Usability Evaluation of an Automated Mission Repair Mechanism for Mobile Robot Mission Specification

Lilia Moshkina Yoichiro Endo Ronald C. Arkin

Georgia Tech Mobile Robot Lab
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280, USA
{lilia, endo, arkin}@cc.gatech.edu

ABSTRACT

This paper describes a usability study designed to assess ease of use, user satisfaction, and performance of a mobile robot mission specification system. The software under consideration, *MissionLab*, allows users to specify a robot mission as well as compile it, execute it, and control the robot in real-time. In this work, a new automated mission repair mechanism that aids users in correcting faulty missions was added to the system. This mechanism was compared to an older version in order to better inform the development process, and set a direction for future improvements in usability.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces – *evaluation/methodology, graphical user interfaces.*

General Terms

Human Factors.

Keywords

Human-robot interaction, usability study, mission specification.

1. INTRODUCTION

In military and other critical applications, it is vital for users with little or no robotic background to quickly and accurately create and execute mobile robot missions, and the importance of empirically evaluating the interfaces that support this task cannot be underestimated [1, 2]. For example, researchers have conducted usability studies in critical robotic applications such as in the area of Search and Rescue [3, 4] and on teleoperation in nuclear environments [5-7], dealing with run-time interaction with the robots.

Throughout the development of *MissionLab*, which is a robot mission specification system designed to support planning and execution of robot missions using a graphical user interface (GUI), we have been also conducting a series of human-subject studies in order to both evaluate the current state of the software and to inform its further design and development [8-11].

Over the past few years, the user interface of *MissionLab* has steadily progressed from the lower level manual mission specification system via Finite State Acceptor (FSA) [12] to higher-level case-based reasoning mission retrieval system. The previous usability study [10] compared the lower-level approach with the higher-level one, and it was found that the automated mission retrieval version of the user interface reduced the time required to plan a robot mission and increased the accuracy of the resulting mission plans for complex missions, as compared to the earlier, lower-level interface. Given the encouraging results, we continued abstracting low-level details, moving from the mission creation to repair of faulty missions.

In our latest usability study we assessed the newest addition to *MissionLab*, a feature that assists users in repairing faulty missions by abstracting low-level details and making automated suggestions based on user input. The interface was evaluated according to its effectiveness, efficiency and user satisfaction – measures suggested for human-robot interaction research [4]. The effectiveness was measured in terms of the accuracy of competed missions and the number of missions completed successfully, efficiency in terms of the time taken to create and modify each mission, and user satisfaction was divided into user satisfaction proper and ease of use.

An overview of *MissionLab* including its new Mission Repair feature is presented in the next section. The description of the latest usability study conducted on the Mission Repair feature is described in Section 2. Finally, conclusions and future work are presented in Section 3.

2. MISSIONLAB

MissionLab [13] is a software package that allows a user to create an autonomous mobile robot mission (sequence of tasks) as well as compile it, execute it, and control the robot during the run-time. Previously [10], we introduced the high-level user assistance interface in the *MissionLab* that was developed to ease the mission specification process by retrieving previously saved successful missions. The Case-Based Reasoning (CBR) Wizard described here is an extension of the interface. The most prominent improvement over the previous version is the addition of a new feature that could help users repair faulty missions. In this section, we first review the basics of the CBR Wizard, and then describe the repair capability.

2.1 CBR Wizard Basics

The diagram in Figure 1 illustrates a CBR cycle that was adapted from [14] for robotic mission specification. In the pre-mission phase, the CBR Wizard first retrieves ballpark solutions (cases) from its memory that match the requirements and preferences (constraints) specified by the user. Here, the constraints may be a type of tasks to be performed (Hostage Monitoring, Explosive

Ordnance Disposal, etc.) or whether the mission should be conducted indoor or outdoor. Once the ballpark solutions are retrieved, the CBR Wizard then adapts them to fit the current situation by, for example, matching the number of robots or adjusting the coordinates of target points.

In terms of *MissionLab* system, the CBR Wizard involves three UNIX processes during the pre-mission phase: namely, *mlab*, *CBRServer*, and *CfgEdit* (Figure 2). *mlab* is where constraints of a new mission are specified by the user using its map-based interface. *CBRServer* serves as database for mission templates (cases) as well as a planner that customizes a mission for the current specification. *CfgEdit* is a user interface that can display the mission graphically and allows the user to modify it manually if necessary.

