Multi-robot User Interface Modeling

Alan R. Wagner, Yoichiro Endo, Patrick Ulam, Ronald C. Arkin

Mobile Robot Lab
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0250
{alan.wagner, endo, pulam, arkin}@cc.gatech.edu

Abstract. This paper investigates the problem of user interface design and evaluation for autonomous teams of heterogeneous mobile robots. We explore an operator modeling approach to multi-robot user interface evaluation. Specifically the authors generated GOMS models, a type of user model, to investigate potential interface problems and to guide the interface development process. Results indicate that our interface design changes improve the usability of multi-robot mission generation substantially. We conclude that modeling techniques such as GOMS can play an important role in robotic interface development. Moreover, this research indicates that these techniques can be performed in an inexpensive and timely manner, potentially reducing the need for costly and demanding usability studies.

Keywords. User interface design, Goals Operators Methods Selection Rules (GOMS), Contract Network Protocol (CNP), Case-Based Reasoning (CBR)

1 Introduction

This paper investigates the problem of user interface design and evaluation for teams of heterogeneous, cooperative, and (generally) autonomous mobile robots. As part of NAVAIR's Intelligent Autonomy program, the purpose of this research is to reduce the burden of deploying teams of heterogeneous robots and conducting multi-robot missions. The methods we detail, however, are general purpose and applicable to virtually any software interface. Moreover, we contend that these methods hold special promise for developers of multi-robot systems.

Our approach is motivated by the notable challenges multi-robot systems present to user interface designers. Currently, as the size of the multi-robot team increases, so do the startup, running, and maintenance demands placed on the user [1]. Users can become overwhelmed with situation awareness information in high workload environments. Moreover, because multi-robot applications tend towards mission critical domains, such as search and rescue and military domains, users must be capable of quickly and easily assessing important situation information while also being shielded from insignificant or redundant information.

This paper details a modeling approach to multi-robot user interface evaluation. This approach is used to assess the effectiveness of novel interface and algorithmic changes made to an existing multi-robot software toolset. Roboticists have traditionally conducted usability experiments to gauge the state of their interface designs [2]. Usability experiments, however, are expensive, time-consuming, and too cumbersome for effective interface development [3]. Moreover, for specialized software, such as multi-robot mission specification systems, usability testing typically requires difficult to find domain experts to assess the interface design. Alternatively, random or semi-random subject populations are used. These populations, however, tend to have little or no experience with the application in question, whereas, in reality, the target population may have significant experience with either previous versions or similar types of software. These reasons serve as a motivation to explore alternatives to formal usability studies such as GOMS (Goals, Operators, Methods and Selection rules) modeling [4].

2 Related Work

At least two aspects of user interface design differentiate existing evaluations in robotics from those in software engineering and HCI. First, within robotics researchers typically evaluate user interface designs at the system architecture level (e.g., [5]), rather than at the level of particular algorithms and features. Some exceptions exist [2]. Software engineering practices and Human Computer Interaction (HCI) researchers, on the other hand, routinely examine the user interface implications of specific changes to existing software packages (e.g., [6]) as well as conducting system level evaluations. Second, usability studies represent the majority of formal user interface evaluations within robotics [2]. The software engineering and HCI disciplines, on the other hand, employ a wider array of user interface techniques including heuristics and user modeling [5]. A GOMS model is a type of user model that describes the knowledge a user must posses in order to perform tasks with the system [3]. Research by Yanco et al. represents the only example of GOMS based user interface design evaluation within robotics located to date [7]. Yanco et al. focus on the challenge of a specific robotics domain and a usability coding scheme inspired by GOMS. Our intention, rather, is to explore the use of GOMS as a primary means of evaluating incremental additions to a multi-robot user interface.

3 User Interface Modeling

A GOMS model explicitly represents the knowledge that a user must have in order to accomplish goals using an interface [3]. Natural GOMS Language (NGOMSL) is one method for explicitly representing GOMS models [8]. NGOMSL is a structured language notation in program form (see examples below). As a knowledge representation, a GOMS model can also serve to characterize ongoing user decisions or as a description of what a user must learn. Moreover, because user goals tend to be constrained by interface design, GOMS models can quantitatively predict aspects of usability such as the efficiency and simplicity of procedures.

