Coordinating Robot Teams for Disaster Relief

Mark Roberts[1], Thomas Apker[2], Benjamin Johnson[1], Bryan Auslander[3], Briana Wellman[4] & David W. Aha[2]

[1]NRC Postdoctoral Fellow; Naval Research Laboratory, Code 5514; Washington, DC 20375
[2]Navy Center for Applied Research in AI; Naval Research Laboratory, Code 5514; Washington, DC 20375
[3]Knexus Research Corporation; Springfield, VA 22153
[4]Department of Computer Science and Information Technology; University of the District of Columbia; Washington, DC 20008
{mark.roberts.ctr; thomas.apker; benjamin.johnson.ctr; david.aha}@nrl.navy.mil | bryan.auslander@knexusresearch.com | briana.wellman@udc.edu

Abstract

To perform complex tasks, a team of robots requires both reactive and deliberative planning. For reactive control, a restricted variant of Linear Temporal Logic called General Reactivity(1) can be used to synthesize correct-by-construction controllers in polynomial time, but they often ignore time and resource constraints to maintain tractable synthesis. For deliberation, hierarchical planning can be used to reason about time and resources. However, the coordination of reactive control and deliberation remains a challenge, which we accomplish through a set of *Coordination Variables*. We integrate these two approaches in the Situated Decision Process (SDP), a system that we are developing. The SDP will allow an Operator to control a team of semi-autonomous vehicles performing information gathering tasks for Humanitarian Assistance / Disaster Relief operations. We demonstrate that the SDP responds to a dynamic, open world while ensuring that vehicles eventually perform their commanded actions.

1. Introduction and Motivation

We study the problem of coordinating a team of semi-autonomous vehicles to gather information soon after a natural disaster strikes (e.g., the Philippines Typhoon). Emergency response personnel need updated information concerning the whereabouts of survivors, the condition of infrastructure, and recommend ingress and evacuation routes. Current practice for gathering this information relies heavily on humans (e.g., first responders, pilots, drone operators). A team of autonomous vehicles with sensors can facilitate such information gathering tasks, freeing humans to perform more critical tasks in Humanitarian Assistance / Disaster Relief (HA/DR) operations (US Dept. of Navy 1996). Coordinating robotic teams in a HA/DR operation presents unique challenges because each disaster is distinct. Thus, creating a single robotic controller is untenable because no single domain model can incorporate all the necessary steps for a mission. Personnel should be able to tailor possible vehicle missions to the current situation.

Reasoning on different granularities of abstraction and time scales is a common challenge in robotics (e.g., task planning vis-à-vis reactive planning). Robotic controllers often employ Finite State Automata (FSAs) to determine a robot's next action. Although they are fast to execute, hand-writing FSAs is error prone, tedious, and brittle. Recent advances apply a restricted variant of Linear Temporal Logic (LTL) called *General Reactivity(1)* to automatically synthesize FSAs in time cubic in the size of the final FSA (Bloem et al. 2012). But synthesis quickly becomes impractical for teams or dynamic environments. For example, Table 1 shows the number of seconds to synthesize FSAs for two vehicles assigned to survey two or more regions; limiting the FSA size is clearly justified.

Task planning is naturally suited to limit the FSA size for teams of vehicles (e.g., by pre-allocating missions to vehicles or by assigning vehicles to teams). We employ hierarchical decomposition (task) planning because it matches well with how humans view HA/DR operations (US Dept. of Navy 1996).

However, linking the task and reactive planning layers is a challenge. In particular, mission plans and vehicle controllers must expose useful abstractions to each other while allowing both to adjust to dynamic changes. To address this, we introduce the use of *Coordination Variables*, which integrate team mission goals with the vehicle controllers by providing abstraction predicates for vehicle commands, vehicle state (e.g., current behavior and health), and abstract vehicle sensor data. A secondary contribution of this paper is applying Goal Refinement (Roberts et al. 2014) to coordinate those vehicle missions in support of larger HA/DR operations.

