Modeling Human-Robot Interaction with GOMS

Jill L. Drury
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730-1420 USA
+1-781-271-2034
jldrury@mitre.org

Jean Scholtz
Pacific Northwest National Laboratory
P.O. Box 70
Rockaway Beach, OR 97136
+1-503-355-2792
jean.scholtz@pnl.gov

David Kieras
University of Michigan
2260 Hayward Street
Ann Arbor, MI 48109-2121
+1-734-763-6739
kieras@umich.edu

ABSTRACT
The Goals, Operators, Methods, and Selection rules (GOMS) method is a well-established means of modeling the procedures that humans use to interact with technology. We focus on two questions: what is different about using GOMS for human-robot interaction (HRI) versus using GOMS for traditional computer applications, and what are promising approaches for using GOMS to evaluate competing HRI designs? This paper raises issues in using GOMS for modeling HRI and illustrates them with GOMS models that compare two interfaces for urban search-and-rescue robots. Very little work has been done with GOMS so far in the HRI domain, so one of our chief contributions is the guidance we provide for using GOMS for HRI.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – graphical user interfaces, input devices and strategies, interaction styles, screen design.

General Terms
Design, Human Factors, Verification.

Keywords
GOMS, human-robot interaction, evaluation, interface design, dialog modeling.

1. INTRODUCTION
Mobile robots have been making a strong showing in the commercial marketplace in the last few years. Suddenly robotic vacuum cleaners are in many thousands of homes, unmanned aerial vehicles are being produced by dozens of different contractors worldwide, and bomb disposal robots are standard equipment in police and military organizations, to name a few examples. With the increase in the numbers and types of robots that are commercially available comes the challenge of ensuring that the intended end users can easily and efficiently use the interfaces to those robots. The technically-oriented researchers and developers who used to be the sole operators of robots have been increasingly supplanted by homeowners, military security specialists, and police officers, among others.

Human-robot interaction (HRI) has been maturing in tandem with robots' commercial success. In the last few years HRI researchers have been adopting—and sometimes adapting—human-computer interaction (HCI) evaluation techniques to assess the efficiency and intuitiveness of HRI designs. For example, Adams (2005) used Goal Directed Task Analysis to determine the interaction needs of officers from the Nashville Metro Police Bomb Squad. Scholtz et al. (2004) used Endsley's (1988) Situation Awareness Global Assessment Method to determine robotic vehicle supervisors' awareness of when vehicles were in trouble and thus required closer monitoring or intervention. Yanco and Drury (2004) employed usability testing to determine (among other things) how well a search-and-rescue interface supported use by first responders. One set of HCI tools that has so far seen little exploration in the HRI domain, however, is the class of modeling and evaluation techniques known as formal methods.

Perhaps the most widely-used of the formal methods is the Goals, Operations, Methods, and Selection rules (GOMS) technique first presented by Card, Moran, and Newell (1983), and which then developed into several different forms, summarized by John & Kieras (1996a, 1996b). Depending upon the type of GOMS technique employed, GOMS models can predict the time needed for a user to learn and use an interface as well as the level of internal consistency achieved by the interface. GOMS has proven its utility because models can be developed relatively early in the design process when is cheaper to make changes to the interface. Analysts can use GOMS to evaluate paper prototypes' efficiency, learnability, and consistency early enough to affect the design prior to its implementation in software. GOMS is also used with mature software to determine the most likely candidates for improvement in the next version. Since GOMS does not require participation from end users, it can be accomplished on shorter time scales and with less expense than usability tests. Based on its use as a cost savings tool, GOMS is an important HCI technique: one that bears exploration for HRI.

Very little work has been done so far in the HRI domain using GOMS. A method we used for coding HRI interaction in an earlier study (Yanco, Drury, and Scholtz, 2004) was inspired by GOMS but did not actually employ GOMS. Rosenblatt and Vera (1995) used GOMS for an intelligent agent. Wagner et al. (2006) used GOMS in an HRI study but did so in limited scenarios that did not explore many of the issues specific to HRI. In an earlier paper (Drury and Scholtz, in submission), we explored more types of GOMS than was presented in Wagner et al. and also modeled a wider range of tasks for a single interface. This paper presents a case study of comparing two interfaces using a Natural GOMS Language (NGOMSL) model and provides a more detailed discussion of GOMS issues for HRI than has been previously published. The primary contribution of this paper is the guidance that we provide for using GOMS in future HRI modeling and evaluation efforts.

The next section discusses GOMS and what is different about using GOMS for HRI versus other computer-based applications. Section 3 contains background information on the two interfaces that we have modeled, prior to presenting representative portions of the models in Section 4. As we present the models, we provide guidance for using GOMS in HRI. Finally, we provide conclusions in Section 5.

2. WHY IS HRI DIFFERENT WITH RESPECT TO GOMS?

Before discussing the nuances of using GOMS for HRI, we briefly describe GOMS. There is a large literature on GOMS and we encourage the reader who is unfamiliar with GOMS to consult the overviews by John and Kieras (1996a, 1996b).

