 tasks in our testing of VIA, noting once again that the reading tasks 3 and 4 were excluded. There are two GOMS measures (time and clicks), two human task completion times (experts versus novices), and the frequency out of ten of novice task completions.

Table 3. GOMS and Tester Performance Measures on VIA Tasks

Task	GOMS Time	GOMS Clicks	Expert Times	Novice Times	Novice Completion
1	10.00	3	38.9	121.2	10
2	10.00	2	26.1	161.9	9

5	10.00	1	12.5	104.4	6
6	10.00	2	11.8	57.0	6
7	12.15	2	13.3	150.1	5
8	10.00	1	14.6	81.2	4
9	14.00	1	11.2	92.9	2
10	14.00	1	26.6	359.7	1
11	4.60	1	12.0	92.8	5
12	12.50	3	18.1	64.8	2
13	9.30	2	13.8	87.3	5
14	5.20	3	16.9	56.2	1
15	25.30	3	33.6	603.3	1
16	8.30	3	66.8	428.0	3
17	15.30	6	29.2	162.0	3
18	12.00	6	80.9	438.0	3
19	11.80	3	59.0	178.7	3
20	28.90	11	141.6	245.0	2
Mean	12.4	3.0	34.8	193.6	3.9
SD	6.0	2.5	33.7	158.8	2.6

A quick analysis of the testers' results and the GOMS model show that as users progressed through the tasks, completion times increased. This reflects the corresponding increase in difficulty. Early tasks had high completion rates, but these numbers went down as novice users encountered tasks that required more clicks to be completed. The novice times should be interpreted with great caution, however, because it is only the completers who are counted. Indeed, the more astute testers may have been the ones who managed to complete the tasks. Nevertheless, we report their time data.

We conducted correlational analyses on the data displayed in Table 3. A correlation of $r = 0.40$ would be statistically significant ($p < .05$) when considering the 18 tasks as a unit of analysis and the mean scores listed in Table 3. We acknowledge the small sample size, but note the promising magnitude of the observed relationships.

Consider first the expert testers, the primary target of the GOMS model because GOMS predicts knowledgeable users of the interface who have experience with the environment. All three

individuals completed the tasks. These task completion times for these experts had a strong correlation with GOMS time ($r = 0.59$) and number of GOMS clicks ($r = 0.87$). Interestingly, the expert times had a significant correlation with novices who were skilled enough to complete the tasks ($r = 0.46$). Therefore, the GOMS model accurately predicts performance (time and clicks) when the testers reach a reasonable threshold of performance. This, once again, is what the GOMS model was developed to explain: performance when the user is sufficiently skilled. In contrast, performance while the user is learning of the interface is not within the scope of GOMS.

Nevertheless, we can ask the question of whether GOMS can predict novice performance. There is a modicum of success, but (as would be expected) not as impressive as the experts. The GOMS time does significantly predict the novices' task completion times ($r = 0.51$) but the GOMS clicks fell just short of significance at the $p < 0.05$ level ($r = 0.39$). The results are still quite promising on the value of GOMS with regard to novice performance. We also analyzed the novice completion likelihoods in the right column and found them to be uniformly in the predicted direction, i.e., more time and complexity negatively correlated with the likelihood of task completion. That is, the task completion likelihoods were negatively correlated with GOMS time ($r = -0.39$) and GOMS clicks ($r = -0.25$).

As we look at the data in Table 3, we note some large deviations between GOMS time predictions and those of novices. For example, novice times are over 25 times longer than GOMS predictions for task 10. These discrepancies may prove particularly informative for the designers of human computer interfaces because they identify areas where the interface is most in need of improvement. Examining the tasks with large differences, we see that students who did not carefully review early tasks explaining panels and their functions did poorly on later tasks such as 10-17 which required them to apply that knowledge. Some students unintentionally skipped early tasks, attributable to the lack of a "Back" button in the testing interface. Utilizing uninitiated participants with no particular computer skills invites mistakes and misunderstandings that would never occur to experts. The GOMS model predictions provide quantifiable markers for these stumbling blocks, helping to diagnose interface problems and guide modifications of authorware in the process of iterative development.

CONCLUSION

It is reassuring that the GOMS model performed quite well in predicting the performance of both expert and novice users of the Virtual Internship Authorware (VIA). The number of GOMS steps, as measured by user clicks, was also promising, although to a lesser extent for novice users in contrast to the near perfect correlations with experts ($r = 0.87$). There of course may be other measures from GOMS that may prove more promising, but we were encouraged by these results on the value of the GOMS model in early phases of our development of VIA.

In our view, it is the deviations from GOMS predictions that are noteworthy for modifications of authorware like VIA. There are problems when the task completion times of VIA experts or novices are substantially higher than those of GOMS. What explains these aberrations? One likely culprit is time spent in cognitive processing rather than physical movement. It may be something as simple as unexpectedly slow reading speed, for which GOMS modeling does not compute by default. Alternatively, the cues on the interface may be hidden, which confuses the author and suggests the need for more salient control features. Perhaps the author is looking at

the wrong place on the display, suggesting a reorganization or co-location of functional objects. Perhaps the author has trouble interpreting a particular label, icon, or symbol, necessitating a redesign more in line with intuitive constructions. Hypotheses (e.g., the downstream implications of not having a "Back" button) can be generated that attempt to explain large deviations from GOMS predictions, a signal of interface pathologies. When the author experiences trouble, the designers can consider various modifications and improvements. Further experimentation is then conducted in A-B testing that compares the old interface with the new interface that attempts to correct the problems. Consequently, both successes and failures of GOMS are helpful in guiding interface design on complex learning environments like VIA.