# Survey Regions	# Propositions	Synthesis Time (seconds)
2	14	1.078
3	16	3.675
4	18	17.608
5	20	104.008

Table 1: *Synthesis times as the number of regions increases in the LTL specification as measured on a Windows 8.1 laptop, running a Core i5 processor.*

We describe an initial prototype of our Situated Decision Process (SDP), which manages vehicles that can perform three mission goals: (1) survey a region to assess roads, (2) locate a person, and (3) act as a communications relay for that person. We envision that a human Operator would input a set of HA/DR mission priorities to the SDP, which would aid the Operator in managing the vehicles to perform those missions. We describe a minimal HA/DR scenario (§2) and then detail the components of our SDP prototype (§3). We then demonstrate how the SDP responds to a dynamic, open world scenario while tracking progress toward mission goals (§4). We conclude by discussing related work (§5) and future work (§6).

2. HA/DR Scenario and Domain Model

Figure 1 shows an airport region (upper left) and, 3-5 km away from the airport, a Very Important Person (VIP) region (lower middle) that is centered on a particular building near the suspected location of the VIP. The VIP emits a radio signal (e.g., cell phone signal). Two fixed-wing air vehicles (in yellow) are tasked with assessing the two regions and finding the VIP. They carry Electro Optical and Radio Frequency sensors that activate when the target is within their sensor radius.

Although we designed significantly more challenging scenarios, we can use this minimal scenario to demonstrate the SDP's key capabilities, namely that: (1) the SDP can create new goals responding to an open world (e.g., it collectively responds to the VIP being found); (2) a vehicle can make decisions autonomously (e.g., a vehicle may begin relaying the VIP once found); (3) the SDP responds to vehicles making autonomous decisions (e.g., it notes the vehicle relaying instead of surveying when the VIP is found); and (4) the SDP can retask a vehicle to complete a mission (e.g., it retasks stalled missions to idle vehicles).

3. Situated Decision Process (SDP)

Figure 2 displays an abstraction of the SDP components we discuss in this paper. The SDP is partitioned into three abstract layers, each composed of components that perform specific tasks. The *UI Layer* (colored white) manages

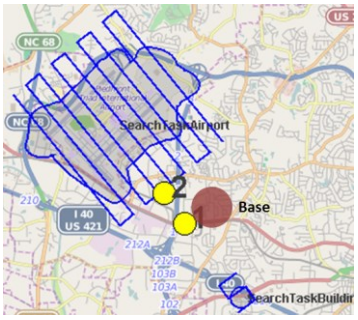

Figure 1: *Example airport and VIP regions. The base is located between the regions. Also shown are the trajectories (blue lines) for two vehicles (yellow dots).*

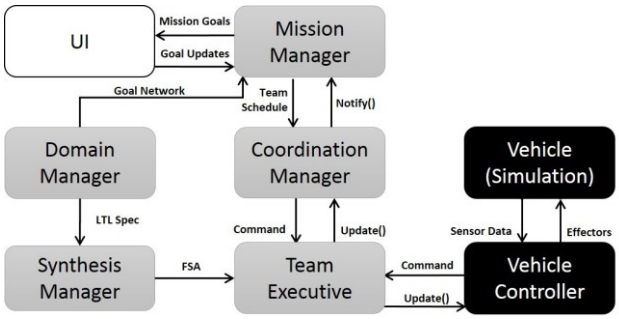

Figure 2: *An abstract view of the Situated Decision Process (SDP). Nodes are colored by layer: UI (white), Coordination (Gray), and Vehicle (black).*

interaction with the Operator. In this layer, the User Interface (UI) component collects input from a human Operator and conveys Operator feedback to the other components. However, it is not a focus of this paper and we will not discuss it further. The *Centralized Coordination Layer* (colored gray) focuses on the mission and task abstractions for the vehicle teams. The *Distributed Layer* (colored black) manages the vehicles or vehicle simulation. We now detail the components in the Centralized Coordination and Distributed layers.

3.1 Domain Manager

To construct a domain model, we elicited feedback from three Navy Reservists who have flown in or commanded HA/DR operations. We then encoded the domain knowledge as a Hierarchical Goal Network (Shivashankar et al. 2013; Geier & Bercher 2012). Figure 3 displays the goal network for the scenario in Figure 1. The root decomposes into two top-level operational goals "Logistics Operations" and "Security Operations." The former of these decomposes into "Assess Infrastructure" while the latter decomposes into a goal to "Maintain VIP Safety."

