A Design Process for Robot Capabilities and Missions Applied to Microautonomous Platforms

Zsolt Kira*[a], Ronald C. Arkin [a], Thomas R. Collins[b]
[a]Mobile Robot Laboratory, Georgia Tech, TSRB S27, 85 Fifth St., Atlanta, GA, USA 30308;
[b]GTRI/ School of Electrical and Computer Engineering, Georgia Tech, Atlanta, GA, USA 30332

ABSTRACT

As part of our research for the ARL MAST CTA (Collaborative Technology Alliance) [1], we present an integrated architecture that facilitates the design of microautonomous robot platforms and missions, starting from initial design conception to actual deployment. The framework consists of four major components: design tools, mission-specification system (*MissionLab*), case-based reasoning system (CBR Expert), and a simulation environment (USARSim). The designer begins by using design tools to generate a space of missions, taking broad mission-specific objectives into account. For example, in a multi-robot reconnaissance task, the parameters varied include the number of robots used, mobility capabilities (e.g. maximum speeds), and sensor capabilities. The design tools are used to intelligently carve out the space of all possible parameter combinations to produce a smaller set of mission configurations. Quantitative assessment of this design space is then performed in simulation to determine which particular configuration would yield an effective team before actual deployment. *MissionLab*, a mission-specification platform, is used to incorporate the input parameters, generate the underlying robot missions, and control the robots in simulation. It also provides logging mechanisms to measure a range of quantitative performance metrics, such as mission completion rates, resource utilization, and time to completion, which are then used to determine the best configuration for a particular mission. These metrics can also provide guidance for the refinement of the entire design process. Finally, a case-based reasoning system allows users to maximize successful deployment of the robots by retrieving proven configurations and determine the robot capabilities necessary for success in a particular mission.

Keywords: Microautonomous Science and Technology, Robotics System Design, Multi-Robot Systems

1. INTRODUCTION

The design of microautonomous robot systems for use in large, multi-robot missions presents many challenges. Unlike their larger counterparts, small robots are extremely limited with respect to many of their capabilities, including power, sensing, mobility, payload, and computational power. As such, severe trade-offs must be made between these characteristics, and often-times adding capabilities in one area results in sacrifices in the other areas. The interaction between these trade-offs is often difficult to model, and design choices made before deployment can produce undesired side-effects.

This paper presents an architecture for exploring the design space of a complex, microautonomous multi-robot mission. Instead of committing to one particular design before implementation and deployment, a large space of designs is iteratively narrowed down by providing the designer with quantitative data that can reveal the effect of design decisions. The purpose of the architecture is to allow designers to easily specify mission objectives and characteristics, quantitatively assess a space of possible designs through simulation, and analyze the resulting data in order to iterate through the design cycle. Furthermore, the architecture relies on reuse of data from each iteration in order to lessen the amount of simulation required in the subsequent design of new missions with similar characteristics.

The main contribution of this paper is to propose that this be achieved by leveraging existing mission specification

*zkira@gatech.edu; phone 1 (404) 894-9311; http://www.cc.gatech.edu/ai/robot-lab/

Report Documentation Page				*Form Approved* *OMB No. 0704-0188*
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.				
1. REPORT DATE **2010**		2. REPORT TYPE		3. DATES COVERED **00-00-2010 to 00-00-2010**
4. TITLE AND SUBTITLE **A Design Process for Robot Capabilities and Missions Applied to Microautonomous Platforms**				5a. CONTRACT NUMBER
				5b. GRANT NUMBER
				5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)				5d. PROJECT NUMBER
				5e. TASK NUMBER
				5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Georgia Institute of Technology,School of Electrical and Computer Engineering,Atlanta,GA,30308**				8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**				
13. SUPPLEMENTARY NOTES				
14. ABSTRACT **see report**				
15. SUBJECT TERMS				
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**12**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

software developed for robotic systems (*MissionLab*) [2], its integration with several robotic simulators, as well as a Case-Based reasoning (CBR) module that has previously enabled the easy creation of robot mission and reuse of prior missions for new situations [3]. The main components of the design cycle rely on: (1) design tools used to intelligently carve out a space of mission configurations that can potentially achieve the desired mission objectives; (2) a mission-specification system and simulation framework used to produce quantitative performance metrics for each possible design; (3) a simulation framework used to simulate each possible design; and (4) a case-based reasoning system that store, potentially generalize, and display the resulting data to the user.

