Integrated Mission Specification and Task Allocation for Robot Teams - Design and Implementation

Patrick Ulam, Yochiro Endo, Alan Wagner, Ronald Arkin
College of Computing
Georgia Institute of Technology
Atlanta, USA
Email: {pulam, endo, alan.wagner, arkin}@cc.gatech.edu

***Abstract*— As the capabilities, range of missions, and the size of robot teams increase, the ability for a human operator to account for all the factors in these complex scenarios can become exceedingly difficult. Our previous research has studied the use of case-based reasoning (CBR) tools to assist a user in the generation of multi-robot missions. These tools, however, typically assume that the robots available for the mission are of the same type (i.e., homogeneous). We loosen this assumption through the integration of contract-net protocol (CNP) based task allocation coupled with a CBR-based mission specification wizard. Two alternative designs are explored for combining case-based mission specification and CNP-based team allocation as well as the tradeoffs that result from the selection of one of these approaches over the other.**

I. Introduction

Two challenges in fielding teams of mobile robots lie in determining the steps a robot should follow when executing a task (mission specification) and determining which robot should execute a given task (task allocation). Mission specification can be a time-consuming and complex process for users experienced in mission design, let alone for those not intimately familiar with the domain. In the case of multi-robot missions, the difficulty of mission specification is compounded as the process must be repeated multiple times, increasing the possibility for error in the design and increasing design time. For heterogeneous teams (i.e., robots that have significantly differing capabilities such as different sensor packages and/or different terrain capabilities such as aerial versus ground versus undersea unmanned vehicles), allocating the available robots to the appropriate tasks places an additional burden upon the operator. Task allocation becomes increasingly difficult as the number of robots increases or if the capabilities of the robots are not known accurately by the operator.

A. Mission Specification

Mission specification, as described in this work, is the process in which step-by-step instructions are generated to guide one or more robots to accomplish a set of tasks. An example of a mission generated for a multi-robot team may be reconnaissance of an unknown area. Such a mission is composed of many separate tasks. In this case, tasks may include patrolling a particular area or tracking targets discovered in the area. These tasks can be further broken down into the individual actions or behaviors that must be undertaken to achieve them. In order to fully specify a mission, therefore, one must detail: (1) the tasks to undertake; (2) the way to perform the tasks; and (3) any temporal constraints that may exist between the tasks or behaviors (i.e., the requirement of finding a target before tracking it).

One method in which mission specification has been conducted in the past has been through traditional programming, where an expert explicitly programs the robots to perform the tasks in the proper order using languages such as C or LISP. Many systems attempt to automate the mission specification process through the use of planners [1][2]. These approaches, however, often result in mission plans that are difficult for a human operator to customize, reuse, or inspect. An alternative to these approaches is to present the user with a graphical mission specification interface with which reusable components at the action, task, or even mission level can be combined or modified to create the desired mission. MissionLab [3] is one such toolset used for multi-robot mission specification, while similar tools exist in other domains (e.g., [4]).

B. Multi-robot Task Allocation

Once a mission has been decomposed into its requisite tasks, the question of which robot should be responsible for executing each particular task still remains. Many techniques for multi-robot task allocation (MRTA) have been examined. Prior work includes approaches utilizing teammate modeling [5], distributed constraint matching [1], and others [6]. Gerkey and Matarić [7] provide a thorough review of several MRTA frameworks.

MRTA is often framed as an optimization problem in which some performance metric is minimized or maximized given a set of tasks and a set of available robots. In the most general case, generating the optimal mapping is NP-Hard [8]. For many applications, however, the ordering of tasks, the

availability of robots, and the suitability of a robot for a particular task are not known. Because of this, recent research has investigated various market-based approaches to team allocation with significant success [9][10].

In market-based task allocation, a series of task proposals corresponding to the tasks available to be allocated are generated and submitted to all available robots. Each proposal contains information pertaining to the type of task to be executed. The available team members accept or reject each proposal. If the proposal is accepted by a robot, it also submits a bid for that proposal. This bid provides a self-estimate of how suitable that robot is to perform the task described in the proposal. After a specified period of time, the task proposer evaluates all the submitted bids, and awards one or more of the bidders the contract. Once the contract has been awarded, the winning robots are now responsible for that task. This process is continued until all tasks have been contracted out.

Market-based MRTA provides several advantages over other mechanisms for task allocation. For instance, market-based MRTA does not rely on a priori knowledge of tasks, task ordering, or robot availability to allocate robots. Because of this, it is highly robust to unanticipated failures and uncertainty in the environment. While the market-based MRTA does not guarantee optimal allocation of robots to tasks when task ordering is known a priori, it does provide provable bounds on this optimal allocation. In addition, some market-based task allocation techniques are guaranteed to produce optimal allocations when task ordering information and robot reassignment is not available [7].