ACKNOWLEDGEMENTS

This research was funded by the National Science Foundation (DRK-12-0918409, DRK-12 1418288). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

Bagley, E., & Shaffer, D. W. (2015). Learning in an urban and regional planning practicum: The view from educational ethnography. *Journal of Interactive Learning Research*, 26, 369-393.

Card, S, Moran, T. P., & Newell, A. (1980). The keystroke-level model for user performance with interactive systems. *Communications of the ACM*, 23, 396-210.

Card, S., Moran, T.P., & Newell, A. (1983). *The psychology of human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Drury, J. L., & Kieras, D. (2007). *Adopting GOMS to model human-robot interaction*. New York, NY: ACM.

Gault, J., Leach, E., & Duey, M. (2010). Effects of business internships on job marketability: The employer's perspective. *Education + Training* (vol. 52,1).

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, 237-309.

John, B. (2013). Cogtool. (Version 1.2.2) (Software). Available from https://github.com/cogtool/cogtool/releases/tag/1.2.2.

John, B. A. & Kieras, D. E. (2006). Using GOMS for user interface design and evaluation: Which technique?. *ACM Transactions on Computer-Human Interaction,* 3, 287-319.

John, B. A. & Newell, A. (1987). Prediction the time to recall computer commands abbreviations. *Proceedings of the CHI '87 Conference on Human Factors in Computing Systems* (pp. 33-40). New York, NY: ACM.

John, B. A. & Newell, A. (1989). Cumulating the science of HCI: From S-R compatibility to transcription typing. *Proceedings of the CHI '89 Conference on Human Factors in Computer Systems* (pp. 109-114). New York, NY: ACM.

Kieras, D. E. (1988). Towards a practical GOMS model methodology for user interface design. In M. Helander (Ed.), *Handbook of human-computer interaction* (pp. 135-158). Amsterdam: North-Holland.

Murray, T., Blessing, S. & Ainsworth, S. (2003). *Authoring tools for advanced technology learning environments.* Dordrecht: Kluwer Academic Publishers.

National Research Council (2011). *Assessing 21st century skills.* Washington, DC: National Academies Press.

Newell, A., & Simon, H. A. (1972). *Human problem solving* (Vol. 104, No. 9). Englewood Cliffs, NJ: Prentice-Hall.

OECD (2013). *PISA 2015 collaborative problem solving framework*. Paris, France: OECD.

Oyewole, S. A. & Haight, J. M. (2011). Determination of optimal paths to task goals using expert system based on GOMS model. *Computers in Human Behavior*, 27, 823-833.

Saitwal, H., Feng, X., Walji, M., Patel, V., & Zhang, J. (2010). Assessing performance of an electronic health record using cognitive task analysis. *International Journal of Medical Informatics*, 79, 501-506.

Shaffer, D.W., Ruis, A.R., & Graesser, A.C. (2015). Authoring networked learner models in complex domains. In. R. Sottilare, A.C. Graesser, X. Hu, & K. Brawner (Eds.), *Design recommendations for intelligent tutoring systems: Authoring tools* (Vol. 3, pp.179-192). Orlando, FL: Army Research Laboratory.

Sottilare, R., Graesser, A.C., Hu, X., & Brawner, K. (Eds.) (2015). *Design Recommendations for Intelligent Tutoring Systems: Authoring Tools* (Vol. 3). Orlando, FL: Army Research Laboratory.

Weller, M. (2007). *Virtual learning environments: using, choosing and developing your VLE*. London: Routledge. pp. 4–5.

Williams, K. E., Hultman, E., & Graesser, A. C. (1998). CAT: A tool for eliciting knowledge on how to perform procedures. *Behavior Research Methods, Instruments, & Computers*, 30, 565–572.

KEY TERMS AND DEFINITIONS

Authoring Tool: Software to assist authors who create a specific type of program.

GOMS model : "a set of Goals, a set of Operators, a set of Methods for achieving the goals, and a set of Selections rules for choosing among competing methods for goals" (Card, Moran, &

Newell, 1983) that provide qualitative and quantitative predictions how users will interact with a system.

Human-Computer Interaction: A field of research that investigates the functional coordination between human cognitive characteristics and computer software.

Internshipinator: A project that develops and tests virtual internships in science, technology, engineering, and mathematics, with the aim of broadening educational experience.

Learning Environments: A situation (real or virtual) that is designed to help individuals learn.

Learning Strategies: Plans put in place by educators or students to facilitate student learning.

Teaching Software: Software intended to facilitate and enhance learning.

Virtual Internship Authorware: An authoring tool that allows those without coding experience to create and modify virtual internships.

Virtual Learning Environment: A Web-based platform for the digital aspects of courses of study, typically affording participant cohorts and including activities and interactions (Weller, 2007).