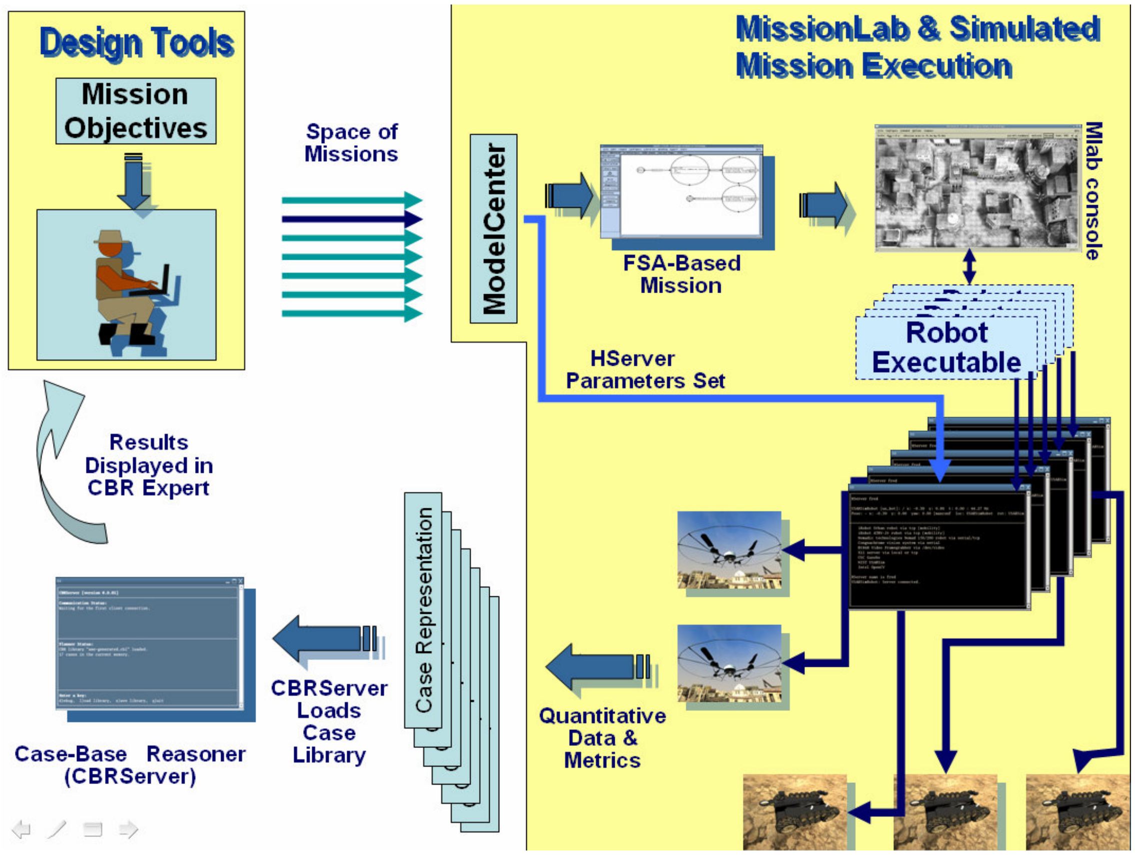

Design Tools
Mission Objectives
Space of Missions
MissionLab & Simulated Mission Execution
ModelCenter
FSA-Based Mission
Mlab console
Robot Executable
HServer Parameters Set
Results Displayed in CBR Expert
Case Representation
CBRServer Loads Case Library
Case-Base Reasoner (CBRServer)
Quantitative Data & Metrics

Figure 1. Architecture depicting the design cycle of a complex, multi-robot mission.

The remainder of the paper is structured as follows. First a review of various design methodologies is covered, with particular emphasis on previous work on system of systems design that inspired this work. Then, an overview of the entire framework is presented, with details of individual components following the overview. We then present a particular instantiation of the framework and show an example reconnaissance mission. We conclude the paper with a review of the contributions and areas of future work.

2. FRAMEWORK OVERVIEW

Figure 1 gives an overview of the system architecture. The designer begins by defining potential capabilities and robot configurations that can be varied (i.e., the design choices). Once these are developed, design tools are used in order to generate a space of possible missions for a microautonomous systems. For example, in a reconnaissance task, the parameters to be varied can include number of robots to be used, mobility capabilities (e.g. maximum speeds, turn angles, etc.), and sensor capabilities. The design tools are used to intelligently carve out the space of all possible combinations of these parameters, and produce a smaller space (consisting of potentially dozens or hundreds) of mission configurations. Exploration of this space is then necessary for several potential uses. For example, such an architecture can be used to assess the direction of research funding, such that the capabilities that are likely feasible and add significant advantage for the mission can be pursued [4]. Assessment of the overall space of missions can also be used to determine which particular feasible configuration of robots and robot capabilities would likely yield better performance and effectiveness before actual deployment.

In order to assess a particular mission configuration, a simulator and mission-specification platform are used. In our case, a realistic three-dimensional simulator with a physics engine is used to run the experiments. The input mission configurations are converted into configuration files that are used by the mission configuration system (*MissionLab*) to implement the simulation parameters. *MissionLab* is then used to: (1) generate the underlying robot missions represented as Finite State Automata (leveraging a large amount of existing robotic behaviors already available); (2) compile the resulting FSA into a robot executable; (3) and then run the mission on a simulation platform and control the robots via the hardware abstraction layer. Using this framework has the added benefit that the same resulting missions can be run on different simulators or a multitude of real robotic platforms via the hardware abstraction layer available in *MissionLab*. The simulation runs result in a range of quantitative performance metrics, such as mission completion, resource utilization (distance traveled, sensor usage, robot failures, etc.), and time to completion. Again, the existing logging mechanisms available in *MissionLab* are leveraged to record these statistics.

These metrics can then be used to determine the best team and robot configuration for a particular mission, the increased effectiveness resulting from a particular added capability (and hence can inform investment into research and development of the capability depending on feasibility), and further refinement of the entire design process. In order to allow the designer to more easily utilize and understand the quantitative data, each configuration (input) and resulting metrics (output) is input into the Case-Based Reasoning system. This existing module residing within *MissionLab* has previously been used to allow non-expert users to easily retrieve and adapt previously-built missions instead of starting from scratch [3]. In this paper, we utilize the same capability to store and allow retrieval of the large amount of quantitative data generated by the simulation runs. Users can then query the best configurations (according to the metrics) given a mission specification, and is described in subsequent sections.

3. DESCRIPTION OF COMPONENTS

We now describe each component in detail, along with an example scenario.

3.1 Example Scenario

Throughout this paper, we will use a biohazard search scenario as an example. Figure 2 shows the environment, one of the simulated robots, and a biohazard. Note that this mission was designed by the Grand Challenge Team in the Aerospace Systems Design Laboratory (ASDL) also at the Georgia Institute of Technology. The environment replicates a city layout consisting of narrow streets. There are chemical objects (shown as a barrel in the left image) that the robots can detect. Furthermore, there are also enemies that the robot should avoid while detecting the biohazards.

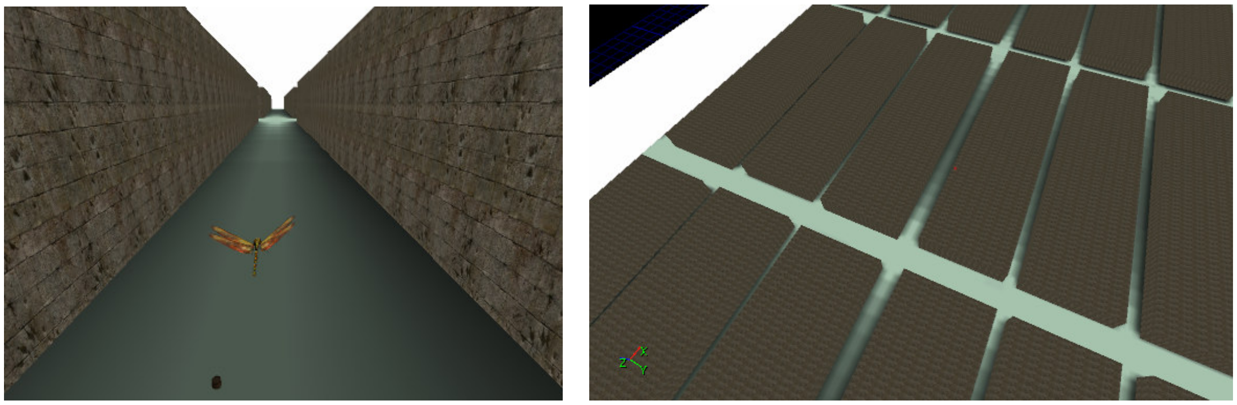


Figure 2. Screenshots of the example scenario, including the microautonomous robot model and barrel representing a chemical agent (left) and larger-scale overhead shot of the environment (right).