2.1 The GOMS Family

GOMS is a "family" of four widely-accepted techniques: Card, Moran, and Newell-GOMS (CMN-GOMS), Keystroke Level Model (KLM), Natural GOMS Language, and Cognitive, Perceptual, and Motor GOMS (CPM-GOMS). John and Kieras (1996a) summarized the four different types of GOMS:

> --CMN-GOMS: The original formulation was a loosely defined demonstration of how to express a goal and subgoals in a hierarchy, methods and operators, and how to formulate selection rules.
>
> --KLM: A simplified version of CMN was called the Keystroke-Level Model and uses only keystroke operators — no goals, methods, or selection rules. The analyst simply lists the keystrokes and mouse movements a user must perform to accomplish a task and then uses a few simple heuristics to place "mental operators."
>
> --NGOMSL: A more rigorously defined version of GOMS called NGOMSL (Kieras, 1997) presents a procedure for identifying all the GOMS components, expressed in structured natural language with in a form similar to an ordinary computer programming language. A formalized machine-executable version, GOMSL, has been developed and used in modeling (see Kieras and Knudsen, 2006).
>
> --CPM-GOMS: A parallel-activity version called CPM-GOMS (John, 1990) uses cognitive, perceptual, and motor operators in a critical-path method schedule chart (PERT chart) to show how activities can be performed in parallel.

We used the latter three types of GOMS in Drury and Scholtz (in submission). In this paper we use NGOMSL only, because it emphasizes the interface procedures and their structure. We discuss our selection of NGOMSL in more detail in Section 4.2.

2.2 HRI Challenges

Traditional GOMS assumes error-free operation on the part of the user and predictable operation on the part of the computer. These assumptions are unreasonable for the HRI domain. Robots take unexpected actions, including autonomous actions that are, in some sense, "wrong." Even when robots are operated without autonomy, they sometimes take actions that the user does not expect. GOMS can be extended to cover user errors (e.g., Kieras, 2005 and Wood and Kieras, 2002), but this would not apply to "errors" on the part of the robot. The fact that the user cannot predict the state of the robot or environment in the near future means that models must account for a great range and flexibility of users' responses to any given situation.

A second challenge for HRI pertains to the seemingly simple task of maneuvering the robot, which normally occurs with a pointing device such as a joystick. While GOMS has long modeled users' use of pointing devices to move cursors or select different items on the computer display, it has not developed mechanisms to model the types of movements users would employ to continuously or semi-continuously direct a robot's movement with a joystick. This missing element is important because there are fundamental differences in the amounts of time that are spent moving a cursor with a pointing device versus pushing a joystick to steer a robot's motion.

Wagner et al. (2006), the only other case study of using GOMS for HRI besides our work, models mission generation for robots and does not include this basic task of driving a robot. GOMS has been used frequently in the aviation domain and so we scoured the literature to find an analogous case, for example when the pilot pulls back on the rudder to change an aircraft's altitude. To our surprise, we found only analyses such as Irving et al. (1994) and Campbell (2002), which concentrated on interactions with the Flight Management Computer and Primary Flight Display, respectively: interactions confined to pushing buttons and verifying the computers' responses. Thus we have had to grapple with the issue of continuous joystick movement in conjunction with our work.

A third challenge relates to modeling mental operations to incorporate the right amounts of time for the users' thought processes at each stage of using an interface. For example, previous empirical work has shown that it takes a user approximately 1.35 seconds to mentally prepare to perform the next action when executing a routine task in a predictable environment (John and Kieras, 1996b). But robot operations are notoriously non-routine and unpredictable, as discussed above. Instead, users may almost continuously extract dynamically changing information from the environment. Luckily, GOMS has always assumed that application-specific mental operators could be defined as necessary: what is difficult is determining the mental operators that make sense for HRI.

A further challenge with mental and perceptual operators in GOMS is that they do not account for the effects of having varying quality sensor data, either within the same system at different times or on multiple systems that are being compared. For example, if video quality is bad on one system but exceptionally clear on another, it will take more time to extract video-based information via the first system's interface than the second. Each GOMS operator is normally assigned a single value as its typical time duration, such as the 1.35 seconds cited above for mental preparation. Unless a perceptual operator is assigned one time value in the model for the first system and a shorter time value for the second model (to continue the example), the models will not take into account an important difference that affects performance.

A fifth challenge pertains to different levels of autonomy. We do not know of any use of GOMS models that accounts for the differences in autonomy levels defined for mobile robots. We believe it would be very useful, for example, for GOMS models to tell us whether it is more efficient for the robot to not allow the user to get too close to objects, as opposed to the user having

to spend time and effort watching out for obstacles immediately around the robot.

As we present our GOMS models in Section 4, we provide guidance for overcoming these challenges.

3. SPECIFIC INTERFACES ANALYZED

But first, we need to describe the interfaces we modeled. User interface analysts model human activity assuming a specific application interface because the models will be different for each application. Our decision regarding which interfaces to analyze was not important as long as the chosen interfaces contained representative complexity and functionality. We chose two mature interfaces that use the same set of urban search-and-rescue (USAR) functionality and the same robotic platform.

3.1 Interface "A"

The architecture for the system underlying Interface A was designed to be flexible so that the same interface could be used with multiple robot platforms. We observed the interface operating most frequently with an iRobot ATRV-Jr.: the same one used for Interface B.

Interface A (Figure 1) is displayed on a touch screen. The upper left corner of the interface contains the video feed from the robot. Tapping the sides of the window moves the video camera left, right, up or down. Tapping the center of the window re-centers the camera. Immediately to the right of the video display are pan and tilt indicators. The robot is equipped with two types of cameras that the user can switch between: a color video camera and a thermal camera; the camera selection radio buttons are also to the right of the video area.

The lower left corner contains a window displaying health status information such as battery level, heading, and attitude of the robot. A robot-generated map is placed in the lower central area. In the lower right corner, there is a sensor map that shows red arrows to indicate directions in which the robot's motion is blocked by obstacles.

The robot is controlled through a combination of a joystick and the touch screen. To the right of the sensor map in the bottom right hand corner of the touch screen, there are six mode buttons, ranging from autonomous to tele-operation. Typically, the user touches one of the mode buttons, then uses the joystick to steer the robot if not in the fully autonomous mode.