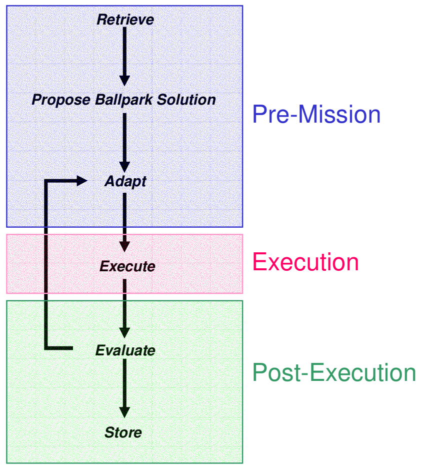


Figure 1: CBR Cycle for Robotic Mission Specification (adapted from [14])

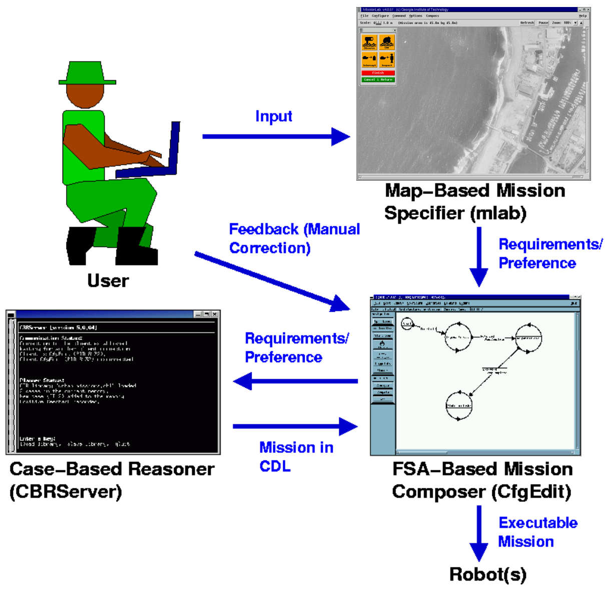

Input
Map–Based Mission Specifier (mlab)
Feedback (Manual Correction)
Requirements/ Preference
User
Requirements/ Preference
Mission in CDL
Case–Based Reasoner (CBRServer)
FSA–Based Mission Composer (CfgEdit)
Executable Mission
Robot(s)

Figure 2 Three processes of the CBR Wizard in *MissionLab*: namely, *mlab*, *CBRServer*, and *CfgEdit*.

The diagram in Figure 3 depicts three modules inside *CBRServer* involved in the pre-mission phase. (The shaded area is the process of *CBRServer*.) After the constraints/preferences are specified by the user, they are sent down to Memory Manager where a number of mission templates are stored. Here, the mission templates are abstract representations of missions that are saved previously, whether by experts or users. Mission Manager utilizes standard decision trees [15] to manage the storage of the mission templates (cases). The constraints specified by the user are thus used as indexes to retrieve the mission templates from the trees. Once relevant mission templates are retrieved, Planner assembles them and customizes their parameters to make them fit with the current specification of the mission. At this point, the missions are still coded with abstract representations; Domain Manager then translates those into the language (Configuration Description Language [13]) that *CfgEdit* operates with.

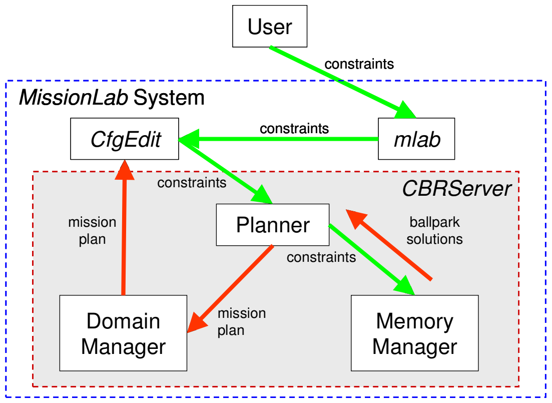
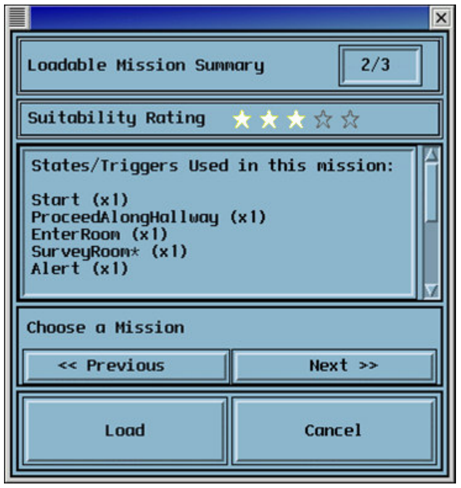