C. CBR Mission Specification and CNP Task Allocation

This work addresses the complexity of these two tasks by examining multiple ways by which mission specification in the form of a case-based reasoning (CBR) planner can be integrated with contract net protocol (CNP) based task allocation. We examine two alternative designs in which to combine these two tools within the MissionLab mission specification toolkit [11]. In the first design discussed, mission specification and task allocation are linked together during the process of mission generation. In the second design, mission specification instead assists creating structures that support task allocation while the mission is executing. Both approaches afford different strengths and weaknesses in terms of usability, design, and scalability.

II. Mission Specification Using MissionLab

MissionLab is a robotic software toolset that allows a user to compose and execute a multi-robot mission through its graphical user interface [3]. In MissionLab, a mission is described with a graphical representation called an FSA (finite state acceptor) [12]. In the FSA representation, actions (behaviors) are denoted with circles while perceptual triggers (conditions for executing the next action) are denoted with arrows (Figure 1). MissionLab is designed in such a way that the user specifies a mission FSA by assembling behaviors and triggers according to the mission's requirements. The benefits

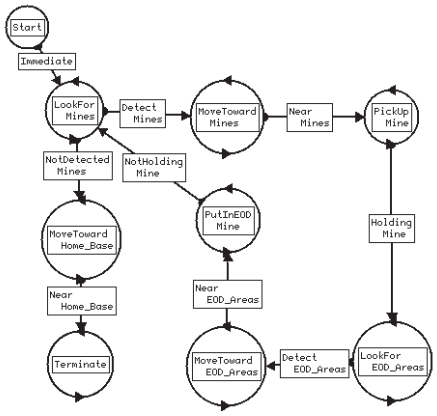

Fig. 1. EOD Mission represented via FSA.

of specifying a robot mission using FSAs over conventional programming methods have been reported in [13]. Once the FSA-based mission is composed, MissionLab translates it into C++ code and compiles this code into robot executable files.

A. The CBR Wizard

The CBR Wizard is a software mechanism integrated into MissionLab to support high-level user assistance during mission specification [14] as well as to guide users in the the process of repairing faulty missions [15]. The CBR Wizard allows the user to specify the mission by retrieving a previous mission from a case library (Figure 2) instead of directly composing a mission with FSAs. Once the user has chosen the various constraints of the mission, the CBR Wizard passes this information to the CBR planner for mission specification.

Cases are designed and tested by experts using the graphical editing tool in MissionLab (CfgEdit). When the expert has generated an exemplar mission, the FSA-based mission code and indexes for the mission are sent to CBRServer, where it is saved in the case library.

The process of retrieving a case from the case library is accomplished by case-based reasoning [16]. More specifically, in order to retrieve a mission, the user specifies constraints and preferences for the desired mission through a map-based interface. These constraints and preferences are used as indexes to collect appropriate cases previously stored by the expert (Figure 3). The cases retrieved from the case library may not exactly match the criteria specified by the user. To handle discrepancies between the retrieved task and the constraints input by the user, the CBR Wizard may adapt the retrieved cases to better match the desired mission. For example, if the previous case only dealt with a mission involving a single robot while the current scenario involves two robots, the solution can be duplicated (i.e. be accommodated to the two-robot mission).

A CBR planner is used for mission specification within this work in lieu of other alternatives for three reasons. By using a CBR planner to aid in mission specification, the user is able to capitalize on libraries of pre-generated, correctly

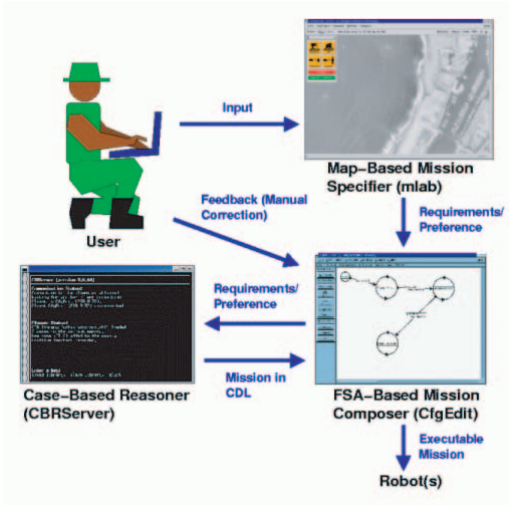

Fig. 2. Overview of mission specification using the CBR Wizard. A user specifies the desired tasks using the map-based interface of MissionLab. CBRServer retrieves a mission based on the user's input. The mission is presented for user inspection, modification, and execution.