When the user wishes to take a closer look at something, he or she touches the video window to pan the camera. For victim identification, the user often switches to the thermal or Infrared (IR) sensor (displayed in the same space as the videostream and accessed via a toggle) to sense the presence of a warm body.

The proximity sensors are shown around a depiction of the robot in the bottom right hand side of the display. The triangles turn red to indicate obstacles close to the robot. The small, outer triangle turns red when the robot first approaches objects, then the larger, inner triangle also turns red when the robot moves even closer to an obstacle. The location of the red triangles indicates whether the blockage is to the front, rear, and/or sides of the robot.

Note that System A's interface does not incorporate menus. Visual reminders for all possible actions are present in the interface in the form of labels on the touch screen.

While the organization that developed System A has explored other interface approaches since this version, users access almost exactly the same functionality with the current interface designs. Also, many other USAR robots incorporate similar functionality.

3.2 Interface "B"

While Interface B is also used in conjunction with an iRobot ATRV-Jr. robot, this platform was modified to include a rear-facing as well as forward-facing camera. Accordingly, the interface has two fixed video windows, as can be seen in Figure 2. The larger one displays the currently selected camera (either front- or rear-facing); the smaller window shows the other video feed and is mirrored to simulate a rear-view mirror in a car.

Interface B places a map at the edge of the screen. The map window can be toggled to show a view of the current laser readings ("laser zoom view"), removing the map from the screen during that time.

Information from the sonar sensors and the laser rangefinder is displayed in the range data panel located directly under the main video panel. When nothing is near the robot, the color of the box is the same gray as the background of the interface, indicating that nothing is there. As the robot approaches an obstacle at a one foot distance, the box turns to yellow, and then red when the robot is very close (less than half a foot away). The ring is drawn in a perspective view, which makes it look like a trapezoid. This perspective view was designed to give the user the sensation that they are sitting directly behind the robot. If the user pans the camera left or right, this ring will rotate opposite the direction of the pan. If, for example, the front left corner turns red, the user can pan the camera left to see the obstacle, the ring will then rotate right, so that the red box will line up with the video showing the obstacle sensed by the range sensors. The blue triangle, in the middle of the range data panel, indicates the true front of the robot.

The carbon dioxide meter to the right of the primary video screen shows a scale in parts-per-million (PPM) and also indicates the level at which "possible life" has been detected as a blue line. (The platform used for Interface A had an IR sensor to serve this same purpose.) The bottom right hand corner shows battery life, whether the light on the front of the robot is illuminated, a clock, and the maximum speed. The level of autonomy is shown in the bottom right hand set of buttons (currently it is in Shared/Goal Mode).

3.3 Tasks Analyzed

We analyzed tasks that are typical of a search-and-rescue operation: maneuver the robot around an unfamiliar space that is remote from the user, find a potential victim, and confirm the presence of a victim.