Operational subgoals eventually decompose into AchieveTeamMission goals, which themselves decompose into the VehicleMission goals associated with specific

Goal Description	Goal Type
AACS Domain Root	OperationalGoal
Logistics Operations	OperationalGoal
Assess Infrastracture	OperationalGoal
Assess Airport LZ	AchieveTeamMission
Mission:Assess Airport LZ	VehicleMission
Security Operations	OperationalGoal
Maintain VIP safety	OperationalGoal
Assess VIP Region(s)	AchieveTeamMission
Mission:Assess VIP Region(s)	VehicleMission
Relay VIP if found	AchieveTeamMission
Mission:Relay VIP if found	VehicleMission

Figure 3: *Goal decomposition during an SDP run.*

teams. These most primitive goals of this goal network are intended to match with tasks that the vehicles can perform. The goal network presented here is hand-coded, but we plan to implement this model in the ANML language (Dvorák et al. 2014) and integrate a full planning system in the SDP. The SDP will eventually guide vehicles in cooperation with its Operator(s), but in this paper we assume static mission goals, a fixed number of vehicles, and a fixed allocation of the vehicles to tasks.

3.2 Mission Manager and Goal Refinement

The Mission Manager decomposes the high-level mission goals provided by the user (e.g., regions of interest, overall vehicle tasks, and available vehicles) into primitive goals for the vehicle teams. One of the challenges in coordinating task and reactive planning is unifying the goals across the system so that goals can be tracked during execution. To unify goals, we employ a theoretical model called *Goal Refinement* (Roberts et al. 2014) that builds on previous literature in Goal Driven Autonomy (Klenk et al. 2013), which is a model of Goal Reasoning (Vattam et al. 2013). Goal Refinement incorporates recent perspectives on the actor (Ghallab, Nau, and Traverso 2014) as well as deliberation functions (Ingrand and Ghallab 2014). Goal Refinement also complements a plan's lifecycle (e.g., (Pollack and Horty 1999; Myers 1999)).

A central part of Goal Refinement is the Goal Lifecyle, shown in Figure 4. This lifecycle captures the possible decision points of goals in the SDP. Decisions consist of applying a *strategy* (arcs in Figure 4) that transitions a goal among *modes* (rounded boxes) in the lifecycle. The g's in the goal lifecycle correspond to goals (e.g., goals in the goal network of Figure 3), while x's correspond to expansions (e.g., decompositions of non-primitive goals, allocations and trajectories for primitive goals).

We focus our discussion the lifecycle strategies that we implemented for the SDP and the goal network in Figure 3. While this particular scenario does not exercise all the strategies of the goal lifecycle, the more advanced scenarios we designed exercises all the strategies.

When loading, the SDP automatically formulates and selects an initial goal to AchieveDomainLoaded (not

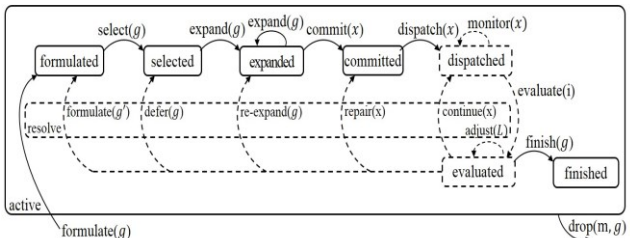

Figure 4: *The Goal Lifecycle by Roberts et al. (2014).*

shown). The expansion of this goal results in the "AACS Domain Root" goal being formulated and selected. A non-primitive domain goal is expanded by instantiating a sub-goal tree for the goal. The SDP commits to and dispatches the only expansion available for these goals, since this is a small example. These non-primitive goals remain in a dispatched state until their subgoals finish.

Expanding an AchieveTeamMission goal results in a specific VehicleMission goal, which includes details regarding a proposed allocation of a vehicle to a specific trajectory (cf. the lawnmower flight paths of Figure 1). Once the VehicleMission details are approved – either automatically or by the Operator – the Mission Manager commits and dispatches the proposed expansion of the VehicleMission for execution. The Coordination Manager and Team Executive then begin sending vehicle commands. The VehicleMission goal remains dispatched until new information (e.g., a progress update) causes it to become finished or need some other resolve strategy.