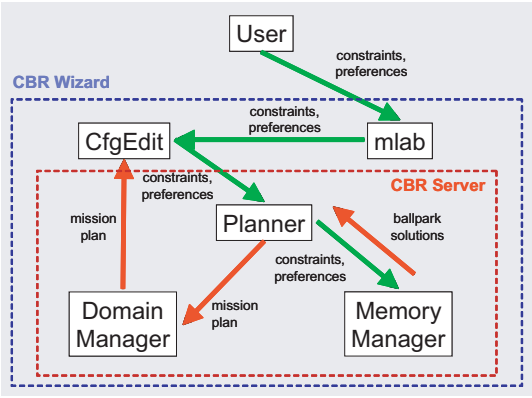

Fig. 3. Case retrieval process: After specifying the constraints and preferences of the tasks for the mission, they are used to retrieve relevant cases from the case library (memory manager). These retrieved cases are adapted and presented to the user for inspection or modification.

specified, and successful missions to serve as the basis for future missions. In addition, by reusing portions of successful mission plans, the cognitive and computational costs of using a CBR mission specification system can be significantly lower than when generating the entire mission by hand or through other planning methods. Finally, faulty domain knowledge by a novice user can be less of a problem when using missions pre-generated by human experts.

III. Contract-Net Protocol Based Task Allocation

The contract-net protocol (CNP) [17] is a distributed negotiation algorithm based on the contract metaphor and often serves as the core negotiation mechanism of market-based MRTA approaches [18][19]. The implementation of the CNP protocol used for MRTA within this work is based upon the Foundation for Intelligent Physical Agents' specifcation [20].

In CNP negotiation, calls for proposals are sent to available resources. These resources in turn accept or reject the proposal. If the proposal is accepted, the resources submit bids indicating their ability to serve the request. In the context of multi-robot task allocation, CNP is utilized to generate mappings between the offers, in this case the tasks required to accomplish the mission, and the available robotic resources. More specifically, the call for proposals takes the form of a description of a task that needs to be executed as well as any information pertinent to the execution of that task. The robots (bidders) who listen to the request can reply with bids indicating their perceived suitability for completing the task described in the request. The highest bidder(s) are then assigned to execute the task by the initiator. Figure 4 provides a high-level overview of the CNP negotiation process.

For example, suppose that the objective of a mission is to monitor the activity of an enemy vehicle in close proximity. First, the initiator generates a call for proposals based on the description of this tracking task. The description may include the information about the enemy vehicle such as its vehicle type, position, or velocity, etc. The robots participating in this auction use this task description to decide if they accept or reject the proposal. If the robots accept, they use the task description as well as knowledge of their own state to calculate a bid for the task. A robot may refuse the proposal if it finds the task infeasible (e.g., an unmanned underwater vehicle cannot track a ground vehicle). When the auction period is expired, the initiator awards the task (contract) to the robot(s) that submitted the highest bid. The winning robot(s) then executes the tracking task until either: 1) the task is successfully completed; 2) the robot determines that completion of the task is no longer possible; or 3) either the initiator or bidder requests a reallocation of the task. In the second and third case, the executing robot will renege on the task (cancel its contract) and the initiator then attempts to reallocate the task among the remaining robots.

Contract reneging takes place any time a robot is unable to or finds it infeasible to continue the task. Such an event could be the result of a mechanical failure or the discovery of new robotic assets in the mission area. When a robot reneges its contract, it informs the task initiator that it is canceling its contract and provides a reason in the form of the list of constraints required to continue the task. The initiator then re-injects the reneged task with the provided constraints for all the remaining robots. The auction process then procedes as normal.

While this discussion has largely assumed perfect communication between proposal initiators and bidders, CNP based approaches can also handle situations where perfect communication is not present. In these cases, negotiations are naturally limited to those within communication range of the initiator and disruptions during the allocation process are handled though a variety of methods such as message timeouts.

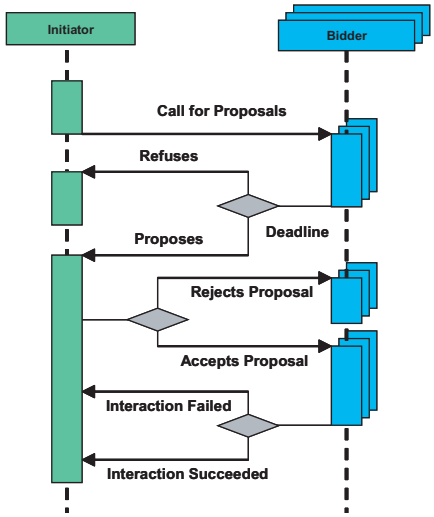

Fig. 4. Interaction diagram for task auctions and bidder responses.

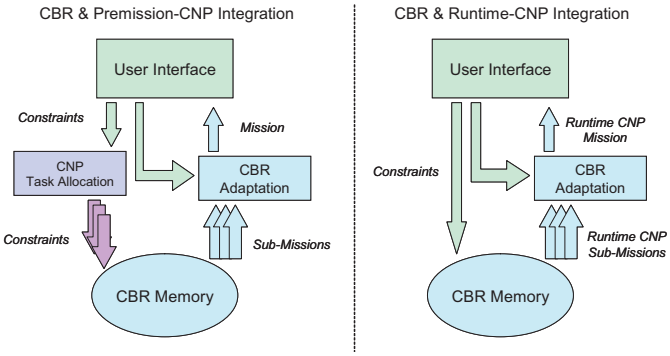

Fig. 5. Alternative integration designs: (L) CBR and pre-mission CNP and (R) CBR and runtime CNP

IV. Integrating CBR and CNP

This section discusses two alternative designs for integrating CBR-based mission specification with CNP-based task allocation (Figure 5). In the first design, CNP provides a robot-to-task mapping during the CBR Wizard's case retrieval process. In the second design, the tasks are assigned by CNP during the execution of the mission. In this design, the CBR Wizard retrieves mission plans that support this runtime task allocation. The design details of both architectures are discussed below.