3.2 Design Tools

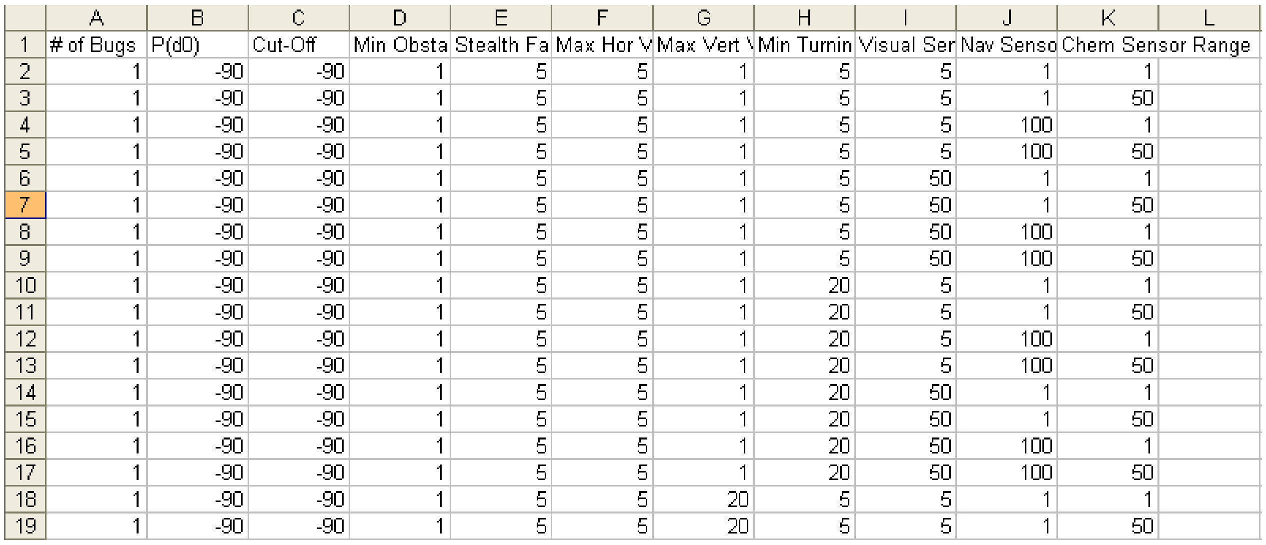
Given a mission consisting of a team of microautonomous systems, many potential designs are possible. The designer must first identify which characteristics can be varied. As mentioned, in a reconnaissance task, the parameters to be varied can include number of robots to be used, mobility capabilities (e.g. maximum speeds, turn angles, etc.), and sensor capabilities. Each of these characteristics has boundaries or constraints that define potential choices. The space of all choices defines the design space. For example, a maximum of ten robots may be available due to cost constraints. Other constraints are imposed by physical reality, for example constraining the maximum velocities of the robots or communication distances between the robots.

Table 1 (left) shows the inputs for the example scenario, obtained from our collaborators at the ASDL. Several aspects of the robot team are addressed, including the number of robots in the team, aspects of communications (e.g. signal strength), aspects of the robot behavior (e.g. obstacle avoidance clearance), and physical constraints on the robots (e.g. maximum velocities).

Table 1 – Left: Inputs for the example scenario, describing the robot capabilities that are applicable to the scenario. Right: Outputs of the simulations in the form of quantitative metrics. (Provided by the ASDL group at Georgia Tech.)

Input Type	**Min**	**Max**	**Units**		**Output Type**
# Robots	1	5			Mission Accomplished
Comm Signal Strength	-90.0	0.0	dBm		Mission Time
Comm Signal Cut-off	-90.0	-30.0	dBm		Number of Robot Failures
Min Obstacle Clearance	1	5	m		Distance Traveled
Stealth Factor	5.0	75.0	%		Sensor Usage
Max Horizontal Velocity	5.0	50.0	m/s		Area Covered
Max Vertical Velocity	1.0	20.0	m/s		Enemies Spotted
Max Turning Radius	5.0	20.0	Radians		Biohazards Found
Visual/IR Max Range	5.0	50.0	m		Spotted by Enemy
Max Navigation Sensor Range	1.0	100.0	m		Amount of Communication
Chemical Sensor Max Range	1.0	50.0	m		% Lone Network

After deciding on which system parameters can be varied and the resulting design space, design tools can be used to intelligently carve out useful portions of the space. Potential methods of exploring such a space include central composite design (CCD) [5], Latin hypercube [6], choosing extreme points in the spaces, or generating random points in the space. The result of this analysis is a sampling of the design space; that is, a reduced space consisting of robot team configurations and their capabilities. In the example scenario, the results of this exploration were a set of 16,000 possible designs, each of which contained particular parameters from the input space (discussed earlier.) Figure 3 shows a small subset of these configurations. These inputs correspond directly to the inputs in Table 1.



	A	B	C	D	E	F	G	H	I	J	K	L
1	# of Bugs	P(d0)	Cut-Off	Min Obsta	Stealth Fa	Max Hor V	Max Vert \	Min Turnin	Visual Ser	Nav Senso	Chem Sensor Range	
2	1	-90	-90	1	5	5	1	5	5	1	1	
3	1	-90	-90	1	5	5	1	5	5	1	50	
4	1	-90	-90	1	5	5	1	5	5	100	1	
5	1	-90	-90	1	5	5	1	5	5	100	50	
6	1	-90	-90	1	5	5	1	5	50	1	1	
7	1	-90	-90	1	5	5	1	5	50	1	50	
8	1	-90	-90	1	5	5	1	5	50	100	1	
9	1	-90	-90	1	5	5	1	5	50	100	50	
10	1	-90	-90	1	5	5	1	20	5	1	1	
11	1	-90	-90	1	5	5	1	20	5	1	50	
12	1	-90	-90	1	5	5	1	20	5	100	1	
13	1	-90	-90	1	5	5	1	20	5	100	50	
14	1	-90	-90	1	5	5	1	20	50	1	1	
15	1	-90	-90	1	5	5	1	20	50	1	50	
16	1	-90	-90	1	5	5	1	20	50	100	1	
17	1	-90	-90	1	5	5	1	20	50	100	50	
18	1	-90	-90	1	5	5	20	5	5	1	1	
19	1	-90	-90	1	5	5	20	5	5	1	50	

Figure 3. This figure shows a subset of the input parameters after the various space-filling designs were applied. Each of these represents a particular robot team configuration, and is subsequently simulated to measure its mission effectiveness.