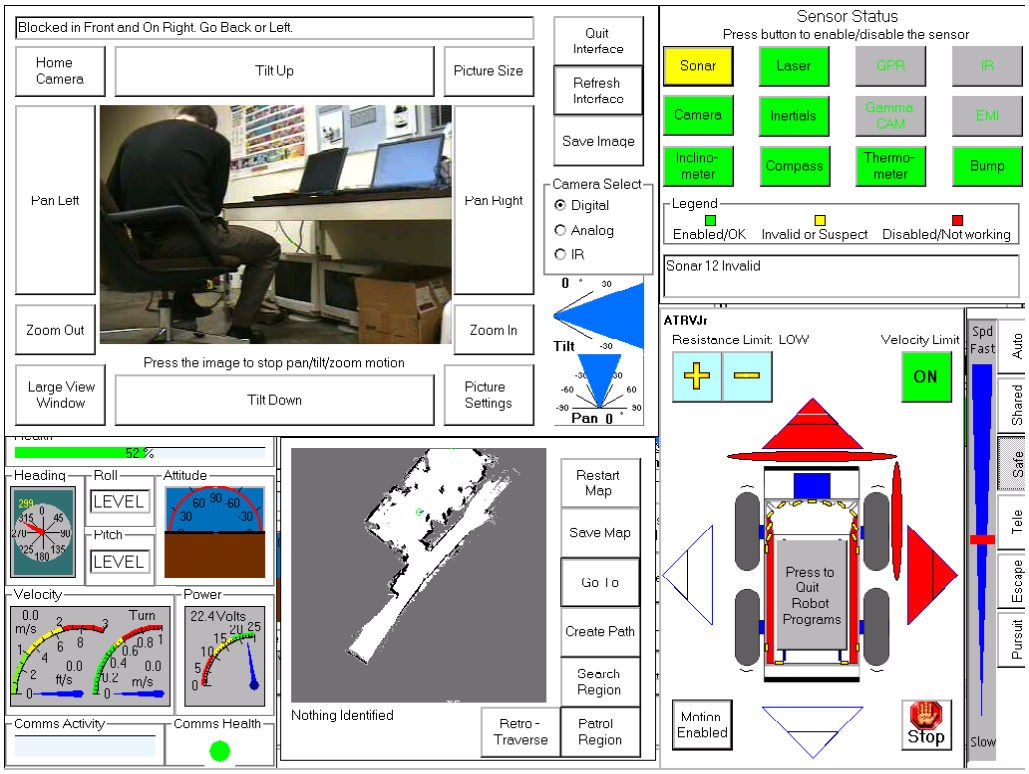

Figure 1. Interface A

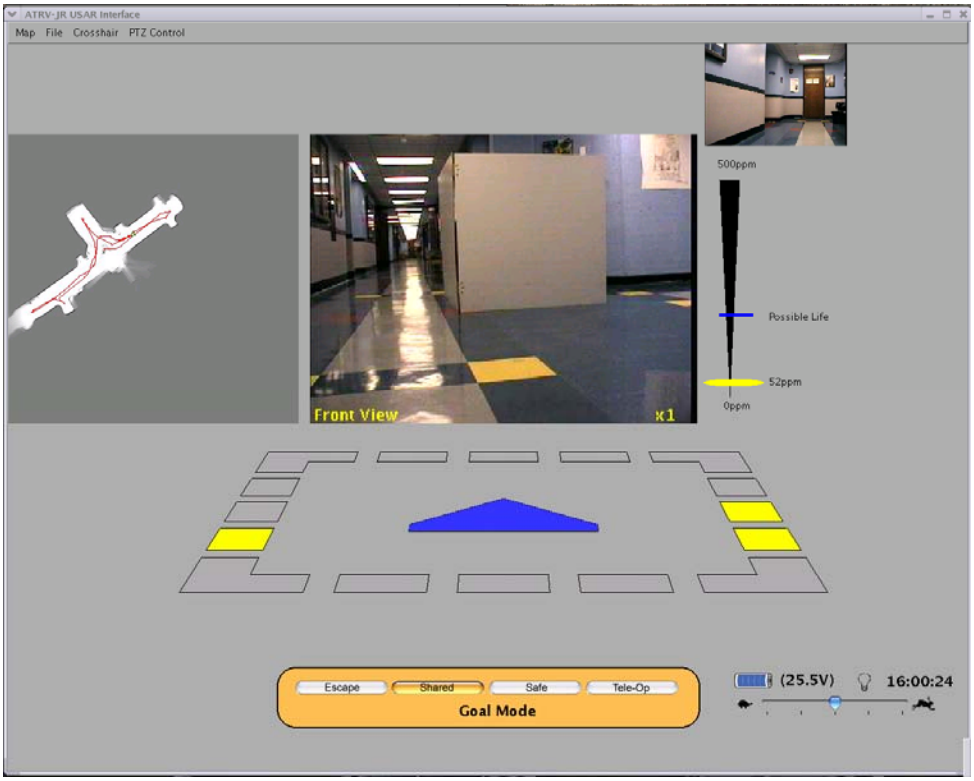

Figure 2. Interface B

4. GOMS ANALYSIS

4.1 Introduction

We view the design process for HRI breaking naturally into two major parts: the perceptual content of the displays and the overall procedural operation.

Designers need to define the perceptual content of the displays so that they can be easily comprehended and used for HRI tasks. This design challenge will normally need to be accomplished using traditional interface design wisdom combined with evaluation via user testing because GOMS does not address whether one visual presentation of an item of information is easier to interpret than another. Rather, GOMS assigns a "mental operator" to the part of the task that involves interpreting display information, but this does not shed any light on whether one presentation facilitates users extracting information more quickly than another presentation—the standard mental operator is a simple "one size fits all" estimate. If analysts can define different types of domain- or task-specific mental operators for displays, then competing designs can be examined to see which requires more instances of one type of mental operator versus another. If these operator durations can be measured empirically, then the GOMS model can make a more quantitative contribution.

GOMS can clearly help with the procedural implications of display design. For example, perhaps one design requires actions to toggle the display between two types of information, while another design makes them simultaneously present; GOMS would highlight this difference. Designers also need to define the overall operation of the user interface in terms of the procedures that the user has to follow to carry out the task with the interface. Evaluating the procedural design challenge can be done easily and well with GOMS models if the perceptual design challenge can be handled so as not to confound any comparisons that analysts might make between competing designs.

This brings us to our first guideline for modeling HRI using GOMS:

(1) Don't get bogged down in modeling the perceptual content of the displays.

The analyst should focus instead on the step-by-step procedures used when interacting with the interface, keeping in mind that issues in perception might determine and dominate issues regarding procedures. Getting bogged down in the perceptual issues is tempting because this part of the task is obviously important, but current modeling technology doesn't allow analysts to make much progress a-priori. Many tasks demand that the operator view video or dynamic sensor data. They may need to do this multiple times in order to understand the situation. Because these activities are so situation-dependent and also dependent upon environmental conditions, skill levels, and physical capabilities of the users (eyesight acuity, dexterity, etc.), there is no way to predict the number of times a user may consult a display. Even if we could know how many times users would need to refer to a particular part of the displays, it is difficult to assign accurate times to the actions.

Instead, we recommend that the primary goal of the GOMS modeling should be to characterize everything else about the task—that is, all the non-perceptual, procedural activities to work with the interface—while holding the perceptual subtasks constant at some representative level of activity as driven by the procedural aspects of the interface. This approach is especially important due to the unpredictable, error-prone nature of working with robots, which requires a significant amount of perception and cognition to understand what the robot is doing and determine corrective steps when necessary.

This modeling strategy is not ideal if the goal is to simply represent a single interface. It becomes much more useful when comparing two interface designs to show where the procedural differences appear and their consequences. When improving designs incrementally, however, an analyst can model a single interface to the point where it exposes inconsistencies or inefficiencies, suggest improvements to that interface design, then use the improved design as a second design to compare to the first to identify the degree of improvement attained.

4.2 Choice of GOMS Technique

We present our models using NGOMSL because this form of GOMS is easy to read and understand while still having a relatively high level of expressive power. NGOMSL can be thought of as stylized, structured pseudocode. The analyst starts with the highest level goals, then breaks the task into subgoals. Each subgoal is addressed in its own method, which may involve breaking the task further into more detailed subgoals that also are described in their own methods. The lowest-level methods contain mostly primitive operations. Design consistency can be inferred by how often "basic" methods are re-used by other methods. Similarly, efficiency is gained when often-used methods consist of only a few steps. The number of methods and steps is proportional to the predicted learning time.