A. CBR and Premission-CNP Integration

In the CBR and pre-mission CNP architecture, both CBR and CNP are used during the mission specification process (Figure 6). Using this architecture, the mission specification process proceeds along the following steps:

1) The user specifies global mission parameters.
2) The user selects the tasks that compose the mission and the task constraints.
3) The data concerning the available robots and desired tasks are sent to the CBR planner.
4) The tasks are allocated to the robots via CNP within the CBR planner.
5) The task specifications are retrieved from the CBR planner.
6) The retrieved task specifications are adapted and assigned to the proper robot.
7) The resulting allocations and mission plan are presented to the user for inspection, modification, and execution.

The specification of global mission parameters and tasks takes place using the map-based interface of MissionLab shown in figure 7. The global parameters adjust specific variables that affect all robots. An example of such a global variable is MaxVelocity shown in the interface in figure 7. This particular variable scales the maximum speed with which the team members execute their mission plans.

Once the global parameters have been selected, the user selects the component tasks for the mission using the map-based interface. The tasks used in this work include: mine removal, target interception, target tracking, target observation, etc. After selecting a task the user then places the task icon at the approximate location on the map where they believe the task will need to be performed. A dialog is then presented to the user with which they may specify any constraints that may be necessary for the execution of that task. For example, the user may want a tracking task to be performed in a stealthy manner. Such constraints for each task are selected in the task preference dialog. This process is repeated for each task that will compose the mission.

Once the user specifies all the tasks and constraints using the interface, this information (e.g. Figure 8) is sent to the CBR planner (CBRServer) along with data concerning the robots that are available for the mission. The CBR planner uses this data to initiate a series of task allocation auctions using CNP. For each task input by the user, it initiates one auction. The call for proposals details the task that must be performed as well as the constraints the user provided about the task (location, stealthiness, mobility preferences, etc). Each of the available robots responds with a message accepting or rejecting this proposal. If the proposal is accepted, the robots also sumbit a bid b indicating their self-evaluated fitness for the task, where $0 < b \leq 1$. After a short period of time has passed, the task is assigned to the robot that offered the highest bid. If no robot offers a bid greater than 0, then task allocation fails. This process continues until failure or a one-to-one mapping between robots and tasks has been established. If failure occurs, the user is alerted and given the option to either modify the mission or make more robots available for the mission.

The bid calculation process is performed largely via constraint matching (similar to that used by Le Pape [1] to generate mappings between robots and generated plans). This process compares the specified task constraints with the robot's hardware constraints to determine its suitability for the task. The use of CNP in the pre-mission system, however, allows for assignment decisions to be made beyond that of binary constraint matching. By including heuristics to evaluate bids

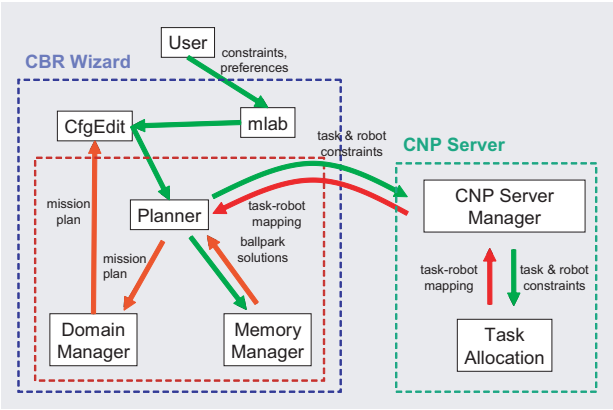

Fig. 6. CBR and Premission-CNP integration

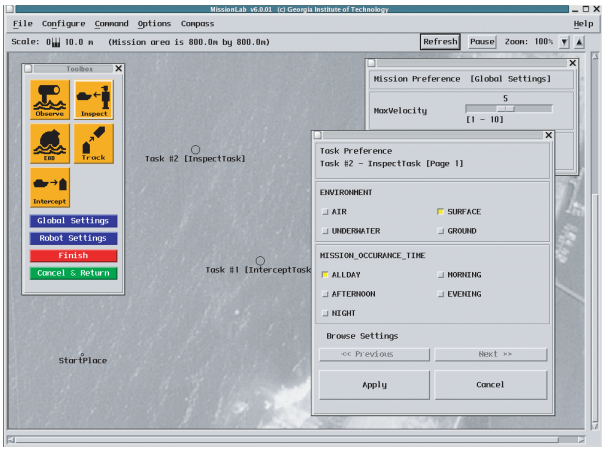

Fig. 7. The user interface for the CBR + pre-mission CNP design: Tasks available for placement on the left. Adjustable global parameters appear in the upper-right window. Individual task preferences are displayed and adjusted by clicking on the task location.

based on additional factors such as proximity to the task or fuel consumption, the mappings have the potential to increase mission performace through more accurate allocation.