A user interface analyst conducts a GOMS analysis by a describing in detail the goals, operators, methods, and selection rules a user must follow for a set of tasks. A goal is something that a user tries to accomplish. For example, one goal resulting from our GOMS analysis is to edit the parameters of a robot behavior. An operator is an action that a user executes. An example of an operator is moving the cursor to a screen location. A method is a sequence of operators for accomplishing a goal. The following example method (in NGOMSL) accomplishes the goal of adding a mission behavior:

```
Method for goal: add behavior                KLM op  Time(s)
  Step 1: Locate add behavior icon             M       1.2
  Step 2: Move cursor to add behavior icon     P       1.1
  Step 3: Click mouse button                   BB      0.2
  Step 4: Think-of new icon location           M       1.2
  Step 5: Move cursor to new location          P       1.1
  Step 6: Click mouse button                   BB      0.2
  Step 7: Return with goal accomplished      Total     3.0
```

Finally, a selection rule (also in NGOMSL) routes control to the correct method for accomplishing a goal when many possible methods are possible.

A GOMS analysis begins by first describing a top-level goal and its associated high-level operators and by then iteratively replacing these operators in a breadth-first manner with methods and selection rules that accomplish each goal until all of the operators are primitive and cannot be further analyzed. The analyst may choose his or her own primitive operators, but typically, standard primitive operators from the Keystroke-Level Model (KLM) are used [3]. These primitives offer a well-documented mean time of operation and in some cases functional estimates.

Once the GOMS analysis is complete, the analyst can perform a qualitative evaluation of the interface to examine the efficiency, consistency, and cleanliness of the design. The analyst can also use the GOMS model to estimate the execution time of specific user tasks and the amount of effort that it will take users to learn procedures represented in the model.

GOMS modeling, however, is not without limitations. The execution and efficiency predictions generated from a GOMS model assume error-free performance. A GOMS model therefore represents a best-case evaluation of an interface. Still, GOMS models provide a valuable baseline for comparison of interface changes. GOMS modeling can also require subjective decisions and judgment calls. In spite of these subjective decisions, the analyst objectively constructs the majority of the model based on the actual state of the interface design. Overall, GOMS modeling serves more to guide interface development than to completely replace usability testing [6].

4 A Case Study in Multi-robot User Interface Modeling

GOMS modeling has been successfully employed within HCI and software engineering [7, 9], but is relatively unknown within robotics [7]. It is our contention that GOMS assessments of multi-robot user interface designs could play a vital role in the generation and rapid prototyping of future multi-robot system interfaces. To explore this hypothesis we conducted a detailed GOMS analysis of features recently added to the *MissionLab* toolset [9].

MissionLab allows users to generate multi-robot missions in the form of a FSA (Finite State Acceptor) in which nodes representing the robot's behaviors are connected via directed edges representing the robot's perceptual trigger schemas. The FSA serves as a flexible robot mission and can be stored, copied, or edited as needed to generate novel missions. This software system also features a Case-Based Reasoning (CBR) wizard that abstracts entire multi-robot missions as cases to be matched to the user's needs (Figure 1 left) [2]. The CBR wizard can also use cases as high-level drag-and-drop robot tasks, hence simplifying the mission creation procedure.

Our current investigation considers a scenario where multiple, heterogeneous robots are available for tasking. With respect to mission generation, multi-robot tasking presents additional challenges to the user. In a system of many robots, or when each robot affords unique capabilities, the user may have to assign each task to a robot. The generation of a multi-robot mission, in this case, demands (1) the user delineate the tasks necessary for the mission, and (2) the user assign each of these tasks to a specific robot or robots. The CBR wizard eases the first challenge but does not assist with the second.

To manage these challenges we have developed a novel method for generating multi-robot missions which employs a Contract Net Protocol (CNP) working in conjunction with the CBR Wizard to reduce the burdens placed on the user (see [10] for a review of multi-robot CNP). In its most general form, CNP is an auction-style algorithm in which the robots of a multi-robot system produce bids based on their estimate of their ability to perform the auctioned task. Typically, when the auction closes CNP assigns the highest bidder the task. Our system uses CNP as a method to aid the user by assigning robots to specific tasks prior to the start of the mission. In this role, the goal of the pre-mission CNP system is thus to generate an a priori mapping of robots to available pre-mission tasks.