The Mission Manager has triggers to monitor the dispatched goals so that it will notice if the goal is stalled or completed by the executive. The Mission Manager uses a repair strategy on the original vehicle allocation to retasking a vehicle for a stalled VehicleMission,

3.3 Synthesis Manager

The Synthesis Manager takes as input an LTL specification and synthesizes an FSA for the Vehicle Controller. To perform synthesis, we extend the work of Kress-Gazit et al. (2009) using portions of the LTLMop toolkit (Jing et al. 2012). LTL compactly encodes the mapping between the Centralized Coordination Layer's commands and the robots' behaviors, along with any restrictions on their capabilities. This allows SDP to generate reactive controllers that match a specification without requiring a hand-coded FSA. Space limitations preclude a full exposition of General Reactivity(1) or LTL Synthesis.

Figure 5 provides a readable example of the LTL English equivalent. "Goals" in an LTL specification describe behaviors that the vehicle is required to perform infinitely often (i.e., always eventually do behavior). Line 1 specifies the vehicle actions, where each action corresponds to a vehicle behavior that performs the intended action. Line 2 details the LTL sensors, which are

```
1. Actions: ExploreA, Relay, ExploreA, Search, Recharge
2. Sensors: DoSearchA, DoSearchB, DoRelay, DoSearchReactive,
            DoReturn, VIPDetected, LowFuel
3. robot starts in Base with false; environment starts with DoSearchA
4. if you sensed LowFuel then always LowFuel
5. if you are sensing DoRelay then
     always not DoReturn and not DoSearchReactive
6. if not LowFuel and DoSearchA and not DoRelay then
     infinitely often ExploreA
7. if not LowFuel and DoSearchB and not DoRelay then
     infinitely often ExploreB
8. if not LowFuel and DoSearchA and DoRelay then
     infinitely often ExploreA or Relay
9. if not LowFuel and DoSearchB and DoRelay then
     infinitely often ExploreB or Relay
10. if not LowFuel and DoSearchReactive then infinitely often Search
11. if not LowFuel and DoReturn then infinitely often Base
12. if LowFuel then infinitely often Recharge
13. do ExploreA iff robot is in AreaA and not sensing LowFuel and
     not sensing (VIPDetected and DoRelay) and sensing DoSearchA
14. do ExploreB iff robot is in AreaB and not sensing LowFuel and
     not sensing (VIPDetected and DoRelay) and sensing DoSearchB
15. do Relay iff you are not sensing LowFuel and sensing
     VIPDetected and sensing DoRelay
16. do Recharge iff you are in Base and sensing LowFuel
```

Figure 5: *The approximate English description used for synthesis of the controllers in the demonstration.*

the *observations* that the robot can take (see §3.4 for more detail). Line 3 specifies the initial conditions. Line 4 specifies a safety condition concerning LowFuel. Line 5 specifies that at least one command is active. Lines 6-12 perform "goal" selection based on sensors. Lines 13-16 perform conditional action selection depending on the state**.** A region file (not shown) specifies the regions and their adjacencies, which include Base, AreaA, and AreaB.

3.4 Coordination Manager & Coordination Variables

To link the mission goals and vehicle FSAs, we use *Coordination Variables*, which capture the key abstractions of each layer for each other. These variables are linked to the VehicleMission goals in the Mission Manager (see Figure 3) and to the sensors of the LTL specification (Figure 5, line 2). Each layer responds differently to these variables during execution.

Command Variables are provided to control the vehicle behavior. For our example scenario the commands are *DoSearchA, DoSearchB, DoRelay, DoSearchReactive,* and *DoReturn.* The Mission Manager uses these variables to command a vehicle to perform a specific task. The Team Executive can send progress updates using these variables.

Mission Variables allow the Vehicle Controller and Team Executive to send updates about notable events. The only Mission Variable is *VIPDetected*.

Health Variables allow the Vehicle Controller and Team Executive to send updates about vehicle status. For this scenario the only Health Variable is *LowFuel*.

3.5 Team Executive, Vehicle Control and Simulation

The Team Executive sets/unsets the Command sensors on the vehicles' FSAs based on the schedule developed by Mission Manager. It also maintains each vehicle's status and sensor information, allowing the Centralized Coordination Layer to monitor the team's progress and detect notable events.

Vehicles are controlled using hybrid controllers that read the FSA and reactively select a physics-inspired behavior implemented using physicomimetics (Apker et al. 2014). We simulated the scenario in MASON (Luke et al. 2005).