After all the tasks have been assigned to the appropriate robots, the CBR planner uses this mapping to assemble and adapt the task specification for each robot as necessary. To retrieve the appropriate task specification, the CBR planner searches its case library using the task descriptions input by the user. The case library is stored as a decision tree, with each branch in the tree representing the various constraints that make up the stored cases (Figure 9). At the leaf nodes of this decision tree, the individual cases generated previously by the expert are stored. The CBR planner retrieves the mission plan that most closely resembles the user's request from the case library (based on locality within the decision tree) and performs adaptation on it. In this case, adaptation consists of modifying the deployment waypoint for the task to match the user's input. The adapted task specification is then matched with the robot that won the contract to execute that task. This process repeats for each task within the mission.

```
overlay shore.ovl
cnp CNP_MODE_PREMISSION

global-feature 0 41.00 1 3 "NumberOfRobots"
option 0 "N/A"
global-feature 1 5.00 1 2 "MaxVelocity"
option 0 "N/A"
global-feature 2 0.25 1 1 "Aggressiveness"
option 0 "N/A"

task 2 "InterceptTask"
coordinate 410.88 254.08
feature 0 0.00 4 0 "ENVIRONMENT"
option 0 "AIR"
option 1 "SURFACE"
option 2 "UNDERWATER"
option 3 "GROUND"
feature 1 1.00 2 0 "MISSION_STEALTHINESS"
option 0 "STEALTHY"
option 1 "NOT_STEALTHY"

task-constraints 2 2 "InterceptTask"
task-constraint 0 0.00 4 0 "ENVIRONMENT"
option 0 "AIR"
option 1 "SURFACE"
option 2 "UNDERWATER"
option 3 "GROUND"
task-constraint 1 1.00 2 0 "MISSION_-
STEALTHINESS"
option 0 "STEALTHY"
option 1 "NOT_STEALTHY"
```

Fig. 8. Example of task constraints specified by the user and sent to the CBRServer.

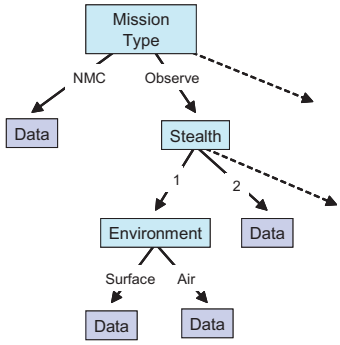

Fig. 9. Decision tree used to store and index cases based on specified constraints.

Once all task specifications have been retrieved, adapted, and assigned to the appropriate robot, the resulting mission plan is visually presented to the user. This mission plan is in the form of the FSA-based specification used within MissionLab. This final mission plan will contain a one-to-one mapping between robots and FSA-based task specifications. This final mission plan may then be inspected, possibly modified, or the execution rehearsed within the graphical editing environment and mlab console prior to downloading to the actual robots.

B. CBR and Runtime-CNP Integration

For the design discussed in the previous section, the task allocation process occurs prior to mission execution (pre-mission phase). In the second CBR-CNP design, task allocation is delayed until mission execution. The CBR planner's role in this architecture is to produce mission plans that support this runtime allocation of tasks.

Mission specification within the CBR and runtime CNP system follows a series of six steps:

1) The user specifies global mission parameters.
2) The user indicates which robots will be available to be tasked during mission execution along with their deployment points.

3) The global parameters and robot selections are sent to the CBR planner.
4) The robots' mission specifications are retrieved from the CBR planner.
5) The retrieved mission specifications are adapted to form the complete mission specification for each robot.
6) The resulting allocations and mission plan is presented to the user for inspection, modification, and execution.

As in the case of the CBR and pre-mission CNP design, the user interacts with the mission specification system via a map-based interface (Figure 10). The user adjusts global mission parameters such as team aggressiveness in a manner similar to the pre-mission system. In addition, however, the user also selects the type of mission the runtime system will be performing. Examples of mission types supported by the system include naval mine countermeasure missions and vessel interception missions.

After the global parameters have been selected the user then selects deployment points for the robots that will be used within this mission. Once a deployment point is selected, the user chooses which of the available robots they wish to be deployed at that point. This process is repeated until all robots that will participate in the mission have had their deployment locations specified.