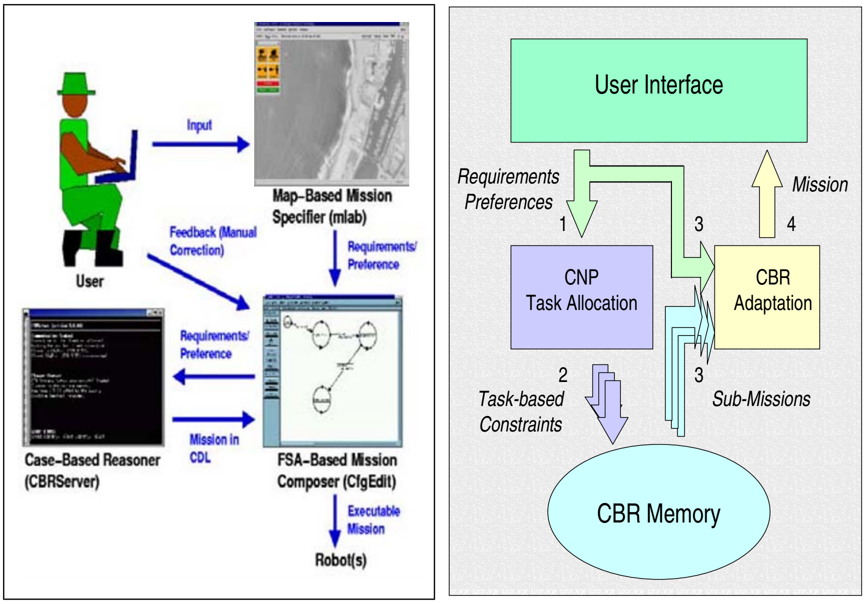

Fig. 1. Integrated CBR-CNP system for multi-robot mission generation

This system operates by first relaying a set of robot requirements and task requirements to a CNP task allocation component (Figure 1 right). The CNP component then conducts an auction resulting in a robot-to-task mapping. The system then uses this mapping to retrieve a sub-mission from CBR memory for each robot. Using additional user task preferences, these sub-missions are adapted into a single full mission. Finally, the system presents the complete mission with robot-to-task assignments to the user for acceptance, alteration, or rejection.

To assess the usability of the new features the authors created two GOMS models, one for the base system (no CBR or CNP) and one for the integrated CBR-CNP system. Both models evaluate the general methods available to users for creating multi-robot multi-task missions, beginning with the decision to build a new mission and ending with a complete mission. In some cases, the analyst made judgment calls concerning which GOMS operator to use or execution time estimates. As much as possible the analyst strived to maintain consistency across both models. Each model required approximately 20-30 hours to construct and was created by the lead author using guidelines available from [3]. Additional supplementary data and both complete models are available at www.cc.gatech.edu/ai/robot-lab/onraofnc/data/goms-2005/.

5 Results

5.1 Mission Generation Time Predictions

The time required to generate a mission is determined from the method and operator execution times in the GOMS model [3]. The base system GOMS model predicts the generation time of a multi-robot multi-task mission to be a function of both the number of robots in the mission and the complexity of the tasks each robot is to perform. The mission generation time of the base system in seconds, t_{gb}, is predicted to be:

$$t_{gb}(n, g, h) = A + Bn + Cgn + Chn \qquad (1)$$

where n is the number of robots, g is the average number of behaviors per task, and h is the average number of triggers per task. Table 1 lists model coefficients. Thus, 78.3 seconds are required to generate any mission without regard to the number of robots or the complexity of the mission. The mission generation time incurs a further cost of 26.7 seconds for each additional robot represented by the second term. The number of behaviors and triggers composing a task is also expected to have large impact on the mission generation time. As shown by equation (1) 34.8 seconds are necessary per robot and per behavior or trigger. Alternatively, one can estimate the mission generation cost as:

$$t_{gb}(n, \tau) = A + Bn + 2Cn\tau \qquad (2)$$

where τ is the number of tasks. Equation (2) assumes that all tasks require the same number of behaviors and triggers ($2\tau = g + h$) and that a single task is equivalent to a single behavior and a trigger. This, however, is generally not the case and equation (2) is offered solely for comparison to the integrated CBR-CNP GOMS model.

Table 1. Model coefficient values.