Because NGOMSL lacks the ability to describe actions that the user takes simultaneously, we adopt a bit of syntax from the executable version GOMSL (Kieras and Knudsen, 2006), the keyword phrase "Also accomplish goal…", when we need to show that two goals are being satisfied at the same time.

Since all detailed GOMS models include "primitive operators" that each describe a single, atomic action such as a keypress, we discuss primitives next.

4.3 Primitives

At the lowest level of detail, GOMS models decompose a task into sequences of steps consisting of *operators*, which are either motor actions (e.g., home hands on the keyboard) or cognitive activities (e.g., mentally prepare to do an action). As summarized by John and Kieras (1996a, 1996b), the following primitive operators are each denoted by a one-letter code and their standard time duration:

- **K** to press a key or button (0.28 s for average user)
- **B** to press a button under the finger (e.g. a mouse button) (0.1 s)
- **M** perform a typical mental action, such as finding an object on the display, or mentally prepare to do an action (1.35 s)
- **P** to point to a target on a display (1.1 s)
- **H** to home hands on a keyboard or other device (0.4 s)
- **W** to represent the system response time during which the user has to wait for the system (variable)

As discussed above, none of these primitives are especially suited to describing manipulating the robot, thus we define a "steer" operator **S** and introduce our second guideline:

(2) Consider defining and then assigning a time duration to a robot manipulation operator that is based on typical values for how long the combination of the input devices, robot mechanics, and communications medium (especially for remote operations) take to move the robot a "reference" distance.

This guideline is based on the fact that the time to take the action to manipulate the robot is driven more by the robot mechanics and environment than by the time needed to activate the steering input device. (It may be helpful to consider driving the robot to be a task that is secondary to whatever is the overarching goal for manipulating the robot in the first place. For example, driving the robot is secondary to searching for victims in a USAR task.) We understand that a single robot can run at various speeds which will change based on lighting conditions, proximity to obstacles, etc. When two interfaces are being examined, however, using a single representative speed for each robot should not harm the comparison of their respective GOMS models.

While all the other "standard" primitive operators apply to HRI, the **M** operator bears close scrutiny. As used in modeling typical computer interfaces, **M** represents several kinds of routine bits of cognitive activity, such a finding a certain icon on the screen, recalling a file name, making a routine decision, or verifying that a command has had the expected result. Clearly, using the same operator and estimated time duration for these different actions is a gross simplification, but it has proven to be useful in practice (see John and Kieras, 1996a, 1996b for more discussion). However, consider data being sent to the user from a remote mobile robot that must be continually perceived and comprehended (levels 1 and 2 of Endsley's (1988) definition of situation awareness). This is a type of mental process that is used to assess the need for further action triggered by external, dynamic changes reflected in the interface (e.g. as seen in a changing video display). We intuit that this mental process is qualitatively different, more complex, and more time-consuming from those traditionally represented with **M**. Since this type of mental operation is nontrivial, we posit that it is useful to identify it as a separate operator to determine if one design versus another requires more instances of this type of assessment. For example, if one interface split up key sensor information on separate screens but another provided them on a fused display, the latter design would require fewer mental operators in general and fewer operators of the type that assesses dynamic changes in the environment.

We define a **C** (Comprehend) operator to refer to a process of understanding and synthesizing complex display information that feeds into the user's continually-changing formulation of courses of action. This operator is expected to take much longer than the conventional **M** operator, and will be used to represent the interpretation of the complex displays in this domain.

This leads us to an additional guideline:

(3) Without violating guideline number 1, consider defining HRI-specific mental operator(s) to aid in comparing the numbers of instances these operators would be invoked by competing designs.

Once the set of primitives has been finalized, the next step is to assign times to the various operators. To a certain extent, the exact times are not as important as the relative differences in times that result from competing interface designs. Thus we suggest using time durations that were derived empirically as a result of research on other systems whenever possible, such as the standard operator times listed above. The time required for our mental operator **C** for robotics tasks will depend on the quality of the video, the complexity of the situation being viewed, and the design of the map and proximity sensor displays. Thus, if two systems being compared have radically different qualities of sensor data, we suggest the following guideline:

(4) Without violating guideline number 1, consider assigning time duration values to HRI-specific mental operators that reflect the consequences of large differences in sensor data presentation.

In the absence of conclusive empirical data specific to the system and conditions being modeled, we "guestimate" the time required by the **C** operator to be on the order of a few seconds. This estimate was derived based on observing search-and-rescue operators working with each system.

The system associated with Interface A requires approximately 0.5 seconds delay time **W** before the user can see a response from steering the robot but there is only an average of approximately 0.25 seconds delay time with Interface B. This time difference was noticed and commented on by users and so needs to be reflected in the models' timing calculations.

For **S**, the ultimately skilled robot user would perform all perceptual, navigation, and obstacle avoidance subtasks while keeping the robot moving at a constant speed, thus making execution time equal to the time it takes to cover an area at a given speed. In practice, we found that users spent an average of 30% of the time reorienting themselves to the exclusion of all other activities (Yanco and Drury, 2004). However, we assigned a reference **S** time of 1 foot/second to both robots.

4.4 Top-Level Model

A major part of creating a model for a task is to characterize the top level of the task. A fragment from a preliminary model for the top level of the robot search-and-rescue task is shown in Figure 3. Due to space reasons, we cannot show all of the methods, so we only show those methods that lead to the user determining whether she has spotted a victim. This "thread" of methods is shown in the figure by the bold-face goals.