The points defining the design capabilities in this reduced space are then simulated in order to ascertain their effectiveness in the desired mission. In addition, the desired output metrics must be identified. These can include whether the mission was successful (according to a task-specific criteria), mission time, the amount of area covered by the robot team, etc. Table 1 (right) shows the output metrics designed by the ASDL team for the example scenario.

3.3 Mission Specification Software

Once a particular set of designs is determined, they must be implemented. In this research we use *MissionLab*, a mission specification system that implements the AuRA robot architecture [2]. *MissionLab* provides many of the requirements necessary for the architectural framework within its large tool set developed over 15 years and now in release version 7.0[1]. This includes the specification and deployment of multi-robot missions, integration with both hardware and simulation platforms, logging mechanisms that measure various aspects of mission success, and the Case-Based reasoning module that is described further below.

Figure 4 shows the overall *MissionLab* architecture. In order to specify a robot mission, *MissionLab* uses a finite state automata (FSA) that describes a behavior-based robot mission. Each node corresponds to a high-level behavior (e.g. moving towards a goal), and consists of one or more primitive sub-behaviors. For example, moving towards the goal consists of sub-behaviors for actual movement toward the goal, obstacle avoidance, and optional noise in the movement. Each sub-behavior outputs a vector that is converted into motor velocities, and the vectors from each sub-behavior are summed up to obtain the final robot control.

The specification of robot missions is done through a user interface (CfgEdit). The resulting FSA is compiled down to a robot executable that obtains sensor data and calculates the resulting behavioral outputs. This executable obtains sensor data and directs robot action by communicating with a hardware abstraction layer (HServer). This abstraction

[1] *MissionLab* is freely available for research and education and can be downloaded at: http://www.cc.gatech.edu/ai/robot-lab/research/MissionLab/

layer can communicate with both simulation platforms as well as robot hardware, and hence the same mission can be run on both without modification. Finally, a console user interface (mlab) displays the robot state and sensor information so that the operator can discern mission success. The behavior for multiple robots can be described in this manner, and additional modules address the creation of robot teams [7].

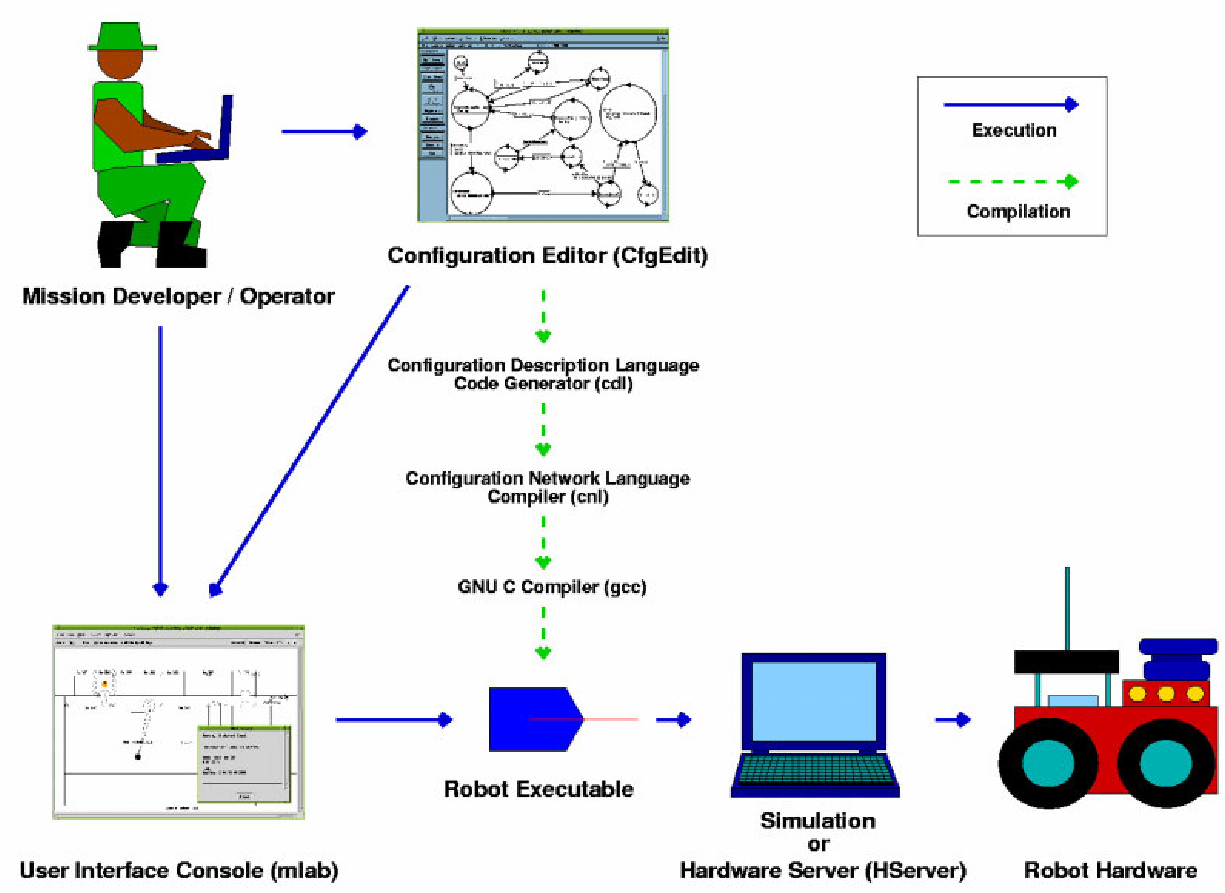

Execution
Compilation
Mission Developer / Operator
Configuration Editor (CfgEdit)
Configuration Description Language Code Generator (cdl)
Configuration Network Language Compiler (cnl)
GNU C Compiler (gcc)
Robot Executable
Simulation or Hardware Server (HServer)
Robot Hardware
User Interface Console (mlab)

Figure 4. The *MissionLab* mission configuration system [2]. *MissionLab* consists of several components. The designer builds missions in the form of finite state automata (FSA) using a graphical user interface (CfgEdit). The resulting FSA is compiled down to a robot executable that obtains sensor data and calculates the resulting behavioral outputs. This executable obtains sensor data and directs robot action by communicating with a hardware abstraction layer (HServer). This abstraction layer can communicate with both simulation platforms as well as robot hardware, and hence the same mission can be run on both without modification. Finally, a console user interface (mlab) displays the robot state and sensor information so that the operator can discern mission success.