	Initial Model Coefficient Values	Refined Model Coefficient Values
A	78.30	74.22
B	26.70	18.28
C	34.80	32.08
D	85.30	77.92
E	24.30	19.56

The integrated CBR-CNP GOMS model, on the other hand, predicts the generation time of a multi-robot multi-task mission will only be a function of the number of tasks. In this case, the mission generation time in seconds, t_{gi}, is governed by:

$$t_{gi}(\tau) = D + E\tau \tag{3}$$

where τ is the number of tasks. The integrated CBR-CNP model incurs a 7.0 seconds greater startup cost ($D - A$). This cost is primarily due to the need to select the option for CBR-CNP. Users incur a further cost for each task. Because the integrated system abstracts from the user the assignment of each robot to a task, mission generation is independent of the number of robots. Moreover, comparing equations (2) and (3), we note that the models predict that the integrated CBR-CNP system requires approximately 44.3 seconds less per task ($2C - E$; assumes a single robot) than the base system given that the assumptions mentioned above hold.

5.2 Learning Effort Predictions

The effort required to learn how to generate a mission is determined from the number and length of the methods in the GOMS model [3]. User learning effort is estimated from the number of NGOMSL statements in each model. The total number of NGOMSL statements in the model describes the amount of procedural knowledge a user must have in order to use all aspects of the software system. User training describes the process of learning this procedural knowledge. Hence, models with fewer individual statements require less effort to learn.

The GOMS model of the base system (no CBR or CNP) included 187 individual statements that encompassed the procedures necessary for creating a multi-robot multi-task mission. The GOMS model of the integrated CBR-CNP system, in contrast, included 147 operators. We, therefore, expect the base system to require approximately 21.3 % more procedural knowledge.

We did not conduct experiments to confirm this result. However, if one assumes that additional procedural knowledge results in less accuracy, then this result corroborates related prior usability studies conducted by our lab [2]. This earlier work examined the use of the CBR wizard for a variety of mission generation tasks and found that it improved the accuracy of mission generation on tasks requiring two robots by approximately 33%. Our GOMS models indicate that this increase in accuracy may partially result from the reduced workload on the user. Our models also predict that less accuracy will be gained when generating a single robot mission compared to a multi-robot mission. This was also found to be the case in [2].

5.3 Comparison of Model Estimations to Actual Expert Performance

The results from the previous two sections clearly and quantitatively indicate the value of the CBR-CNP. As far as the case study is concerned, these results are sufficient. We decided, however, to also investigate the methodology itself by examining the accuracy of the predicted execution times for both GOMS models. In particular, we hoped to determine (1) if the primitive KLM operators used for the models accurately reflected experimental values for expert users and (2) if these primitive operators are immune to experimenter bias. To accomplish this, we conducted an experiment involving system experts. These experiments attempted to gauge the accuracy of both the overall models and of several GOMS methods that could then use to refine the models. The experiment required the expert to create multi-robot multi-task missions 20 times using both the base system and the integrated CBR-CNP system. During data collection, time data was recorded related to all of the users' actions. The experts used for the study consisted of six members of the same research lab including three authors of this paper. We hypothesized that because GOMS operators consist of low-level primitives such as individual key strokes that experimenter bias would be minimal. Moreover, the authors decided which GOMS methods to compare after experimentation but before analyzing the data. Thus, no subject knew which part of the experiment would be used.

Table 2. Predicted and actual execution times.

	Predicted Exec. Time (s)	Empirical Exec. Time (s)	Refined Exec. Time (s)
Base System	1662.9	187.80 ± 20.02	1522.3
CBR-CNP	133.9	53.96 ± 7.75	117.04

Table 2 lists the mean execution times for both the integrated and base systems. The models predict that the execution time for approximately the same mission ($n = 2, g = 11, h = 11, \tau = 2$) will require about 12.4 times the execution time on the base system than on the integrated CBR-CNP system. Empirical results indicate that the actual execution times are less for both systems (based on five of six subjects). These experiments reveal that the execution time on the base system requires approximately 3.5 times the execution time compared to the integrated CBR-CNP system.

There are several possible reasons for the discrepancy between the predicted and empirical results. First, some primitive operator execution times may not be correct for this particular experiment. Gong and Kieras found that mouse movements are more accurately estimated from Fitts' Law than by the Keystroke-Level Model (KLM) time of 1.2 sec used here [11]. Second, the base system model does not assume that the user will use shortcuts although some subjects did. Third, neither model factors in the performance gains associated with repeatedly constructing the same missions. Regardless of the precise gains in performance realized by one system over the other, the most important point is that the GOMS models accurately indicate the utility of the CBR-CNP interface changes.

