Situating GOMS Models Within Complex, Sociotechnical Systems

Robert L. West (robert_west@carleton.ca)
Department of Psychology; Carleton University
Ottawa, Canada

Gabriella Nagy (gnagy@chat.carleton.ca)
Department of Psychology; Carleton University
Ottawa, Canada

Abstract

In this paper we present a methodology for situating GOMS models in complex sociotechnical work domains. The methodology is presented within a larger theoretical framework that relates GOMS modeling to other modeling systems according to principled and systematic guidelines.

Increasingly, computers play critical roles in the running of complex systems such as telecommunications networks and nuclear power plants. However, the role of human agents in these systems is also critical. As computer and software technology improve we see a decrease in the number of technical errors caused by computers, but there is also evidence of a corresponding rise in errors attributable to humans (e.g., Bennett, 1998). No doubt, this is due to the increasing complexity of computers and network systems.

In this paper, we consider the role of GOMS (Card, Moran, & Newell, 1983) in designing systems situated within complex, sociotechnical systems, that is, systems with multiple humans and multiple computers all interacting (see Vicente, 1999 for a more complete definition). GOMS is a method for modeling tasks according to a human agent's goals, operators, methods and selection rules (John, 1995). But in complex sociotechnical systems the task is often a small part of a larger, distributed task. The design problem is analogous to designing a complex operating system. Individual programmers design different components of the system, but each time a new component is added it is unclear if it will create a conflict in the system. Similarly, changing the way an individual performs a task within a complex sociotechnical system can have unforeseen consequences (for a discussion of this point and some interesting examples, see Hutchins, 1995). To deal with this problem operating systems are beta tested. Unfortunately, changes in a sociotechnical system cannot be beta tested and then fixed the next day. In fact, such changes are usually costly and time consuming, especially if people need training. Thus, we need a means to evaluate changes before they are implemented.

Task Analysis

A task analysis is important for understanding the sort of knowledge driven tasks common in technical areas and large organizations. By knowledge driven we mean that the agent knows, implicitly and/or explicitly, the steps that must be completed. The need for a task analysis presupposes that the process for completing the task quickly and without error is not common knowledge. Many studies have found that experts often have specialized knowledge that is not expressed in any manual, but is nevertheless crucial for completing the task in an acceptable manner (Mayer, 1997). This is particularly true of tasks situated in sociotechnical environments, which often involve a considerable amount of undocumented knowledge concerning how the various agents, computers, and artifacts involved are coordinated to complete the task.

The result of a task analysis is a model, which is then used to simulate changes in the system. The level of detail of the model will depend on the modeler's goals, and the representation of the model can range from a mental representation, to a paper and pencil representation, to a computerized representation. Furthermore, the goal may be to represent the whole task or only the major components, relationships, and/or functions that characterize the task. The important point is that this process allows some level of foresight into the effect of the proposed changes.

In this paper, we will be concerned with "*modeling systems"*. This term is further defined below but for now we can say that a modeling system tells the researcher what types of behaviors to observe and how to organize the data into a functioning model. Thus, a modeling system both guides the task analysis and produces the model. A modeling system could be quite formal (e.g., NGOMSL, Kieras, 1988) or very informal, based on common sense notions about what is important in the task (in this case the researcher may be unaware they are using a modeling system). However, both our formal and informal modeling systems have difficulty coping with complex, distributed systems. One reason for this is that it is easier to think in terms of tasks performed by single agents than tasks distributed across multiple agents, especially when the distributed system is not under some sort of centralized control. When agents act locally and organize themselves, multiple different ways of completing the task can emerge. This results in several different levels of analysis, including the following: (1) the knowledge level, the steps that must be taken to complete the task, (2) variations on a theme, the different ways the task can be done given the constraints of the knowledge level, and (3) the different ways that agents can organize themselves to accomplish different steps of the task. To cope with this, a modeling system must be able to represent the

task at different levels and also be capable of integrating factors involved in completing the task with the factors involved with organizing and sustaining cooperation between the agents. In this paper, we describe how GOMS can be used to cope with this type of system, and the relative advantages of using GOMS under these conditions.