Once this is done, the data concerning the global parameters and selected robots are then sent to the CBR planner for mission retrieval. Unlike the CBR and pre-mission CNP system, however, the runtime system does not perform task allocation upon case retrieval. It instead retrieves a mission plan for each robot that implements runtime CNP-based task allocation such as shown in figure 11. The mission plans retrieved from the CBR planner's case memory are adapted via the augmentation with the selected deployment waypoints. The resulting FSA mission plans for each robot that will participate in task execution are identical (except for the deployment behavior) to each other regardless of the mission type. This FSA configuration provides the behaviors necessary to participate in the CNP task allocation process and to execute any tasks assigned to the robot. The CNP_ExecuteWonTask behavior within this FSA serves as a behavioral assemblage which implements the execution of any of the possible tasks that the robot can undertake.

In addition, each robot participating in the mission has a series of behaviors that monitor its progress during task execution. If the robot detects that it is unable to continue the mission or significant changes such as a hardware malfunction occur, the robot reneges its contract for the task. This results in the task allocation process reoccurring. Such a behavior acts to ensure that the appropriate robot performs the task even in highly dynamic environments.

In addition to these FSA, the CBR planner retrieves one final mission plan based on the mission-type that the user selected earlier. This mission plan is for a notional command and control vehicle and is responsible for monitoring the progress of the overall mission. It does this through the generation of a call for proposals when the appropriate conditions are met

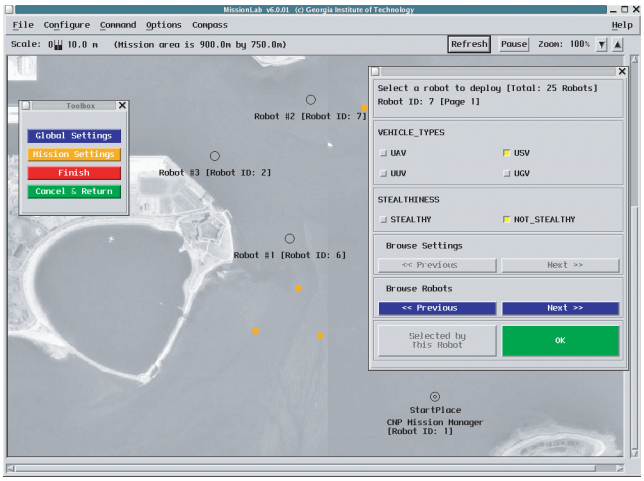

Fig. 10. The user interface for the CBR and runtime CNP integration design: Robot selection dialog is shown on the right side of the screen.

during mission execution (e.g. an enemy vessel is located). The lead robot generates the call for proposals outlining the new task to be accomplished (e.g. tracking the enemy vessel) and then uses this call for proposals to initiate the CNP negotiation process with the other robots deployed in the mission area. An example mission plan for the command and control vehicle participating in a naval mine countermeasure mission can be seen in figure 12

The key interactions between the command and control robot and the other deployed robots can be seen in figure 13. In this figure, the lead robot is labeled as robot 0 and its main responsibility lies in looking for additional tasks to inject into the system and monitoring the process of executing tasks. The deployed robots' (1...n) responsibility, on the otherhand, lies in waiting for tasks to become available, bidding on these tasks, executing tasks, and reporting progress to the command and control vehicle.

After these mission plans have been retrieved and adapted, they are presented to the user for inspection, possible modification, and execution. It is during mission execution, however, that task allocation takes place in the CBR and runtime CNP system.

To further illustrate the runtime allocation process a vessel interception mission is described. The goal of this mission is to intercept a naval vessel departing from an arbitrary port location. In this mission, the leader robot is a notional command and control vehicle (UAV) flying at high altitude. This command and control vehicle observes the activity of the target vessel and tracks its location and speed. The other robots participating in the mission standby at their deployment positions as specified by the user during mission specification. When the target vessel gets sufficiently close to the expected interception area, the leader robot creates a call for proposals detailing the interception and passes it to the CNP initiator. The CNP initiator then broadcasts this call for proposals and handles the resulting CNP negotiation process (described in section III) with the deployed robots. The deployed robots

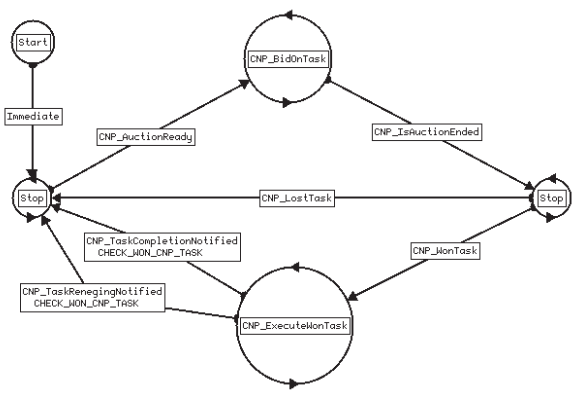

Fig. 11. FSA generated for a bidder robot. Behaviors within in it are in support of runtime task allocation.