Figure 3: Pre-Mission Phase (Shaded area represents *CBRServer*)

Note that depending on the constraints and the number of relevant cases stored in the memory, multiple missions may be suggested by *CBRServer*. To help users decide what missions to load into *CfgEdit*, the CBR Wizard provides a ranking associated with each loadable mission. As shown in Figure 4, the ranking is quantified with five stars. If the retrieved mission is ranked with five stars, it is considered to be best suited for the desired mission. The ranking may be degraded if: a) The mission does not exactly match the constraints specified by the user; and/or b) executing this mission produced an unsuccessful result, which was marked by the CBR Wizard based on the user's feedback in the previous cycle.


Loadable Mission Summary 2/3

Suitability Rating ★★★☆☆

States/Triggers Used in this mission:

Start (x1)
ProceedAlongHallway (x1)
EnterRoom (x1)
SurveyRoom* (x1)
Alert (x1)

Choose a Mission

Figure 4: Suitability rating of a mission (3 out of 5 stars)

2.2 Repairing Faulty Missions

The subject of this usability study is a newly integrated feature of the CBR Wizard that allows the user to repair faulty missions during the post-execution phase. Here, *faulty mission* means that execution of this mission did not end with successful results. The CBR Wizard attempts to repair faulty missions with two steps: 1) Identifying the offending part of the faulty mission; and 2) retrieving and applying a repair rule to fix the offending part of the mission. It should be noted, however, that different users may have different intentions upon specifying a mission. Determining whether a mission is successful or unsuccessful is largely up to the user, and fully automating this process can be formidable. Hence, the CBR Wizard attempts to accomplish the task by engaging in a dialogue with the user.

First, identification of the offending part in a faulty mission is accomplished via the Playback Interface of *mlab* (Figure 5). After loading a log file that was recorded during the execution of the mission, the user can play back the mission by using the interface similar to that of a VCR or CD player; hence, the user can observe the trajectory of the robot during the mission. The user is then asked by the CBR Wizard to stop the robot at the place where the mission was considered to have failed. By parsing the logged information for that instance of the mission, the CBR Wizard identifies the state (i.e., offending part of the mission) that robot was in when the failure occurred. Once the CBR Wizard identifies the offending part of the mission, the mission is sent back to Planner of *CBRServer* for its repair.

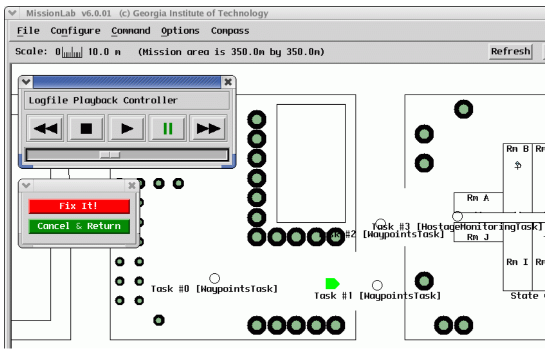

MissionLab v6.0.01 (c) Georgia Institute of Technology
File Configure Command Options Compass
Scale: 0 10.0 m (Mission area is 350.0m by 350.0m)
Refresh
Logfile Playback Controller
Fix It!
Cancel & Return
Rm B
Rm A
Rm J
Rm I
Task #3 [HostageMonitoringTask]
#2 [WaypointsTask]
Task #0 [WaypointsTask]
Task #1 [WaypointsTask]
State