GOMS

GOMS is a family of relatively formal modeling systems, but we would argue that it has a special status amongst modeling systems. In this regard, it is useful to consider Anderson's (1993) distinction between frameworks, theories and models of human cognition. According to this scheme, frameworks are "bold, general claims about cognition," (p. 2). Theories are created by adding specific assumptions as to how frameworks could be applied to the relevant class of behaviors, and models are created by adding assumptions as to how a theory could be applied to a specific situation or task. The idea that cognition can be understood in terms of production rules (i.e., if…then statements) is therefore a framework, and systems embodying assumptions as to how to use production rules are theories. However, rather than theory, we will use the term *modeling system*, because we are focusing on the process of model building, rather than on testing theories. So, to be clear, we will define a modeling system as a system that allows us to create a model within a specified framework.

The general idea behind GOMS is that well learned human behavior can be modeled using goals, operators, methods, and selection rules (e.g., John, 1995). This claim places GOMS clearly within the production rule framework. Using selection rules to choose between different methods for accomplishing a task essentially embodies the idea of the production rule (i.e., if this, then use this method). Also, operators are necessary for any production system to specify how the system retrieves information from the world and generates behaviors in the world (although operators are sometimes not explicitly represented in production system models of cognition, they are always assumed to exist). The idea that people have goals, or more specifically the idea that people create sub goals to bring them closer to their end goals, is the only element of GOMS that is not directly tied to implementing production systems. For example, the first attempts at implementing production systems (e.g., SOAR, ACT) did not contain any mechanisms for managing goals (Anderson & Lebiere, 1998). However, as Anderson and Lebiere (1998) point out, all of the current production system architectures have a structure for keeping track of goals. Thus, the idea that we use goals to organize cognition can be considered another framework (i.e., it is a bold, general claim about cognition). Therefore, GOMS can be interpreted as asserting that well learned behaviors can be captured using the combined frameworks of production rules and goal structures. At this level, GOMS itself is a general claim about a class of behaviors and remains at the framework level (it is also not possible to falsify this claim without adding further assumptions, another hallmark of the framework level, see Anderson, 1993).

Cognitive architectures can be considered as relatively complete modeling systems (Anderson, 1993). Unlike these systems, GOMS has no mechanisms for constructing or searching the problem space, it presupposes that the agent has already learned how to get to the end goal. *The key insight, on which GOMS was founded, is that once a path through the problem space has been learned, the complexity of the modeling task is hugely reduced*. This makes moving from the framework level to the modeling system level easier. In fact, the simplest possible GOMS modeling system can be created by merely assuming the appropriate operators exist and structuring goals by connecting them serially, essentially creating a flow chart of goals with branching paths gated by production rules. This type of GOMS modeling system is frequently used, often to sketch out the task structure before creating a more fully functional model. Since this system has no name we will refer to it as Minimal-GOMS.

Other, specific GOMS modeling systems, such as NGOMSL (Kieras, 1988) and CPM-GOMS (Gray, John, & Atwood, 1993), have more detailed assumptions that are contained in the human information-processor (Card, et al., 1983). In this sense, GOMS can be considered a general outline for moving from the dual production rule/goal framework to a specific modeling system by adding assumptions concerning the human information-processor. Following from this, any model that (1) is explicitly or implicitly based on the dual production rule/goal framework, (2) refers only to knowledge driven behaviors (i.e., no learning or problem solving), and (3) makes assumptions concerning the behavior of the human agents involved, can be interpreted as a type of GOMS model. For lack of a better term, we will refer to models that fall into this category, but have not been explicitly created and labeled as GOMS models, as GOMS-like.

Since we are currently interested in modeling errors within complex sociotechnical systems, we searched the literature for error modeling systems and found over 50. However, **c**omparing GOMS to these modeling systems it is clear that they are not on the same conceptual level. In fact, the product of many of the modeling systems we reviewed would be a GOMS-like model. This issue is often the source of confusion and contention between designers that favor GOMS and designers that do not. It is not uncommon to hear people say that modeling system X is a better approach than GOMS, when in actuality modeling system X is a system that produces GOMS-like models.