Logging of robot data (such as position and velocity information) can be performed both within the robot executable as well as in the console. Quantitative metrics regarding the performance of the robot team are also calculated. As part of the design cycle, the designer must specify what quantitative metrics are to be used in assessing mission success. As stated earlier, examples can include mission completion times, distances traveled, or task-dependent metrics such as percent time on target if a target-tracking task is being performed. These metrics are measured for all potential design configurations identified using design tools above. The end result is a large set of quantitative data that the designed can analyze. This data maps the input parameters (i.e. design decisions) with the output metrics (i.e. the performance metrics). In order to make the utilization of such data easier, we leverage the existing case-cased reasoning (CBR) system in *MissionLab*.

Figure 5 shows the current Finite State Automata for the example mission. The robot wanders around, attempting to cover new ground, while avoiding obstacles and enemies. If it detects a chemical agent, it goes to it (again, while avoiding enemies). Once it is near it, it stays there for a little bit, and then continues wandering. It wanders around ignoring chemical agents, until it reaches the first state again and starts over (this is to avoid moving near the same chemical agent over and over again).

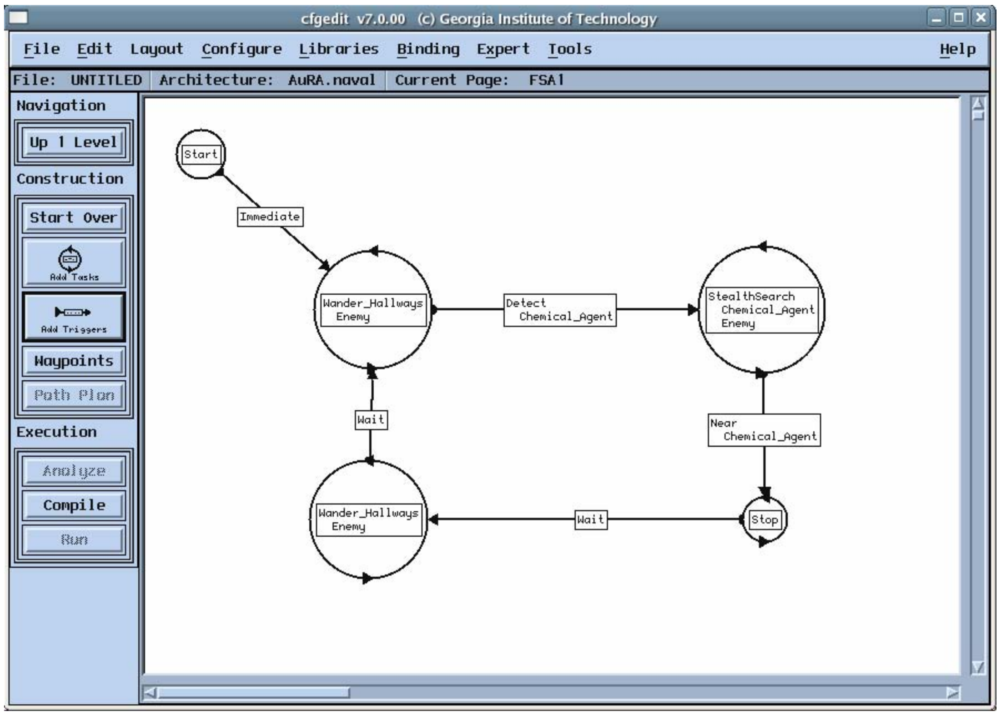


Figure 5. The Finite State Automata (FSA) implementing the robot behaviors for the example mission. The behavior involves wandering the hallways of the environment in a random manner, while avoiding enemies. Upon detecting a chemical agent, the robot moves towards it, again while avoiding enemies. Once it reaches the detected chemical agent, the robot hovers above it for a specified time before moving on.

3.4 Case-Based Reasoning

Case-based reasoning is an instance-based learning technique that leverages knowledge learned in prior situations to solve new tasks [8]. *MissionLab* has previously used this technique in order to allow novice users to retrieve previously built missions to create new robot missions [3]. The resulting system is called the CBR expert. In this research, we use the same module to allow the designer to analyze the large amount of quantitative data obtained during the simulation of missions as well as to utilize it to decide on particular configurations.

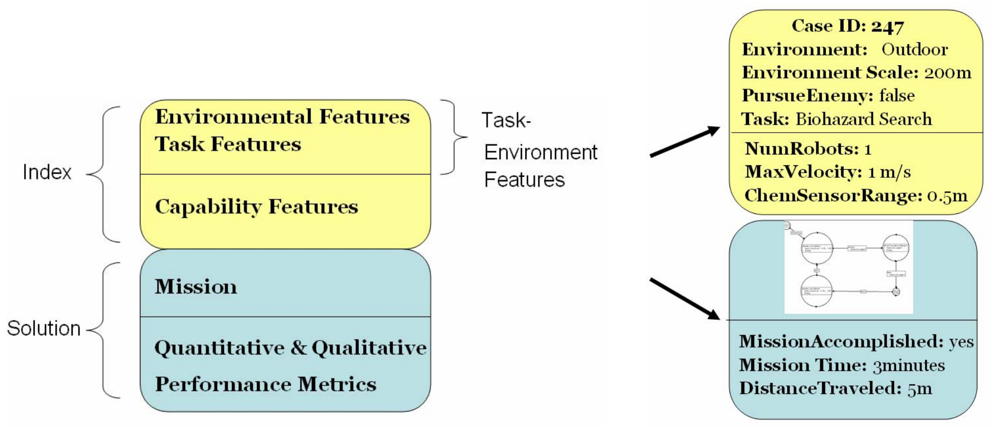
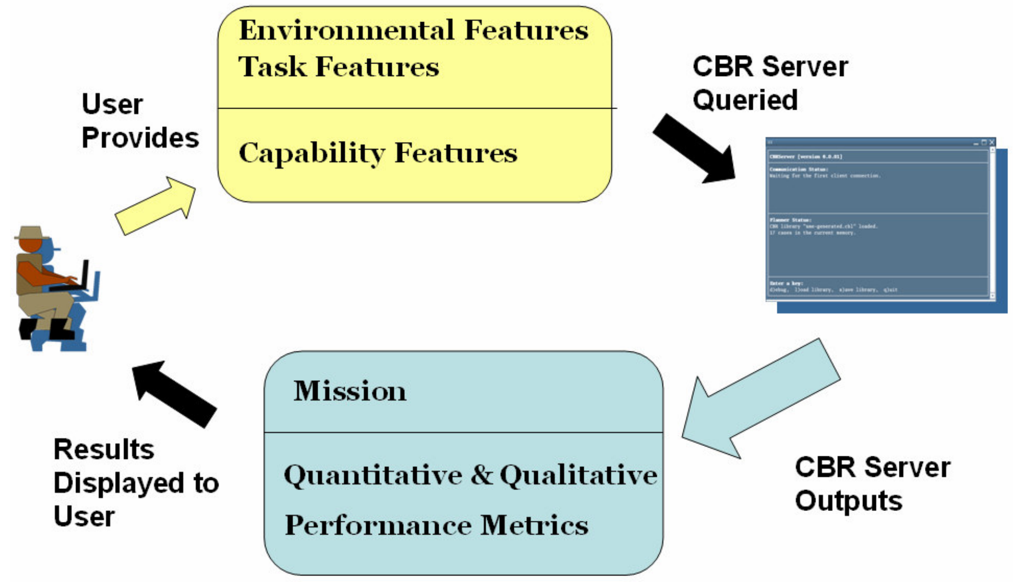