This top-level model shows the overall assumed task structure. After getting some initial navigation information, the user repeatedly chooses an area to search until all areas have been covered. Each area involves driving around to different locations in that area and looking for victims there. Locating a victim involves repeatedly choosing an area to view, viewing it, and then checking the sensors to see if a victim is present. This last goal will be examined in more detail in the next subsection.

The top-level method focuses attention on a basic issue in the task, namely the extent to which the user can simultaneously drive the robot to cover the area, and locate a victim using the video and sensors. Both interfaces seem to be compatible with simultaneous operation, compared to some other interface that, for example, used the same joystick for both camera motion control and driving. The method shows this simultaneity assumption with the use of the "Also accomplish goal" operator. However, Yanco and Drury (2004) observed that users were able to drive and look for victims simultaneously only about 70% of the time, and often had to pause to reorient themselves. GOMS lacks a direct way to express this sort of variability, so we have commented this step in the method as a place-holder.

Method for goal: **Perform search and rescue mission**
1. Accomplish goal: obtain global navigation information.
2. Choose next local area.
3. If no more local areas, return with goal accomplished.
4. Accomplish goal: **search local area.**
5. Go to 2.

Method for goal: **search local area**
1. Accomplish goal: drive to new location.
The following step applies 70% of the time.
2. Also accomplish goal: **locate victim.**
3. Return with goal accomplished.

Method for goal: **locate victim**
1. Choose next area of location.
2. If no more areas, return with goal accomplished.
3. Accomplish goal: view area of location.
4. Accomplish goal: **view sensors for indication of victim.**
5. If indication shown, return with goal accomplished.
6. Go to 1.

Figure 3. Top-Level Methods for Search-and-Rescue

4.5 Comparing Interfaces A and B

In addition to showing the overall flow of control, the top-level model acts to scope and provide a context for the more detailed modeling, such as the consideration of how different displays might support the goal of viewing sensors for indication of a victim. This is illustrated by critiquing the methods for this goal supplied by Interface A, and then comparing them to Interface B. Note that because the two sensors are different for the two platforms we are comparing the procedure necessary for viewing a thermal sensor as illustrated in Interface A with the procedure for viewing a carbon dioxide sensor as illustrated in Interface B.

The method for Interface A is shown in Figure 4A. As shown in this method, the display must be togged between the normal video display and the IR display. Since it will probably be done frequently, the time cost of this operation might well be significant. While the GOMS model cannot predict how often it would be done, the preliminary estimate is that it would take about 2.2 s per toggling (two **P**, or Point, operators). Clearly this aspect of the design could use improvement. Hestand and Yanco (2004) are experimenting with a USAR interface that places infrared data on top of video data. While research is needed to determine if it takes longer for a user to comprehend combined video/infrared data, this model indicates that the total time may be less than adding additional operators to view and comprehend a separate infrared data window. Similarly, the model predicts that providing multiple camera views in a single display will be more efficient in that such an arrangement may prevent having to manipulate a single camera to alternately point in different directions.

In addition, the IR display is color-coded in terms of temperature, and there is no on-screen cue about the color that indicates possible life, suggesting that the user will need to perform extra mental work. We have represented this as a step to recall the relevant color code.

In contrast, Interface B shows a different approach for another sensor datum, carbon dioxide level. The method is shown in Figure 4B. There is an on-screen indication of the relevant level, requiring a simple visual position judgment rather than a comparison to a memorized color.

Method for goal: **view sensors for indication of victim**
1. Look at and interpret camera display (C).
Using a touchscreen is similar to pointing with a mouse.
2. Point to touchscreen IR button to toggle display (P).
3. Recall IR display color-code that indicates possible life (M).
4. Look at and interpret IR display (C).
5. Decide if victim is present (M).
Need to restore display to normal video to support next activity
6. Point to touchscreen Digital button to toggle display (P).
7. Return with goal accomplished.

Figure 4A. Fragment of a GOMS Model for Interface A

Method for goal: **view sensors for indication of victim**
1. Look at and interpret camera display (C).
2. .Look at and determine whether carbon dioxide level is above "Possible life" line (M).
3. Decide if victim is present (M).
4. Return with goal accomplished.

Figure 4B. Fragment of a GOMS Model for Inteface B

Interface B's method is shorter than Interface A's for several reasons. Interface A cannot show video and infrared sensor information (to show the presence of body heat) at the same time, incurring switch costs, whereas Interface B can show video and carbon dioxide (present in humans' exhalations) sensor readings simultaneously. Also, Interface B explicitly shows the level above which the presence of nearby human life is likely, whereas users looking at Interface A will need to remember which color-coding in the infrared display indicates heat equivalent to human body temperature. This difference in approaches requires one less operator (to recall the appropriate color) as well as changes the nature of the mental operator (from a **C** to an **M** indicating a simple comparison). For one pass through the method, Interface A requires 2 more steps, two **P** operators, and an additional **C** operator - at least a few more seconds - than Interface B.

This example shows how although GOMS cannot predict the interpretability of display elements, the costs and benefits of

different design decisions about when those elements are present can be modeled, and even quantified to some extent. This must be balanced with the effectiveness of different sensors. If a number of sensors are present and are helpful, then procedures for viewing all of the sensors and comprehending the information need to be modeled.

4.6 Modeling Different Levels of Autonomy

Previously in this paper we have stated our contention that GOMS would be useful for showing the impact of differing autonomy levels on the user's efficiency. We illustrate this point by showing the difference in workload between extricating a robot when in tele-operation mode versus in escape mode. In tele-operation mode, the user directs all of the robots' actions; it is a complete absence of autonomy. In contrast, once the user puts the robot into escape mode, the robot itself figures out how to move away from all obstacles in the immediate environment and then, once clear of all obstacles, stops to await further commands from the user.