Part of the problem seems to have arisen from the association of GOMS with models of how long it takes to perform isolated tasks described at the level of individual mouse clicks and button presses. GOMS is particularly good at describing low level activities because the operators are relatively simple and can be described with a reasonable accuracy in the human information-processor (Card, et al., 1983). Since a lot of research, explicitly represented as GOMS research, was done at this level, there is a strong tendency for people to view GOMS as synonymous with the use of low level operators. In actuality, the grain size of the operators should depend on the goals of the researcher (West, Wong, and Vera, 1998).

In terms of complex sociotechnical systems, it is unlikely that GOMS could produce very accurate time estimates as it

is often not possible to assign very precise times to high level social operators (e.g. how long does it take arrange a lunch meeting with a colleague), although, it should still be possible to get good time estimates for well-defined sub tasks. However, the value of GOMS in a multi-agent system is that is allows us to examine the goals and methods of individual agents, and how these relate to the overall task. For example, multi-agent tasks are often described using a critical path analysis. In the case of a centrally controlled task the critical path represents the plan of the central controller. However, when the task is not centrally controlled (i.e., a complex system) the critical path is an emergent property of the interactions between the agents. A multi-agent GOMS model can allow us to examine these interactions for inefficiencies, goal conflicts, and sources of error.

Complex Sociotechnical Systems

One of the most influential modeling systems in terms of modeling complex systems has been Rasmussen's decision ladder model (1980). As Vicente (1999) points out, the step ladder model is not really a model, but rather a *template* for creating models. Essentially, it is a generic model of information processing that can guide the modeler in terms of the general form a model should take (Vicente, 1999). We believe that the template approach is important for modeling complex, sociotechnical systems, and, more specifically, that it can be used to effectively situate GOMS models within such systems. As Vicente (1999) points out, work within a sociotechnical system cannot be fully captured by GOMS or GOMS-like models because this type of work involves ongoing learning and problem solving, which these models cannot handle. However, as John (1995) points out, GOMS can be very useful for elucidating the components of a task that are amenable to GOMS modeling. In other words, GOMS doesn't have to be the whole solution, but can be part of the solution.

Another important aspect of sociotechnical modeling systems is that they need to be multifaceted in focus. For example, Vicente's modeling system is actually a collection of modeling systems for examining various aspects of the sociotechnical environment, including: the work domain, control tasks, strategies, social organization and cooperation, and worker competencies. Likewise, a modeling system advocated by a well known consulting firm in this area involves a work flow model, a cultural work model, a sequence work model, an artifact model, and a physical environment model (this system is adopted from Beyer & Holzblatt, 1997). What GOMS adds is the potential to integrate knowledge gained in these different domains into a unified model of the knowledge driven portions of the process. Our approach to this is to use a template that (1) allows the task to be described at different levels of complexity and (2) describes how people situate knowledge driven tasks within a complex environment involving ongoing learning and problem solving.

The Basic Model

Our modeling system is closely related to Norman's (1986) seven-stage model of user activities. However, similar to Rasmussen's decision ladder model, we intend our model to be a generic template for information processing in general, rather than a specific model of human cognition. The framework, which is described in Figure 1, revolves around the goal, *create-plan*. This goal is meant to deal with learning and problem solving, so overall it lies outside the reach of GOMS. One approach to modeling this component would be to use a production/goal based cognitive architecture (e.g., ACT-R, SOAR). This would tie in nicely with the GOMS aspects of the model since they share a common framework, however, any approach can be used, including treating *create-plan* as a black box.