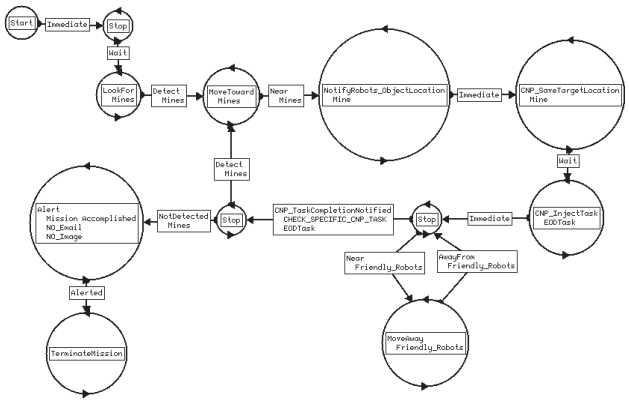

Fig. 12. FSA generated for leader robot. Its behaviors are largely for observing environmental state and injecting tasks based on that state.

calculate their bids based on enemy proximity, velocity, etc. Once the CNP negotiation process has been completed, the robot submitting the highest bid is assigned the task. The lead robot continues to monitor the mission area for additional tasks or mission completion.

The use of CNP in this design is similar to that used by Gerkey and Matarić's allocation system MURDOCH [21] [22]. The major difference between the task allocation systems lies in our use of contract reneging in the CBR and runtime CNP design. This allows both the task initiator and task executor to monitor and initiate the reassignment of tasks at anytime.

C. Design Comparison

While both designs use a common set of techniques to perform mission specification and task allocation, the point at which the integration occurs results in differing capabilities, strengths, and weaknesses. The CBR and pre-mission CNP design's major strength lies in its ability to aid the user during the mission specification process. Through the retrieval and adaptation of previously stored missions, the CBR Wizard reduces the need for the user to create complex FSA representations of the mission by hand. The task allocation in this design further eliminates the burden of generating a

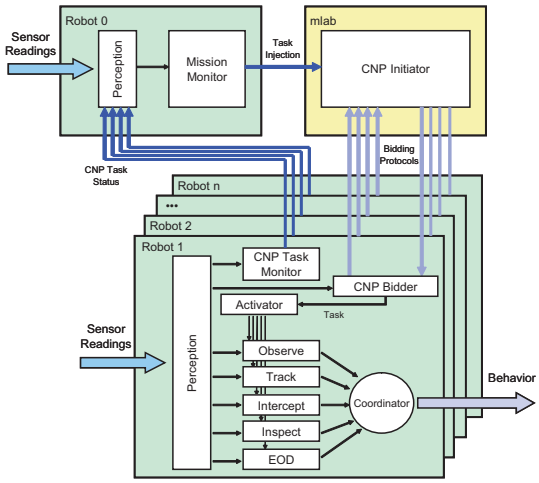

Fig. 13. Architectural framework of the execution phase for the CBR and runtime CNP design

task-to-robot mapping manually. In addition, using CBR and CNP in this manner allows the user to inspect and fine-tune the resulting mission plan prior to execution if necessary. Finally, the pre-mission design affords preliminary verification concerning the completion of the mission given the available resources. All of these components support the design of complex missions by less sophisticated human operators.

The CBR and pre-mission CNP design is not without disadvantages, however. In this design the task assignment for each robot is fixed after mission creation. This may force a mission to be aborted if a robot performing a vital task fails, as dynamic reassignment is not supported. In addition, this design does not permit additional tasks to be injected opportunistically while the mission is executing, limiting its effectiveness in highly dynamic scenarios. Finally, the pre-mission system requires a level of user knowledge concerning the mission itself to be effective. The more accurate the user's knowledge regarding the tasks to be performed and their location, the more effective the task allocation process will be. Conversely, a user with inaccurate information concerning the tasks or locations will result in overall poorer mission performance.

Unlike the pre-mission design, the CBR and runtime CNP system delays task allocation until mission execution commences. This delay provides a number of capabilities not present in the pre-mission design. Dynamic addition of tasks at runtime is supported directly. In addition, through contract reneging, the team can react to failures or environmental changes without compromising the mission's success. Knowledge concerning the tasks to be performed and their location is also minimized as the task allocation process provides the robot-to-task mapping at runtime.

The two major weaknesses inherent in the CBR and runtime CNP design are in part due to its dynamic nature. The first is its inability to provide pre-mission verification of the achievability of a mission. Without additional information concerning the tasks composing the mission, no guarantees can be given that

the available robots can accomplish the tasks. In addition, the runtime design requires relatively frequent amounts of communication throughout the mission. In adverse communication environments, mission performance may degrade as robots are unable to receive offers or provide status updates.

V. Evaluation

The architectures outlined here were evaluated from two perspectives. The CBR and pre-mission CNP system was evaluated via a usability analysis while the CBR and runtime CNP system's performace was evaluated empirically. We apologize, but due to space constraints, we can not include the detailed results in this paper but encourage readers to refer to the appropriate sources for discussion of the results. A brief summary of the results follow:

We performed a formal Goals, Operators, and Method, and Selection rules (GOMS)[23] analysis of the CBR and pre-mission CNP system along with the base mission specification system. The results of this analysis show that the CBR and pre-mission CNP system results in mission creation times significantly lower then that of the base system, especially as the number of robots and tasks increase [24].