Figure 6. The contents of a case in the CBR system. Environmental features describe features such as whether the task is to be performed indoors or outdoors. Capability features describe the actual capabilities of the robots.

We begin by describing what a case consists of. Figure 6 shows a generic case as well as a particular example. The environmental features consist of features such as whether the task is to be performed outdoors, the scale of the environment, etc. Task features consist of the particular types of tasks that are required (e.g. Biohazard, Recon, etc.). These features are known *a priori* by the designer. Capability features correspond directly to the input parameters in our framework, for example the example shown in Table 1. In other words, these capability features make up the design space. The environmental, task, and capability features make up the index of the case. In other words, cases are retrieved using these features.

The solution portion of the case is the FSA describing the mission. Unlike previous work where the users created the missions and specified their estimates of how well the mission performed in terms of stealth, performance, etc. [3], in this case we obtain these values directly via the output metrics that are automatically generated based on simulation. Once the simulation runs are complete and results in a data set mapping input parameters to output metrics, this data are converted into the case-library representation. This allows other users in the future to query the library and use the knowledge obtained by the simulation runs. Note that the case library can contain a mixture of simulation-based cases as well as previous cases containing only user-created missions.

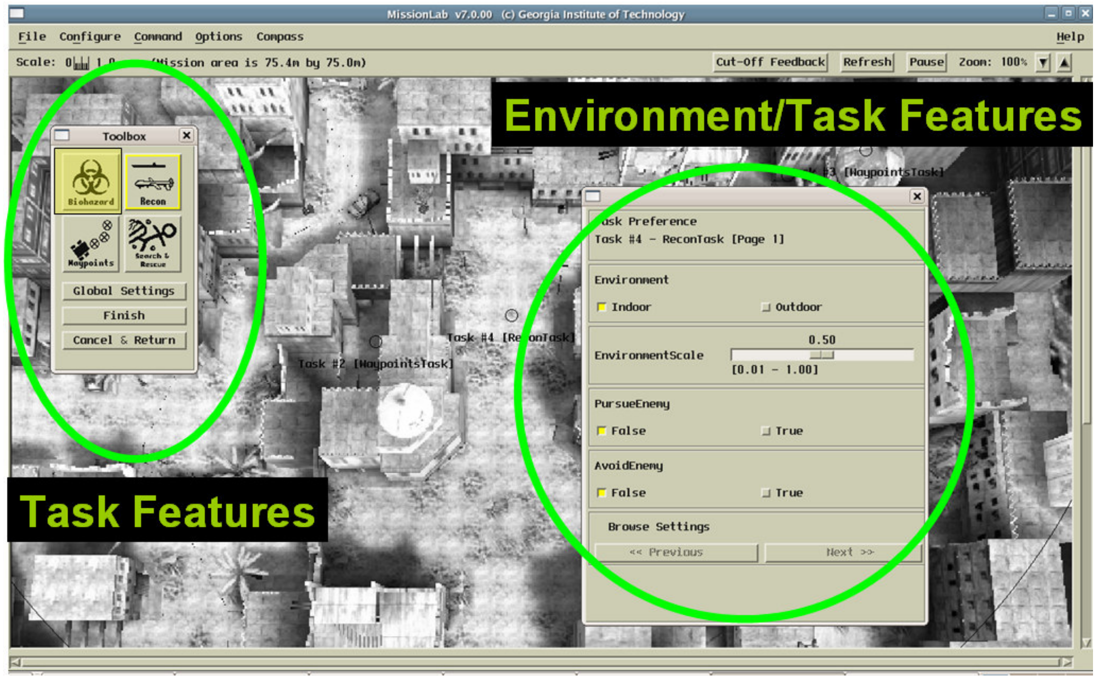


```
CBRServer [version 0.0.01]

Communication Status:
Waiting for the first client connection.

Planner Status:
CBR library "sme-generated.cbl" loaded.
17 cases in the current memory.

Enter a key:
d)ebug,  l)oad library,  s)ave library,  q)uit
```

Figure 7. First query method possible after the case library is populated. The designer specifies the features of the environment, task, and capabilities and obtains the finite state automata (FSA) implementing the mission, along with the estimated performance metrics that would result.

Once the case library is populated with the cases, several methods of query are possible. Figure 7 shows the first method of query possible. In this case, the user specifies the environmental and task features as well as the capability features. In other words the user specifies what the environment will look like, what the task will be, how many robots will be used, what their speed and sensing capabilities will be like, etc. Given this information, the CBR Expert returns the best-suited mission, and if it corresponds to a mission that was already explored by simulation runs, it yields the estimated performance in terms of the output metrics. This provides the user with a powerful estimate that has been obtained by real simulation results as opposed to simply estimates given by the users. Of course, these are still estimates as there may be differences in the particular tasks or environment, but they can be used as a starting point for deciding upon a design. Such queries can also allow the designer to analyze how different design decisions will affect the resulting performance, without having to wait for simulation runs to complete. Finally, other users can query the same library to create new missions for similar tasks, obviating the need to start the design process from scratch.