In our current work on telecommunications network maintenance and management we are using Vicente's (1999) work domain analysis to provide the underpinnings for the create-plan component. This involves understanding the constraints imposed by the sociotechnical system and, rather than specifying what a worker should do, specifying what a worker should not do. For example, the main constraint that we have identified is that of the working path (the path through the network carrying live traffic) and the protection path (a path to which the traffic could be shifted). This constraint is critical because whenever work needs to be done or a problem occurs the traffic must be rerouted along a protection path. We are also using Hutchins' (1995) concept of organizational learning to look at how workers pick up on this constraint. GOMS modeling, based on the Figure 1 template, provides the means for describing and evaluating how knowledge driven, procedural tasks fit into the picture. The use of GOMS is very important since this type of sociotechnical system involves many knowledge driven components.

From the perspective of the rest of the model, the function of the create-plan component is to output a knowledge driven plan. The plan may be complete and well thought out but in many cases this will not be the case. Essentially, the cycle embodied by the template is to continue with a plan until it is evaluated as inappropriate or is completed. To further structure this process we need to invoke another GOMS concept, the *unit task* (see Card, et al., 1983). In theory, a plan could be of any size, but we conceptualize plans as unit tasks in the sense that they should correspond to actions that the agent believes can be accomplished without a terminal interruption. Thus, the size of the plan is determined by the nature of the task. For example, the results of Kvan, West, and Vera (1998) indicate that architects in the process of collaborating over a shared whiteboard use very short plans, whereas maintenance procedures on network hardware can involve lengthy procedures that must be completed once started.

Another important function of the create-plan component is to integrate technical, environmental, and social aspects of the task. Thus, in addition to technical procedures a plan should include how to deal with issues arising from the physical environment the task is situated in, as well as the social issues involved in getting cooperation from other agents. As West, et al. (1998) argued, in many cases there are routine ways of dealing with these issues if they

represent routine occurrences. However, in other cases these issues may be dealt with in unique, creative ways. Either way, the model is capturing valuable information (i.e., routine solutions or different case based solutions). Note, though that we are not saying that plans are always complete in this sense. In many cases, plans fail because they do not include ways to deal with problems arising from the physical or social environment. In this case, the system returns to the create-plan process to fix the plan or come up with another.

The other components of the template are described below:

1. Retrieve-next-action – This is meant to reflect the fact that the representation of the plan may be distributed. It is often the case that workers do not have all the knowledge necessary for the task, but they know where to get it (e.g., memory, personal notes, manuals, colleagues).
2. Execute-action – This step refers to firing of operators. As is normally the case in GOMS models, operators can be either physical (e.g., move the mouse), perceptual (e.g., search the screen), or cognitive (e.g., add two numbers). Operators can also vary in grain size and represent complex tasks. For example, an architect might use an operator, *make-aesthetic-judgement*. Such an operator could be represented in terms of the % chance that such a judgement will be positive, or may merely represent the fact that the judgement takes place at a certain point in the model. The grain size and function of the operators will depend on the modeler's goals (see West, et al., 1998 for a discussion about high level operators). This step is also where communication is initiated between agents by using an operator to place messages in the environment (e.g., voice, email, etc.).
3. Update-situation-knowledge – After having acted in some way this section refers to updating the task knowledge to reflect these changes and any other relevant changes that may have occurred during that time (including messages from other agents). For isolated, low level actions this step could be assumed to occur as the actions are being executed. For complex, interactive actions the process of checking may be quite extensive, and may also involve retrieving knowledge from various sources. In this case adding a box above it entitled, *retrieve-data*, might be a good idea.
4. Evaluate – Like create-plan, this box may involve actions that step outside of GOMS. If the situation has changed in an unexpected way there must be a judgement as to whether or not the plan is still appropriate. By definition, unexpected changes will not be part of the plan (Vicente, 1999), so there is a need to step outside of the plan into problem solving or creative thinking to make this judgement. However, it is possible to handle expected or common problems within a plan. Another issue that is important here is the agent's evaluation of risk. Human agents will often engage in risky behaviors, especially if they are under time pressure. Often workers will have heuristics for evaluating risk that can be captured by GOMS.
5. Execute-Patch – if there is a known and immediate fix for a problem the agent goes to execute-patch where the patch with highest probability of success is executed. These known patches can be considered to be implicitly part of the plan. If there is no immediate fix the agent goes to create-plan where a known fix is inserted into the plan to be executed later, or the plan is recreated to cope with the problem.
6. Parallel External Monitoring (PEM) – This module operates in parallel, monitoring the environment for alarms. Creating the model for PEM involves understanding the extent to which the agent can pay attention to the task and also to the general environment. For example, an alarm siren could be assumed to be always picked up, whereas an alarm on a screen could only be picked up when the agent is looking at the screen. The other aspect of the PEM model is that it contains rules for when to interrupt the system and go directly to update-situation-knowledge, and when to store the information in memory until the update-situation-knowledge process comes up. The goal of this model is to capture expert knowledge about monitoring and interruptions.