Figure 5: Playback Interface

The next step is to retrieve a repair plan that is a set of rules to fix the offending part of the faulty mission. As shown in Figure 6, the exact infrastructure used in the mission retrieval can be also utilized for this process. Before retrieving the repair plan, the CBR Wizard first asks the user a few predefined questions that can categorize the nature of the failure. The answers to the questions are then itemized as a set of constraints to retrieve the repair plan that is stored within the decision trees (different from the ones for the mission templates) in Memory Manager. Currently, the CBR Wizard supports three types of repair plans: 1) parametric adjustment; 2) structural modification; and 3) functional modification. More specifically, the parametric adjustment is designed to repair a faulty waypoint-following mission (e.g., reconnaissance mission) when at least one waypoint was placed incorrectly. The repair plan attempts to fix the mission by adjusting the coordinates of the waypoint. The structural modification is for the case when a robot failed to take some necessary action during the execution. It attempts to resolve the problem by inserting a new state that commands the robot to take the necessary action. The functional modification is designed to help the case when a robot performed a wrong action at some point during the execution. The repair plan for this case is to replace the offending state of the mission with a correct one.

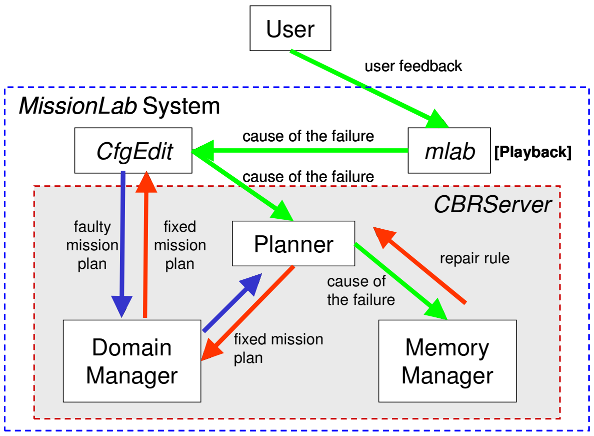


Figure 6: Repairing of a faulty mission

Once an appropriate repair rule was found, Planner applies it to the offending part of the mission, and Domain Manager translates it into CDL, so that *CfgEdit* can load the newly fixed mission.

3. USABILITY STUDY

This usability study was designed to assess the feasibility of the new *Mission Repair* feature of the CBR Wizard described above. In this section, both preparation of the usability study, and the results of the study are presented.

3.1 Experiment Design

The study followed a mixed 2x3 design: the factor in the between-subject condition is the presence of the Mission Repair mechanism, and the factor in the within-subject condition is the nature of the modification task. The between-subject factor will be referred to as "Repair Method" in the remainder of the paper, and the within-subject factor will be referred to as "Task". The participants were divided into two groups (15 subjects each), according to the Repair Method. For the first group (No-Repair Condition, Group A), the experiments were carried out using the CBR Wizard without the Mission Repair feature (control condition). The second group (Repair Condition, Group B) used the Mission Repair feature.

The following three modification task types were tested in the within-subject design: 1) parametric adjustment; 2) structural modification; 3) functional modification. In the parametric adjustment test (referred to as "Waypoint Navigation", or WP, mission further), the users were presented with a task requiring them to reroute a waypoint (a.k.a., adjust waypoint task parameters) in order to direct the robot on an appropriate path. In particular, enemy robots were introduced during run-time which prevented the robot from following the initially specified path. The user had to modify the existing path so that the robot would reach its final destination without being exposed to enemy robots. In the structural modification test (referred to as "Biohazard", or BH, mission further) the users were presented with a task that would require them to find and add a missing state to the mission plan (a.k.a., modify the structure of the FSA). For this test, the users were instructed to guide a single robot into a potentially contaminated building and perform a biohazard survey task. Finally, in the functional modification test (referred to as "Hostage Monitoring", or HM, mission further) the users were presented with a task that required them to find an incorrectly used task and replace it with the correct one (a.k.a., modify the function of one of the tasks). For this test, the users were instructed to guide a single robot into a hostage monitoring site and perform a hostage monitoring task. In all the tests in the Repair condition the users were instructed to employ the Mission Repair feature of the software to modify the faulty missions; however, the functionality of modifying the FSA manually was available to them as well.

The within-subject tests were designed in order to generalize the results of the study across a number of tasks, as well as to identify any differences between them. The within-subject runs were counter-balanced to minimize practice effects. One-way ANOVA was performed on all of the measures by the order the tests were presented to participants, and did not yield any significant results, suggesting that the test order was counterbalanced successfully.