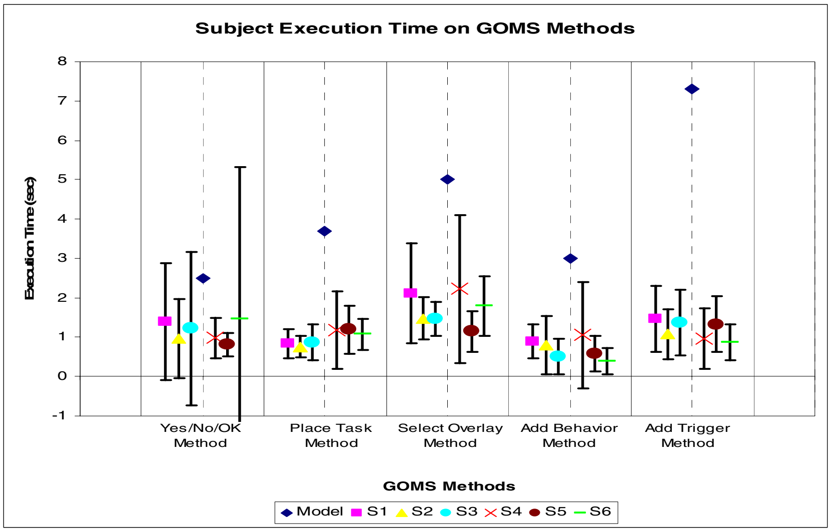

Fig. 2. A scatter diagram of the experimental execution time for expert users on five GOMS methods is depicted. Error bars represent 95% CI. The two rightmost methods are from the base model. The next two methods are from the CBR-CNP model and the leftmost method occurred in both models. The blue diamond on the dashed line depicts the execution time for each method predicted by the GOMS models. The left of each dashed line depicts the actual time for three experimenters. The right of each dashed line depicts the time for experimentally naïve expert users.

Figure 2 depicts the predicted execution times and actual execution times for several arbitrarily selected methods in the GOMS models. The independent variable represents the GOMS method selected and the dependent variable describes the subject's execution time. The `Yes/No/OK Method` (see appendix for GOMS methods) occurs in both GOMS models and experimental conditions. The `Place Task Method` and the `Select Overlay Method` occur in only the CBR-CNP model and experimental condition. Finally, the `Add Behavior Method` (presented in section 3) and the `Add Trigger Method` occur in the base system model and experimental condition. The dashed line denotes the execution time predicted by the models. The subjects to the left of each dashed line are also the experimenters. The subjects to the right of the dashed line are naïve subjects. The figure shows that the execution time predictions from our initial model are significantly greater then the actual expert execution times. As a result, we can now revisit the model and update the execution times for greater accuracy. Table 1 compares the initial model coefficients to refined model coefficients. Table 2 presents execution times based on these refined models. The predicted execution times for both refined models are closer to the actual execution times.

Several other points are also of interest. First, as indicated by figure 2, no experimenter bias is apparent. This is important because it increases the subject pool for potential user interface experiments. Hence, it appears that expert subjects can be drawn from the authors of the study itself, given the restrictions outlined above; possibly further reducing the challenge of user interface evaluation. Second, GOMS experiments are robust to experimental error. A data collection error occurred for one subject (*S4*) and another subject misunderstood the directions (*S3*). The data collection error (*S4*) resulted in elimination of this subject's mission execution time results (in Table 2) but did not affect the subject's data collected while completing the GOMS methods (data in Fig. 2). Subject 4's misunderstanding of the directions resulted in fewer data points for the `Add Behavior Method` (~150 versus ~220 normal). In spite of this experimental error, enough data was collected to produce statistically significant conclusions. Usability studies often face similar challenges and must completely exclude data from some subjects due to errors such as these. Nevertheless, because GOMS models are constructed from low-level user interface primitives, data from these subjects could still be salvaged.

Overall, both GOMS models and our experimental results indicate the value of the CBR-CNP user interface. Moreover, our modeling results and empirical results can be used for additional future interface design evaluations of this system.

6 Conclusions

This paper has investigated user interface modeling as a method for evaluating multi-robot interface design. We compared two GOMS models, one representing the base system with additional features for multi-robot multi-task mission generation and the other without. Our results indicate that these new multi-robot mission generation features substantially improve the usability of the *MissionLab* software. We intend to evaluate future interface designs using the same techniques, which may include the construction of a GOMS library of expert operator execution times. This should aid in the construction of more accurate future models.

We believe, and our case study has shown, that modeling techniques such as GOMS can play an important role in robotic interface development. Moreover, our work indicates that researchers can perform these techniques in a relatively inexpensive and timely manner. It is our sincere hope that other robotics researchers will consider the lessons described here, and in detail by HCI specialists [3], when designing user interfaces for critical robot applications operating in hazardous environments.