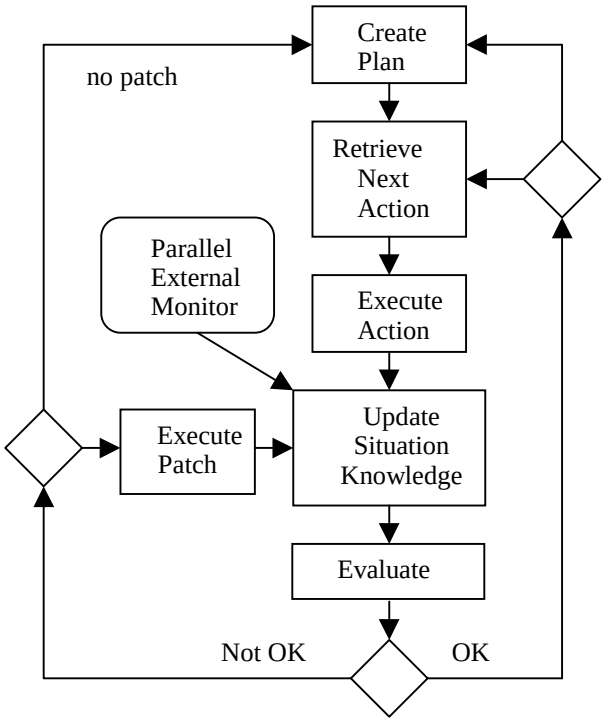

Figure 1. A generic Minimal-GOMS template

Multiple Agents

So far we have dealt only with modeling an individual. In fact, the original version of this template was developed in an attempt to make sense of data gathered from pairs of architects working collaboratively over a shared whiteboard. As reported in Kvan, West, and Vera (1998), the architects never developed a plan for the collaboration, instead they

dealt with issues and organized themselves as they went along. This resulted in very different organizational structures, all of which were difficult to model. To simplify things a version of the Figure 1 template was developed to first understand the behavior of the individual architects. To create a model of two agents working together you just simply add another template. No lines of communication need to be drawn between the two templates. Instead what is needed is a simple model of the environment that the agents can act on by altering the physical components of the task and by creating messages (e.g., voice, notes, email, etc.). Since the agents are modular they can be added or deleted without too much trouble, so it is possible to have more than two agents.

Using this approach, it was obvious that architects generate very small plans with regard to the task (e.g., draw box at location X) that serve a constantly evolving creative vision. Thus, low level GOMS models of the task components, as defined by the plan size, would be appropriate. However, in addition to creating objects the architects also needed to understand the objects that their partner was creating. This caused a problem in one condition of the experiment in which the architects used a chat line to communicate. To attach a message to an object (e.g., "what is that?") they would either have to describe the object in the message or tell the other person to watch the their whiteboard pointer while they pointed (the white board could get quite complex in terms of the number of objects on it). A solution that would involve fewer steps would be to attach a text box to the pointer to combine the activities of message passing and pointing. This particular solution is not complex, but recognizing the need for it was facilitated by integrating the collaborative elements of the task into the model. Also, notice that although the pointing/messaging solutions the architects came up with were the result of online problem solving, once created they could be treated and evaluated as GOMS type methods.