3.2 Hypotheses

The following form the hypotheses of expected usability improvements, and were used to guide the experiment to measure quantitatively just how much, if at all, the Mission Repair feature improves the usability of *MissionLab*.

Hypothesis 1: The CBR Wizard with Mission Repair feature reduces the time required to create a suitable mission plan, including the necessary modifications to failed plans.

Hypothesis 2: The CBR Wizard with Mission Repair feature increases the accuracy of mission plans created by users.

Hypothesis 3: Users find it easier to create and modify a mission plan using the CBR Wizard with Mission Repair feature than without such a feature.

Hypothesis 4: The level of user satisfaction with the Mission Repair feature is higher than without it.

We were also hypothesizing that these statements will hold true for each type of modification task, and that there will be no differences in the within-subject factor.

3.3 Study Setup and Procedure

The experimental equipment was set up in a quiet small office. A Dell Precision 610 desktop computer with a separate monitor, keyboard, and three-button mouse was used by the subjects. A version of *MissionLab,* capable of both enabling and disabling the Mission Repair feature, was installed on the computer. In order to ease the data analysis, all the subject logs were stored systematically in separate directories, indexed by the date, subject number, and type of tests; and the computer screen output during the tests was captured on videotape.

Each subject participated in a single session lasting up to five hours. They were first given four computer-based tutorials explaining how to create and repair simulated robot missions using the *MissionLab* software on the Dell desktop; the last of the tutorials was hands-off, where they were given an opportunity to create a mission without the administrators' help, and then were explained the correct solution. After a mandatory 10-minute break the subjects were given three tests in which they were asked to create and modify (if needed) simulated robot missions (the order was counterbalanced). There were 5-10 minute breaks after the 2nd and 4th tutorials, as well as mandatory 10-minute breaks between each of the tests. Each test was over either when a successful mission was produced (the users were presented with the success criteria for each test) or at the end of 45 minutes. The 45-minute period was based on the average time required by the subjects creating similar missions in the previous TMR usability study [8].

In addition to the tutorials and the tests, the subjects were also asked to fill out two questionnaires: demographics and post. Demographics questionnaire was designed to acquire the background information of the subject, to include gender, and technical, military, and computer experience; it was filled out at the beginning of the session. Post questionnaire was designed to obtain opinions on how easy it was to create and modify mission plans using *MissionLab* as well as to elicit the level of user satisfaction. This questionnaire was given after all the tests were completed.

3.4 Measures

Total actual duration of each test was used as a measure to evaluate *Hypothesis 1*. To more accurately approximate the time the users were engaged in the actual mission creation and modification activity, the total actual duration was calculated by subtracting compilation and simulation time based on the time stamps of various keyboard and mouse events recorded during the tests.

The accuracy of successful mission plans created by the subjects (after modification) served as a measure to evaluate *Hypothesis 2*. The criteria of "success" were based on various mission-specific aspects (e.g., "the robot reached the designated area", "the robot found the biohazard", etc.), clearly described in the instructions for each test. The accuracy was measured as a percentage of the specific aspects completed per mission; both Hostage Monitoring and Biohazard mission included 7 such aspects, and Waypoint Navigation mission had 3 aspects. As a significant portion of missions was competed successfully (a.k.a., the accuracy equalled 100%), we also computed the percentage of missions completed successfully per test per between-subject condition, referred to later as "Performance".

In order to analyse *Hypotheses 3 and 4*, the subjects were asked to fill out the post-questionnaire. There were 10 questions total, equally divided between those assessing ease of use and satisfaction. Five out of 10 questions assessed the Ease of Use, and were as follows:

1. It was easy to create a robot mission using *MissionLab*;
2. If I wasn't pleased with the outcome, it was easy to modify the mission;
3. Learning to use *MissionLab* was easy;
4. The *MissionLab* interface was easy to follow;
5. Overall, I found it easy to use *MissionLab*

The questions designed to assess user Satisfaction were as follows:

6. *Missionlab* is a satisfactory tool for creating robot missions;
7. *Missionlab* is a satisfactory tool to repair faulty missions;
8. I was satisfied with the level of assistance that *MissionLab* provided;
9. The final mission was completed to my satisfaction;
10. Overall, I was pleased with my *MissionLab* experience.