As part of the example scenario, support for the specific missions, input parameters, and output metrics has been added to the CBR Expert module in *MissionLab*. Furthermore, a module to convert the results of the design space exploration (Section 3.2) to the case representation has also been created. Figure 8 shows the CBR expert for our biohazard mission where the task and environment features can be entered by the user. In this figure, we are performing the Biohazard task (upper left of Task Features). Note that these features correspond to the input parameters determined in Section 3.2. Figure 9 shows the interface for the robot capability features. Note that the features are scaled from 0 to 1, corresponding to the min and max values as specified in Table 1.



ask Preference
Task #4 - ReconTask [Page 1]

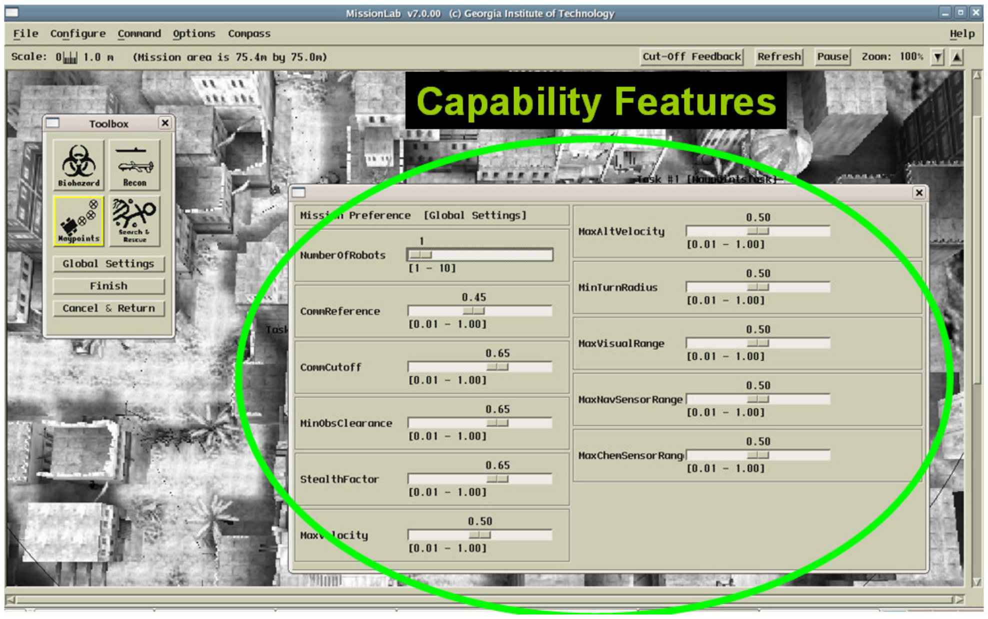
Environment
[✓ Indoor] Outdoor

EnvironmentScale 0.50
[0.01 - 1.00]

PursueEnemy
[✓ False] True

AvoidEnemy
[✓ False] True

Figure 8. Screenshot of the CBR Mission Expert module for the example scenario. This interface is used by the user, after the simulation runs have been loaded into the case library, in order to define the task and environment features. After the user inputs these features, the case-based reasoning module can retrieve the appropriate robot configuration based on the quantitative data.


MissionLab v7.0.00 (c) Georgia Institute of Technology
File Configure Command Options Compass Help
Scale: 0 1.0 m (Mission area is 75.4m by 75.0m)
Cut-Off Feedback Refresh Pause Zoom: 100%
Capability Features
Toolbox
Biohazard
Recon
Waypoints
Search & Rescue
Global Settings
Finish
Cancel & Return
Task #1 [WaypointsTask]

Missi n Preference [Global Settings]

NumberOfRobots	1	[1 - 10]
CommReference	0.45	[0.01 - 1.00]
CommCutoff	0.65	[0.01 - 1.00]
MinObsClearance	0.65	[0.01 - 1.00]
StealthFactor	0.65	[0.01 - 1.00]
Maxv locity	0.50	[0.01 - 1.00]
MaxAltVelocity	0.50	[0.01 - 1.00]
MinTurnRadius	0.50	[0.01 - 1.00]
MaxVisualRange	0.50	[0.01 - 1.00]
MaxNavSensorRange	0.50	[0.01 - 1.00]
MaxChemSensorRang	0.50	[0.01 - 1.00]

Figure 9. Screenshot of the CBR Mission Expert module for the example scenario. This interface is used by the user, after the simulation runs have been loaded into the case library, in order to define the capability features of the robot. After the user inputs these features, the case-based reasoning module can retrieve the appropriate robot configuration based on the quantitative data.

A second potential query method is shown in Figure 10. Here, the designer only specifies the environmental and task features. In other words, the designer only specifies what he or she wants done from a mission perspective. A weighting of the metrics can optionally be specified, representing the needs of the particular mission. Notice that the actual number and types of robots are not specified. The CBR system uses the quantitative simulation data in order to retrieve the configuration of robots (consisting of the number of robots needed, the types of sensing required, etc.) that will achieve the best performance based on prior simulations. In addition, the actual finite state automata (mission plan) and its estimated performance are retrieved. The designer can then iterate through the design cycle and request additional simulations if needed, or use the results of the query directly if they result in acceptable performance. As many designers iterate through the design cycle, the case-based reasoning system can obtain more and more data, resulting in better estimates of what robot capabilities are needed to achieve a particular task, along with estimated performance. Note that the implementation of this query method still remains as future work.