Distributed Agents

Although the template is useful for organizing models in which individuals interact, only a relatively small number of agents can be included before the model gets unwieldy. In contrast, complex sociotechnical systems often involve a considerable number of agents. However, we have found that the template is scalable to what we call *distributed agents*. The central premise of distributed cognition is that cognitive agents can organize themselves to form larger, distributed cognitive systems (Hutchins, 1995). Our approach is to treat these distributed cognitive systems as individual agents and apply the same template. This is not to say that there are no differences between brain based cognition, distributed cognition occurring across small groups, or distributed cognition occurring across large groups. There are important differences between these types of structures. However, our argument is that the template captures something basic about the way cognitive systems, in general, deal with interactive, knowledge driven tasks.

We tried this approach at Oxfam, Hong Kong, for modeling the process of deciding how to deliver aid to flood victims in China and found that it simplified the process considerably (West & Yeun, 1999). It also brought to our attention the distinction between distributed agents and official groups defined within the management structure (i.e., specific departments and their subdivisions). There is a strong tendency for organizations to understand themselves in terms of their official subdivisions, and this information should be part of a complex systems task analysis. However, the goal of GOMS is to build task models, not organizational models. Therefore, a distributed agent is meant to map onto agents that function together to complete a particular task, and will not necessarily map onto a particular department or section. This also means that a person may be a part of different distributed agents depending on the task they are working on. One benefit of this type of analysis is that it can provide insight into the relationship between the task and the management structure.

Following this approach, it is possible to create a higher level model describing the interaction between distributed agents. As with individual human agents, the approach is to represent each distributed agent using the template structure, with communication occurring by placing messages in the environment. Also note that it is possible to combine distributed agents into higher level distributed agents or to break them up into lower level distributed agents, depending on the level of the analysis. It is also possible to mix agents representing individuals with distributed agents. This allows the model to focus in on an individual without representing every other individual connected to the task.

Currently, we are using this modeling system to model tasks involved in telecommunications network maintenance and management. We have found that using this system greatly simplifies the modeling process and also allows the flexibility to address a wide variety of questions.

References

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Bennett, J. (1998). FCC-Reportable Service Outages (3Q92-4Q98) with Procedural errors as Root Cause. Telcodia White Paper.

Beyer, H., & Holtzblatt, K. (1997). *Contextual design: A customer-centered approach to systems design*. Morgan Kaufmann Publishers.

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

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), 237-309.

Hutchins, E. (1995). *Cognition in the wild*. Cambridge, MA: The MIT Press.

John, B. E. (1995). Why GOMS? *Interactions. 2* (10), 80-89.

Kieras, E. E. (1988). Towards a practical GOMS methodology for user interface design. In M. Helander

(Ed.), *The handbook of human computer interaction* (pp. 135-138). Amsterdam: North-Holland.

Kvan, T., West, R. L., & Vera, A. H. (1998). Tools for a virtual design community: Modeling the effects of different tools on design communication. *International Journal of Virtual Reality, 3* (3), 21-33.

Mayer, R. E. (1997). From novice to expert. In M. Helander, T.K. Landauer, and P. Prabhu (Eds.), *Handbook of human-computer Interaction* (pp. 781-795). Amsterdam: Elsevier Science.

Norman, D. A. (1986). Cognitive engineering. In D. A. Norman and S. W. Draper (Eds.), *User centered system design: New perspectives on human-computer* (pp. 31-61). Hillsdale, NJ: Lawrence Erlbaum Associates.

Rasmussen, J. (1980). The human as a systems component. In H. T. Smith and T. R. G. Green (Eds.), *Human interaction with computers* (pp. 67-96). London: Academic Press.

Vicente, K. J. (1999). Cognitive work analysis: Toward safe, productive, and healthy computer-based work. Mahwah, NJ: Lawrence Erlbaum Associates.

West, R. L., Wong, A., & Vera A. H. (1998). GOMS, Distributed Cognition, And the Knowledge Structures Of Organizations. *Proceedings of Cognitive Science 1998*.

West, R. L., & Yuen, K. L. (1999). *A framework for incorporating social context into GOMS models*. Poster presented at Cognitive Science 1999, Vancouver, B.C.