Figure 5 illustrates the portion of the GOMS model that pertains to getting a robot "unstuck": the unenviable condition where it has very little leeway in how it can move. Figure 5 pertains to extricating a robot using Interface A. The model for Interface B is similar (only a few less steps).

Note that Figure 5 employs an informal means of passing a variable to a method. We denote the passing of a variable by a phrase in square brackets.

Method for goal: **get unstuck when tele-operating**
1. Accomplish goal: **determine direction to move**
2. Accomplish goal: **drive to new location**
3. Accomplish goal: **check-movement–related sensors**
4. Return with goal accomplished.

Method for goal: **determine direction to move**
1. Look at and interpret proximity display (**C**)
2. Accomplish goal: **move camera** in direction of obstacle
3. Return with goal accomplished

Method for goal: **move camera** [movement direction]
1. Point to touchscreen button for [movement direction] (**P**)
2. Look at and interpret video window (**C**)
3. Decide: if new movement direction is needed (**C**), go to 1.
4. Return with goal accomplished.

Method for goal: **drive to new location**
1. If hands are not already on joystick, home hands on joystick (**H**)
2. If movement direction not yet known, Accomplish goal: **determine direction to move**
3. Move joystick until new location is reached or until stuck (**S**)
4. If stuck, then accomplish goal: **get unstuck when tele-operating**
5. Return with goal accomplished.

Method for goal: **check movement-related sensors**
1. Look at and interpret video window (**C**)
2. Look at and interpret map data window (**C**)
3. Look at and interpret sonar data window (**C**)
4. Return with goal accomplished.

Figure 5. Model Fragment for Tele-Operating Stuck Robots

As might be expected, getting a robot unstuck can be a tedious process. Not counting the shared method for checking movement-related sensors, there are 16 statements in the methods in Figure 5, and it often takes multiple iterations through most of the steps to completely extricate a robot. Each iteration requires at least 6 **C** operators, a **P,** and possible **H**, in addition to the robot moving time included in the **S** operator—many seconds of user activity, all of it attention-demanding. Our experience is that robots become wedged into tight quarters surprisingly frequently, which motivated the development of the escape mode.

Figure 6 shows in a quantitative manner how much easier it is for users to change the autonomy mode from tele-operation to escape and watch the robot use its sensors and artificial intelligence to move itself from a position almost completely blocked by obstacles to one that is largely free of obstacles. In contrast, the single method shown in Figure 6 lists only 5 statements. This method assumes that the user will do nothing but wait for the robot to finish, but clearly, the user could engage in other activity during this time, opening up other possibilities for conducting the task more effectively.

While this comparison might seem obvious, these models were simple to sketch out, and doing so is a very effective way to assess the possible value of new functionality. Using GOMS could save considerable effort over even simple prototype and test iterations (see Kieras, 2004 for more discussion).

Method for goal: **get unstuck when using escape**
1. Point to touchscreen button for escape (**P**).
2. Wait for robot to finish (**W**).
Same method called as in manual get unstuck method
3. Accomplish goal: **check movement-related sensors**
4. Decide: if robot still stuck, go to 1.
5. Return with goal accomplished.

Figure 6. Model Fragment for Extricating Robots using Escape Mode

5. CONCLUSIONS AND FUTURE WORK

In this paper we have shown how GOMS can be used to compare different interfaces for human-robot interaction. GOMS is useful in determining the user's workload, for example when introducing different displays for sensors.

GOMS models are also useful in determining what the upper and lower time limits are for different procedures. For example, checking movement-related sensors will depend on how many sensor windows the user has to look at and how many of these windows have to be manipulated to be viewed. This can be extremely helpful in deciding what should be visible at what time to maximize the user's efficiency.

GOMS can also be used to evaluate autonomy modes. In our example we used the escape mode on the INL robot to show how autonomy modes can be modeled. In actuality, the escape mode was originally incorporated into robots because it was clear to everyone that enabling this robotic behavior could save the user a lot of work and time. In other words, designers did not need a GOMS model to tell them that such functionality

would be worthwhile. However, this example shows how GOMS can be used to model other autonomy modes to determine possible human-robot ratios. By examining the maximum expected times for different procedures, it is possible to use Olsen and Goodrich's equations on "fan out" (Olsen and Goodrich, 2003) to determine the upper bound on the number of robots that a single person can control simultaneously.

While GOMS was designed to model user behavior with a particular user interface, what it has done in the escape/tele-operation example is to render explicit the effect of both the robotic behavior and the user interface on the user. GOMS could be used in less obvious cases to "test drive" the effects that new robot behaviors, coupled with their interaction mechanisms, might have on users. To the extent that the effects of the interface design and robot behavior can be teased apart, GOMS can thus have utility in the process of designing new robot behaviors.

Future work might productively include experimentation with the effect of degraded video on the time necessary for users to perceive and comprehend information. Simulators could introduce a controlled amount of noise into the videostream in a repeatable fashion so that user tests could yield empirical data regarding average times for comprehending video data under various degraded conditions. This data could be codified into a set of "reference" video images. By comparing a systems' video quality with the reference images, analysts could assign a reasonable estimate of video comprehension times a priori, without further empirical work.

Another future work area might be to examine whether it is useful to employ GOMS to model the robots' performance in interacting with the environment, other robots, and/or humans. Perhaps such an analysis would be desirable to help determine how many autonomous robots would be needed to complete a task under critical time limitations such as a large-scale rescue operation, for example. Such a modeling effort would likely involve a whole new set of issues and would depend on the nature of the robot's environment, the behaviors that were programmed for the robot, and the mechanical limitations inherent in the robot platform.