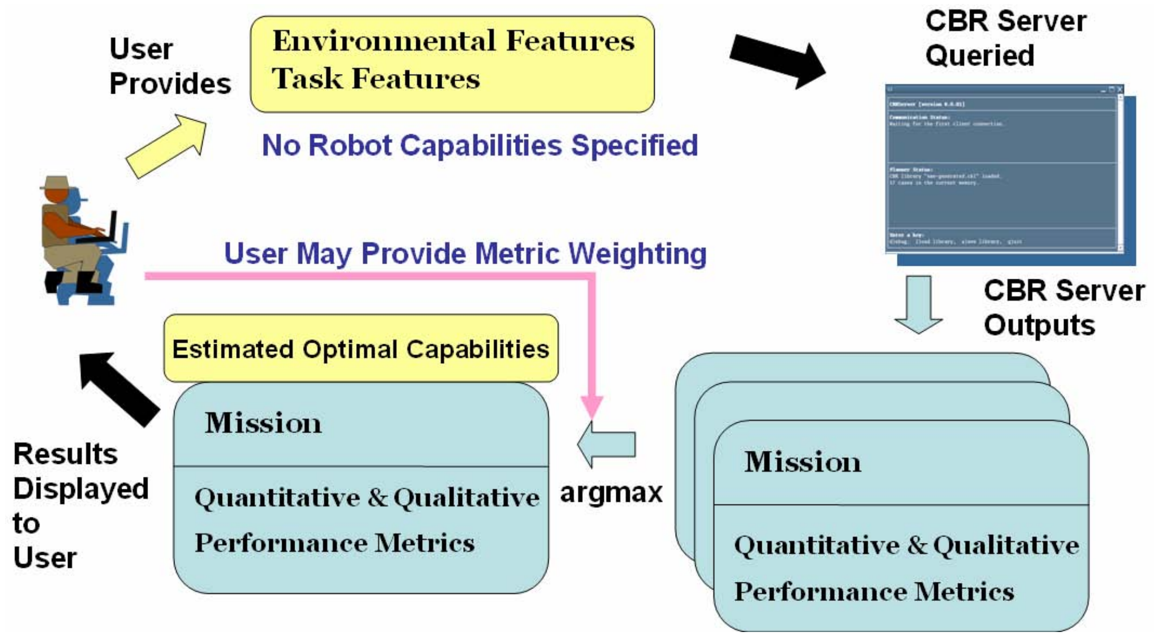

User Provides
Environmental Features
Task Features
No Robot Capabilities Specified
CBR Server Queried
User May Provide Metric Weighting
CBR Server Outputs
Estimated Optimal Capabilities
Mission
Quantitative & Qualitative Performance Metrics
argmax
Mission
Quantitative & Qualitative Performance Metrics
Results Displayed to User

Figure 10. Second query method possible after the case library is populated. Here, the designer only specifies the environmental and tasks features, along with a potential weighting of the performance metric. The CBR system can then retrieve the robot configuration (consisting of the number of robots needed, the types of sensing needed, etc.) that will achieve the best performance based on prior simulations.

3.5 Simulation Platform

Although *MissionLab* already has built-in two-dimensional and three-dimensional simulation capabilities, it can also interface with other simulation platforms (as well as real robot hardware). This support includes two popular simulation platforms: 1) The Gazebo open-source three dimensional simulation environment [9] and 2) USARSim, a three dimensional simulation platform based on Unreal Tournament 2004, featuring realistic lighting, texturing, and physics engine [10]. For the example scenario, USARSim was used. Figure 11 shows a screenshot of the scenario within this simulation framework.

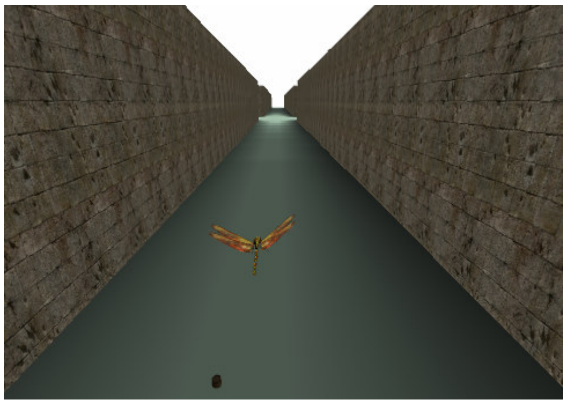


Figure 11. Screenshot of the example scenario implemented within the USARSim simulation environment.

4. CONCLUSIONS

In this paper, we have presented an entire integrated architecture that facilitates the design cycle of complex, multi-robot missions. The architecture is specifically intended to allow the exploration of a large design space through simulation. The framework consists of four major components: design tools, mission-specification system (*MissionLab*), case-based reasoning system (CBR Expert), and a simulation environment (USARSim). By leveraging the various strengths and existing modules of the *MissionLab* mission specification software, the simulations can be performed via any of the available simulation platforms (e.g. USARSim). Furthermore, logging mechanisms are used to derive quantitative metrics to assess the effectiveness of the robot configuration at solving the specified task. The case-based reasoning module is used to allow the designer to easily retrieve and analyze the resulting data in order to make design decisions or to iterate on the design cycle. Other users can also access the same case library in order to allow for the reuse of data from prior design space explorations. Finally, the same underlying missions that are retrieved can be used to task real robots without modification, due to the hardware abstraction layer available in *MissionLab*.

5. ACKNOWLEDGMENTS

This work was performed in collaboration with the Grand Challenge Team in the Aerospace Systems Design Laboratory at Georgia Tech. We would like to thank Michael R. Ellis, Steve Isley, David Leuschke, and Christopher A. Wyatt as part of the collaboration for introducing us to the system of systems design methodologies, providing the Design of Experiments inputs, and simulation development. The work in this paper was funded under the U.S. Army Research Laboratory Micro Autonomous Systems and Technology Collaborative Technology Alliance project (BAE # 329420.)

REFERENCES

[1] Beekman, D.W.; Mait, J.N.; Doligalski, T.L., “Micro Autonomous Systems and Technology at the Army Research Laboratory, In the proceedings of the 2008 Aerospace and Electronics Conference (NAECON), p. 159 – 162, July 2008.

[2] MacKenzie, D., Arkin R., and Cameron, J., “Multiagent mission specification and execution,” Autonomous Robots, vol. 4, no. 1, pp. 29–52, 1997.

[3] Endo, Y., Mackenzie, D.C., and Arkin, R.C., “Usability evaluation of high-level user assistance for robot mission specification”, *IEEE transactions on systems, man and cybernetics*, part C, applications and review, 2004.

[4] Kirby, M.R. and Mavris, D.N., “An approach for the intelligent assessment of future technology portfolios”, in *40th AIAA Aerospace Sciences Meeting*, Reno, Nevada, 2002.

[5] Myers, Raymond H. *Response Surface Methodology*. Boston: Allyn and Bacon, Inc., 1971.

[6] Iman, R.L., Helton, J.C., and Campbell, J.E., "An approach to sensitivity analysis of computer models, Part 1. Introduction, input variable selection and preliminary variable assessment". Journal of Quality Technology 13 (3): 174–183, 1981.

[7] Hsieh, M.A. and Cowley, A. and Keller, J.F. and Chaimowicz, L. and Grocholsky, B. and Kumar, V. and Taylor, C.J. and Endo, Y. and Arkin, R.C. and Jung, B., "Adaptive teams of autonomous aerial and ground robots for situational awareness", Journal of Robotic Systems, Vol. 24(11-2), pp.991-1014, 2007.

[8] Kolodner, J., "Case Based Reasoning", Morgan Kaufmann Publishers, San Mateo, 1993.

[9] Brian Gerkey, Richard T. Vaughan and Andrew Howard. "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems". In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003)*, pages 317-323, Coimbra, Portugal, June 2003.

[10] Balakirsky, S.; Scrapper, C.; Carpin, S. & Lewis, M. (2006),UsarSim: providing a framework for multirobot performance evaluation, *in* 'Proceedings of PerMIS'.