All of the questions, with the exception of questions 5 and 10 (used to assess the overall ease of use and satisfaction), were subdivided into four subparts: one per each of the tests, and overall. Each of the subquestions was presented as a 5-point likert-type scale, with "Strongly Agree" anchored at 1, and "Strongly Disagree" anchored at 5. Figure 7 shows an example of such a question with subquestions:

It was easy to create a robot mission using *MissionLab*.

a) Waypoints Navigation scenario:

Strongly Agree Agree Neutral Disagree Strongly Disagree

b) Biohazard scenario:

Strongly Agree Agree Neutral Disagree Strongly Disagree

c) Hostage Monitoring scenario:

Strongly Agree Agree Neutral Disagree Strongly Disagree

d) Overall:

Strongly Agree Agree Neutral Disagree Strongly Disagree

Figure 7: Example Question of the Post-Questionnaire

3.5 Study Participants

A total of 30 subjects participated in the study. The subjects varied widely according to their demographics:

- age: from 19 to over 50 years old;
- gender: 14 females and 16 males, distributed equally among the two between-subject conditions;
- background: a variety of educational backgrounds, including both technical (31%) and non-technical fields (69%);
- military experience: 21% of the participants had limited military experience;
- occupation: less than half of the participants were undergraduate or graduate students, out of those only three were Computer Science or Computer and Electrical Engineering majors; none of the subjects was a member of our or any other robotics laboratory;
- computer experience: two of the subjects had advanced programming experience, and 10 others – some programming or network administration experience; all others described themselves as users.

Each participant was compensated for his/her participation in the study. The data of one participant from the Repair condition were excluded from the analysis due to the subject's disregarding the administrator's instructions and exhibiting a very low (insufficient) level of computer experience.

3.6 Analysis and Results

The measure of duration, accuracy and the test-specific answers to the Ease-of-Use and User-Satisfaction questions were analyzed by means of Repeated Measures ANOVA; and the overall answers were analyzed by means of One-Way ANOVA tests. 3 out of 90 tests conducted were not completed due to the lack of certain functionality required by the users, and the measures of accuracy and duration for these were treated as missing data.

3.6.1 Hypothesis 1

No significant effects were found for the measure of Duration: the interaction effect of Repair Method and Task yielded F=.641, p<.531; the main effect of Repair Method and Task yielded F=1.144, p<.295, and F=1.56, p<.221, respectively. Although our hypothesis that the presence of the Mission Repair feature will reduce the time required to produce a successful mission was not proven, the presence of this feature did not produce a negative effect on the duration.

3.6.2 Hypothesis 2

For accuracy, there was no significant interaction effect of Task and Repair Method (F= .727, p<.489), nor there was a main effect of Task (F=.961, p<.390). However, a main effect of Repair Method was observed (M_{groupA} = .820, M_{groupB} = .968, F=4.936, p<.036), suggesting that Group B (with the Repair feature) produced more accurate missions than Group A. The standard error/means plot of accuracy by Repair Method and Task is presented in Figure 8.

However, the percentage of missions completed successfully (100% accuracy) was high, and the normality assumption was somewhat violated. To provide additional information, we also calculated the percentage of successfully completed missions for each factor; the results of this Performance measure are presented in Figure 9 graphically. Thus, the percentage of missions completed successfully was consistently higher for Group B than for Group A, suggesting a positive effect of the Mission Repair feature.

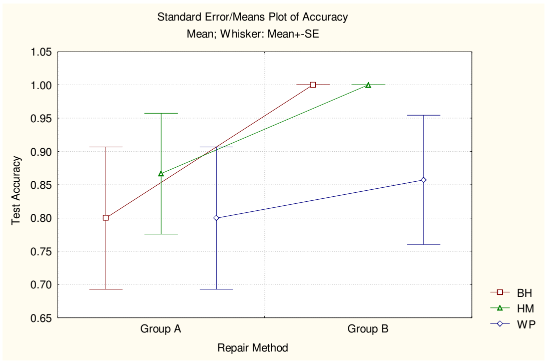
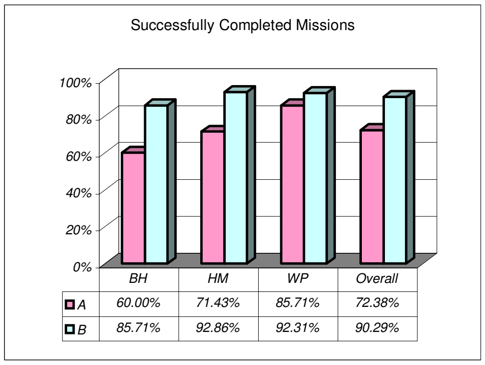


Figure 8: Standard Error/Means Plot of Accuracy for Repair Method and Task. Main Effect of Repair Method was observed, suggesting that Group B produced more accurate missions.