The use of GOMS models for comparing alternative designs can be much less costly than conducting user testing assuming that little additional empirical testing is needed to determine execution times for domain-specific operators. User testing necessitates building robust versions of user interfaces, obtaining sufficiently trained users, and having robots that are in operating condition for the number of days needed to conduct the tests.

While open issues exist with applying GOMS models for HRI, we are confident that much useful design information can be obtained earlier and with much less effort than user testing.

6. ACKNOWLEDGMENTS

This work was supported in part by NSF (IIS-0415224) and NIST. We would like to thank Douglas Few and David Bruemmer of Idaho National Laboratories; Elena Messina, Adam Jacoff, Brian Weiss, and Brian Antonishek of NIST; and Holly Yanco and Brenden Keyes of UMass Lowell.

7. REFERENCES

[1] Adams, J. A. (2005). Human-Robot Interaction Design: Understanding User Needs and Requirements. In *Proceedings of the 2005 Human Factors and Ergonomics Society 49th Annual Meeting*, 2005, Orlando, FL.

[2] Campbell, C. B. (2002). Advanced integrated general aviation primary flight display user interface design, development, and assessment. In *Proceedings of the 21st Digital Avionics Systems Conference*, Vol. 2.

[3] Card, S., Moran, T., and Newell, A. (1983). *The Psychology of Human-Computer Interaction.* Hillsdale, New Jersey: Erlbaum.

[4] Drury, J. L. and Scholtz, J. (2006). Adapting GOMS to model human-robot interaction. Submitted to the HRI 2007 conference.

[5] Endsley, M. R. (1988). Design and evaluation for situation awareness enhancement. In *Proceedings of the Human Factors Society 32nd Annual Meeting*, Santa Monica, CA, 1988.

[6] John, B. E. (1990). Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. In *Proceedings of the CHI 1990 Conference on Human Factors in Computing Systems.* New York: ACM, pp. 445 – 451.

[7] Hestand, D. and Yanco, H. A. (2004). "Layered sensor modalities for improved human-robot interaction." In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, October 2004.

[8] Irving, S., Polson, P., and Irving, J. E. (1994). A GOMS analysis of the advanced automated cockpit. In *Proceedings of the 1994 CHI conference on Human Factors in Computing Systems*, Boston, April 1994.

[9] John, B. E. (1990). Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. In *Proceedings of the 1990 Conference on Human Factors in Computing Systems.* New York: ACM, pp. 107 – 115.

[10] John, B. E. and Kieras, D. E. (1996a). Using GOMS for User Interface Design and Evaluation. ACM Transactions on Human-Computer Interaction, 3(4), December 1996.

[11] John, B. E. and Kieras, D. E. (1996b). The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. ACM Transactions on Human-Computer Interaction, 3(4), December 1996.

[12] Kieras, D. E. (1997). A Guide to GOMS model usability evaluation using NGOMSL. In M. Helander, T. Landauer, and P. Prabhu (Eds.), *Handbook of Human-Computer Interaction*. (Second Edition). Amsterdam: North-Holland. 733-766.

[13] Kieras, D. E. (2004). Task analysis and the design of functionality. In A. Tucker (Ed.) *The Computer Science and Engineering Handbook* (2nd Ed). Boca Raton, CRC Inc. pp. 46-1 through 46-25.

[14] Kieras, D. E. (2005). Fidelity Issues in Cognitive Architectures for HCI Modeling: Be Careful What You Wish For. In *Proceedings of the 11th International Conference on Human Computer Interaction (HCII 2005)*, Las Vegas, July 22-27.

[15] Kieras, D., and Knudsen, K. (2006). Comprehensive Computational GOMS Modeling with GLEAN. In *Proceedings of BRIMS 2006*, Baltimore, May 16-18.

[16] Leveson, N. G. (1986). "Software safety: why, what and how." *ACM Computing Surveys* **18**(2): 125 – 162, June 1986.

[17] Olsen, D. R., Jr., and Goodrich, M. A. (2003). Metrics For Evaluating Human-Robot Interactions. In *Proceedings of PERMIS 2003*, September 2003.

[18] Rosenblatt, J. and Vera, A. (1995). A GOMS representation of tasks for intelligent agents. In *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms.* M. T. Cox and M. Freed (Eds.), Menlo Park, CA: AAAI Press.

[19] Scholtz, J., Antonishek, B., and Young, J. (2004). Evaluation of a Human-Robot Interface: Development of a Situational Awareness Methodology. In *Proceedings of the 2004 Hawaii International Conference on System Sciences*.

[20] Wagner, A. R., Endo, Y., Ulam, P., and Arkin, R. C. (2006). Multi-robot user interface modeling. In *Proceedings of the 8th International Symposium on Distributed Autonomous Robotic Systems*, Minneapolis, MN, July 2006.

[21] Wood, S. D. and Kieras, D. E. (2002). Modeling Human Error For Experimentation, Training, And Error-Tolerant Design. In *Proceedings of the Interservice/Industry Training, Simulation and Education Conference*. Orlando, Fl. November 28 – December 1.

[22] Yanco, H. A. and Drury, J. L. (2004). "Where Am I?" Acquiring Situation Awareness Using a Remote Robot Platform. In *Proceedings of the 2004 IEEE Conference on Systems, Man, and Cybernetics.*

[23] Yanco, H. A., Drury, J. L., and Scholtz, J. (2004). "Beyond usability evaluation: analysis of human-robot interaction at a major robotics competition." *Human-Computer Interaction*, Vol. 19, No. 1 & 2, pp. 117 – 149.