Acknowledgements

This research is funded as part of Navair's Intelligent Autonomy program. We would like to thank Ed Clarkson for lending his HCI expertise and Zsolt Kira, Lilia Moshkina, and Eric Martinson for assisting with this research.

References

1. J.A. Adams. (2002) Critical Considerations for Human Robot Interface Development, *Proceedings of 2002 AAAI Fall Symposium*, 1-8.

2. Y. Endo, D.C. MacKenzie, and R.C. Arkin. (2004) Usability Evaluation of High-Level User Assistance for Robot Mission Specification, IEEE Transactions on Systems, Man, and Cybernetics, **34**, 168-180.

3. D. Kieras (1994). *A guide to GOMS model usability evaluation using NGOMSL*, In M. Helander & T. Landauer (Eds.), The handbook of human-computer interaction. (Second Edition), North-Holland Amsterdam.

4. S.K. Card T.P. Moran, and A.P. Newell (1983) The Psychology of Human-Computer Interaction, Lawrence Erlbaum, Hillsdale, N.J.

5. D.C. MacKenzie and R.C. Arkin. (1998) Evaluating the Usability of Robot Programming Toolsets, International Journal of Robotics Research, **17**(4), 381-401.

6. B.E. John and D.E. Kieras. (1996) Using GOMS for User Interface Design and Evaluation: Which Technique?, ACM Transactions on Computer-Human Interaction, **3**(4), 287-319.

7. H. A. Yanco, J. L. Drury, and J. Scholtz. (2004) Beyond usability evaluation: Analysis of human-robot interaction at a major robotics competition, *Journal of Human-Computer Interaction*, 19, 117-149.

8. B.E. John and D.E. Kieras. (1996) The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast, ACM Transactions on Computer-Human Interaction, **3**(4), 320-351.

9. D.C. MacKenzie, R.C. Arkin, and J.M. Cameron. (1997) Multiagent Mission Specification and Execution, Autonomous Robots, **4**, 29-52.

10. M.B. Dias, R.M. Zlot, N. Kalra and A. Stentz. (2005) Market-Based Multirobot Coordination: A Survey and Analysis, Carnegie Mellon University Tech Report CMU-RI-TR-05-13.

11. Gong, R., and Kieras, D. (1994) A Validation of the GOMS Model Methodology in the Development of a Specialized, Commercial Software Application. In *Proceedings of CHI,* Boston, MA, USA, pp. 351-357.

Appendix

```
Method for goal: select yes-no-ok                KLM op Time(s)
 Step 1: Locate yes-no-ok button                     M       1.2
 Step 2: Move cursor to yes-no-ok button             P       1.1
 Step 3: Click mouse button                          BB      0.2
 Step 4: Return with goal accomplished               Total   2.5
                                             Refined Total 1.14
Method for goal: place task                      KLM op Time(s)
 Step 1: Think-of placement point                    M       1.2
 Step 2: Move cursor to placement point              P       1.1
 Step 3: Click mouse button                          BB      0.2
 Step 4: Verify that placement point is correct      M       1.2
 Step 5: Return with goal accomplished               Total   3.7
                                             Refined Total 0.91
Method for goal: select overlay                  KLM op Time(s)
 Step 1: Locate name in the file list box            M       1.2
 Step 2: Move cursor to the file name location       P       1.1
 Step 3: Click mouse button                          BB      0.2
 Step 4: Locate ok button                            M       1.2
 Step 5: Move cursor to ok button                    P       1.1
 Step 6: Click mouse button                          BB      0.2
 Step 7: Forget overlay name and return              Total   5.0
                                             Refined Total 1.82
Method for goal: add trigger                     KLM op Time(s)
 Step 1: Locate trigger icon                         M       1.2
 Step 2: Move cursor to trigger icon location        P       1.1
 Step 3: Click mouse button                          BB      0.2
 Step 4: Locate trigger tail behavior location       M       1.2
 Step 5: Move cursor to trigger tail behavior        P       1.1
 Step 6: Press mouse button down                     B       0.1
 Step 7: Locate trigger tip behavior                 M       1.2
 Step 8: Move cursor to tip behavior                 P       1.1
 Step 9: Release mouse button                        B       0.1
 Step 10: Return with goal accomplished              Total   7.3
                                             Refined Total  0.87
```