	BH	HM	WP	Overall
A	60.00%	71.43%	85.71%	72.38%
B	85.71%	92.86%	92.31%	90.29%

Figure 9: Percentage of missions completed successfully for each factor

3.6.3 Hypothesis 3

As mentioned earlier, the Ease of Use measure was assessed by means of five questions. The analysis of two of the questions produced statistically significant results. There was a significant interaction effect for the question regarding Ease of Mission Modification (F=4.507, p<.015), suggesting that for some test(s) the effect of the Repair Method was reversed. We analysed the simple main effects of the Repair Method, and found that for Hostage Monitoring scenario the Ease of Mission Modification was higher in Group B than in Group A (M_{groupA} = 2.2, M_{groupB} = 1.4, F=4.505, p<.043; the lower number for the mean indicates greater perceived ease, as "Strongly Agree" was anchored at 1). Figure 10 presents this result visually.

The participants also found that the *MissionLab* with the Mission Repair feature was easier to learn – One-Way ANOVA on the Overall Ease of Learning yielded a statistically significant result (M_{groupA} = 1.67, M_{groupB} = 1.21, F=5.203, p<.031). Figure 11 depicts this result.

Our hypothesis that the Mission Repair Feature will make MissionLab easier to use was partially confirmed: in particular, the Mission Repair Feature made the software easier to learn, and, for one of the modification tasks (functional modification task in Hostage Monitoring Mission), easier to modify the mission.

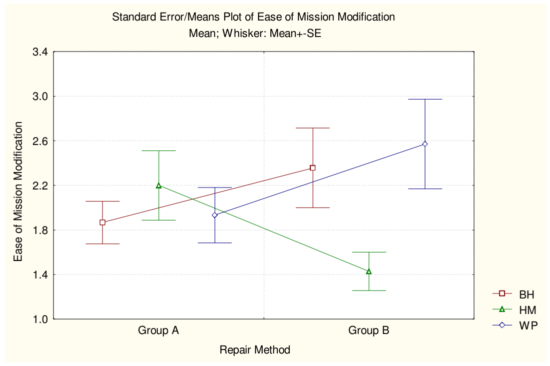


Figure 10: Standard Error/Means Plot of Ease of Mission Modification for Repair Method and Task. Simple Main Effect of Repair Method was observed for Hostage Monitoring Task – it was easier to modify the mission for Group B.

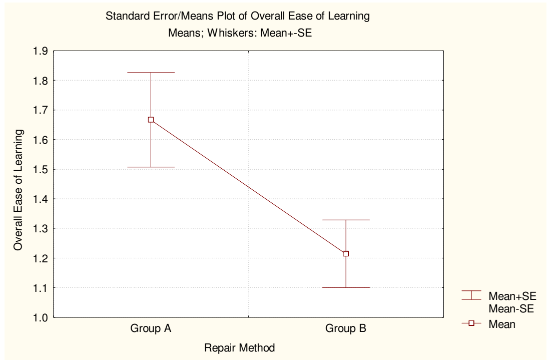


Figure 11: Standard Error/Means Plot of Overall Ease of Learning for Repair Method. The participants in Group B found it easier to learn to use the *MissionLab*.

3.6.4 Hypothesis 4

Our hypothesis that the presence of the Evaluation feature will increase User Satisfaction with the software did not receive confirmation: no statistically significant effects were observed for any of the questions regarding User Satisfaction.

3.7 Discussion

Overall, the addition of the Mission Repair feature produced a number of positive effects, compared to the base CBR Wizard: it increased the percentage of successfully completed missions, as well as the perceived ease of learning of the software and, for certain modification tasks, the perceived ease of mission modification. In addition, we have not observed any harmful effects of this feature, suggesting that further exploring an automatic mission repair mechanism may prove advantageous.