Our empirical evaluation of the CBR and runtime CNP system examines the effect of the CNP allocation over a variety of metrics in naval mine countermeasure missions as well as target intercept missions. In addition, a series of experiments in which the CBR and runtime CNP system is compared against a system in which the ability to dynamically reallocate resources has been lesioned is examined. Results indicate the CBR and runtime CNP perform significantly better over a number of mission metrics when compared to the baseline systems [25].

VI. Conclusions and Future Work

This work presented two different approaches for combining case-based reasoning mission specification with a contract net protocol-based task allocation system. In the first, a CBR planner aids in the generation of complex plans based on task-level input from the user. A CNP-based task allocation system then automatically generates mappings from the the available robots to the tasks during mission specification. The CBR planner then retrieves and adapts similar cases to generate mission plans for accomplishing the tasks.

In the second approach, the CBR planner provides mission FSAs encapsulating behaviors for participating in CNP-based task allocation as well as the steps for accomplishing these tasks. It is shown that this design allows for the generation of missions in which user knowledge requirements are minimized.

Future work will investigate a third possibility for integration of CBR-based mission specification and CNP-based task allocation. In this third design, task allocation occurs during the CBR Wizard's case generation and storage. This future investigation will study how the pre-mission and runtime systems may interoperate to allow for the creation of increasingly sophisticated multi-robot missions.

Acknowledgment

This research was funded as part of NAVAIR's Intelligent Autonomy program.

References

[1] C. Le Pape, "A combination of centralized and distributed methods for multi-agent planning and scheduling," in *Proceedings of ICRA*, 1990, pp. 488–493.

[2] S. Botelho and R. Alami, "Plan-based multi-robot cooperation," in *Proceedings of the Dagstuhl workshop on plan-based control of robotic agents*, 2001.

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

[4] P. Scerri and P. Reed, "The EASE actor development environment," in *Proceedings of the Workshop of the Swedish AI Society*, 1999.

[5] L. Parker, "ALLIANCE: An architecture for fault-tolerant multi-robot cooperation," *IEEE Transactions of Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[6] S. Botelho and R. Alami, "M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Proceedings of ICRA*, 1999, pp. 1234–1239.

[7] B. Gerkey and M. Mararic, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[8] L. Parker, "Heterogeneous multi-robot cooperation," Ph.D. dissertation, MIT, 1994.

[9] R. Zlot, A. Stentz, and M. Dias, "Multi-robot exploration controlled by a market economy," in *Proceedings of ICRA*, 2002, pp. 3016–3023.

[10] M. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," Carnegie Mellon University, Tech. Rep. CMU-RI-TR-05-13, 2005.

[11] *MissionLab: User manual for MissionLab version 5.0*, Georgia Tech Mobile Robot Laboratory, 2002.

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

[13] D. MacKenzie and R. Arkin, "Evaluating the usability of robot programming toolsets," *The International Journal of Robotics Research*, vol. 17, no. 4, pp. 381–401, 1998.

[14] Y. Endo, D. MacKenzie, and R. Arkin, "Usability evaluation of high-level user assistance for robot mission specification," *IEEE Transactions of Systems, Man, and Cybernetics*, vol. 34, pp. 168–180, 2004.

[15] L. Moshinka, Y. Endo, and R. Arkin, "Usability evaluation of an automated mission repair mechanism for mobile robot mission specification," in *HRI 2006*, 2006.

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

[17] R. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. 29, no. 12, 1980.

[18] S. Sariel and T. Balch, "A distributed multi-robot cooperation framework for real time task achievement," in *Proceedings of DARS 8*, 2006.

[19] T. Lemarie, R. Alami, and S. Locroix, "A distributed task allocation scheme in multi-UAV context," in *Proceedings of ICRA*, 2004.

[20] FIPA, "FIPA contract net protocol specification," Available at http://www.fipa.org, 2000.

[21] B. Gerkey and M. Mataric, "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions of Robotics and Automation*, vol. 18, no. 5, pp. 758–768, 2002.

[22] B. Gerkey and M. Matarić, "Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination," in *Proceedings of ICRA*, 2002, pp. 464–469.

[23] B. John and D. Kieras, "The GOMS family of user interface analysis techniques: Comparision and contrast," *ACM Transactions on Computer-Human Interaction*, vol. 3, no. 4, 1996.

[24] A. Wagner, Y. Endo, P. Ulam, and R. Arkin, "Multi-robot user interface modeling," in *Proceedings of DARS 7*, M. Gini and R. Voyles, Eds. Springer, 2006, pp. 237–248.

[25] P. Ulam, Y. Endo, A. Wagner, and R. Arkin, "Integrated mission specification and task allocation for robot teams - testing and evaluation," Georgia Institute of Technology, Tech. Rep. GIT-GVU-07-02, 2007.