4. CONCLUSION AND FUTURE WORK

In this paper, we presented the latest empirical usability study conducted to assess the usefulness of the newest addition, Mission Repair Feature, to *MissionLab*, robot mission specification system. As *MissionLab* is in constant state of development and improvement, the results of this study give a direction for further improvement of the current system. In particular, some benefits of an automated mission repair mechanism were found: the Mission Repair feature increased the percentage of successfully completed missions thus improving the effectiveness of the overall system, as well as increased the perceived ease of use of the software in certain cases, thus contributing to greater user satisfaction. Unfortunately, these finding were not as strong as we had hoped. One reason for this may be due to the fact that the Mission Repair feature is a prototype, and is designed to handle a limited number of failure types. Therefore, during the experiments, a few participants made errors that the mechanism could not repair efficiently. Given the findings and our observations, it seems reasonable to investigate the mechanism for automated mission repair further by: 1) Increasing the number of the failure types it can handle; and 2) identifying other potentially weak parts in the user interface design by evaluating its individual components.

ACKNOWLEDGMENTS

This research is supported by DARPA/U.S. Army SMDC contract #DASG60-99-C-0081. Approved for Public Release; distribution unlimited. We would like to thank Doug MacKenzie for his continuous support throughout the system development, experiment design, and results interpretation, as well as Zsolt Kira and Ananth Ranganathan for their help in conducting the experiments.

REFERENCES

[1] M. Burke J.L., R.R., Rogers, E., Lumelsky, V.J., Scholtz, J., "Final report for the DARPA/NSF interdisciplinary study on Human–Robot Interaction," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 34, 2004, pp.

[2] J. Scholtz, "Theory and Evaluation of Human Robot Interactions," *36th Hawaii International Conference on System Sciences*, 2003, pp.

[3] M. Burke J.L., R.R., Coovert, M.D., Riddle, D.L., "Moonlight in Miami: A Field Study of Human-Robot Interaction in the Context of an Urban Serach and Rescue Disaster Response Training Exercise," *Human-Computer Interaction*, vol. 19, 2004, pp. 85-116.

[4] H. A. Yanco, Drury,J.L., Scholtz, J., "Beyond Usability Evaluation: Analysis of Human-Robot Interaction at a Major Robotics Competition," *Journal of Human-Computer Interaction*, vol. 19, 2004, pp. 117-149.

[5] J. Draper, "Human Factors in Telemanipulation: Perspective from the Oak Ridge National Laboratory Experience," *Proc. SPIE Telemanipulator Technology and Space Telerobotics*, The Int'l Soc. for Optical Eng., 1993, pp. 162-173.

[6] J. Draper, "Teleoperators for Advanced Manufacturing: Applications and Human Factors Challenges," *Int'l J. Human Factors in Manufacturing*, vol. 5, 1995, pp. 53-85.

[7] J. Draper and L. Blair, "Workload, Flow, and Telepresence during Teleoperation," *Proc. Int'l Conf. Robotics and Automation*, 1996, pp.

[8] "Real Time Cooperative Behavior for Tactical Mobile Robot Teams," Georgia Tech College of Computing and Georgia Tech Research Institute, Final Report A003, 2001.

[9] K. S. Ali, *Multiagent Telerobotics: Matching Systems to Tasks*. Georgia Institute of Technology, Ph.D. Dissertation, 1999.

[10] Y. Endo, D. C. MacKenzie, and R. C. Arkin, "Usability Evaluation of High-Level User Assistance for Robot Mission Specification," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 34, 2004, pp. 168-180.

[11] D. C. MacKenzie, Arkin, R.C., "Evaluating the Usability of Robot Programming Toolsets," *The International Journal of Robotics Research*, vol. 17, 1998, pp. 381-401.

[12] R. C. Arkin, *Behavior-Based Robotics*. MIT Press, Cambridge, Mass., 1998.

[13] D. C. MacKenzie, R. C. Arkin, and J. M. Cameron, "Multiagent Mission Specification and Execution," *Autonomous Robots*, vol. 4, 1997, pp. 29-52.

[14] J. Kolodner and D. Leake, "A Tutorial Introduction to Case-Based Reasoning," in *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, D. Leake, ed., AAAI Press, 1996, pp. 31-65.

[15] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, 1986, pp. 81-106.