4. Demonstration

To demonstrate how the SDP responds to a notable event in an open world we generate 30 scenarios based on Figure 1. We select 30 random airports from OpenStreetMaps data for North Carolina (Geofabrik 2014) and then select buildings within 3-5 kilometers of the airport. Buffer regions of 300 meters around the airport and the building serve as the airport and VIP regions, respectively. Each run completes when (1) both regions are completely surveyed and the VIP is found or (2) the simulation reaches 35,000 steps. Each step is approximately one second of real time simulation.

At the start of the scenario, one vehicle is assigned to assess the Aiport Region, denoted by AirportVehicle, and the other vehicle is assigned to the VIP Region, denoted VIP Vehicle. Vehicles return to the base when their fuel is sufficiently low. Vehicle behavior depends on whether the vehicles can retask themselves to relay when the VIP is found (denoted *+Relay*) or they do not relay (*–Relay*). Regardless of whether a vehicle begins relaying, the Mission Manager should always create a new "Relay VIP" goal when the VIP is found. The Mission Manager behavior depends on whether it is allowed to retask a vehicle (*+Retask*) or not (*–Retask*).

Condition 1: Find VIP (-Relay -Retask) provides a baseline. In it the vehicles detect the VIP and a new goal to relay the VIP appears when the VIP is found. Getting the SDP to do something meaningful with the "Relay VIP" goal is our next condition.

Condition 2: Relay VIP (+Relay -Retask) demonstrates that a vehicle can retask itself with a new goal by automatically relaying the VIP once found. The retasking is embedded in the Vehicle Controller (see Figure 5, line 15). However, this change of behaviors needs to be shown in the goal network, where the goal "Mission: RelayVIP" should appear after the VIP is found. However, nothing is done with the new goal and VIP Vehicle does not complete the entire survey of the VIP region because it switches its own task to relaying.

Condition 3: Relay and Retask (+Relay +Retask). To address the problem of the VIP region remaining unfinished, the Centralized Coordination Layer is allowed to retask the Airport Vehicle so it finishes the VIP Region survey after completing its area first.

When we run the simulation on the three conditions, we observe exactly the expected results. In every case, a new goal is observed in the Mission Manager after the VIP is found. In the Relay VIP condition, the VIP Vehicle begins relaying as expected, leaving the VIP Region unfinshed. When the Mission Manager is allowed to retask vehicles, we observe that all three missions complete.

5. Related Work

Planning trajectories for teams *a priori* to achieve a single objective requires solving a high dimensional optimization problem (Yilmaz et al. 2008) to compute optimal trajectories that are tightly coupled to the initial assumptions/goal. Bio-inspired and other reactive guidance strategies simplify this problem by using more goal-directed behaviors for area coverage (Liu and Hedrick 2011) and discrete target tracking (Haque et al. 2008; Kruecher et al. 2007). These behaviors rely on local measurements and instantaneous gradients to guide robots. Still, no behavior or trajectory generator can handle all contingencies a priori in complex, open environments.

A promising approach, inspired by animal behavior, uses FSAs for mobile robot guidance (Balch et al. 2006). Hand-coding an FSA for each execution of a robot is tedious and error prone. Kress-Gazit et al. (2009) instead synthesize an FSA from an LTL specification using a game theory approach (Bloem et al. 2014) in which the robot takes actions to achieve its goals against actions taken by the environment (i.e., the adversary). This strategy guarantees correct behavior if the LTL-spec is never violated, but synthesis is quadratic in the number of (environmental and sensing) goals (Bloem et al. 2012) and is intractable for large teams of robots. Our approach preselects missions for vehicles prior to the synthesis of an FSA, which reduces the size of the LTL specification and thus reduces the computational time for synthesis.

Goal refinement builds on the work in plan refinement (Kambhampati, Knoblock, & Yang 1995), which equates different kinds of planning algorithms in plan-space and state-space planning. Extensions incorporated other forms of planning and clarify issues in the Modal Truth Criterion (Kambhampati and Nau 1994). More recent formalisms such as Angelic Hierarchical Plans (Marthi et al. 2008) and Hierarchical Goal Networks (Shivashankar et al. 2013) can also be viewed as leveraging plan refinement. The focus on constraints in plan refinement allows a natural extension to the many integrated planning and scheduling systems that use constraints for temporal and resource reasoning.

The goal lifecycle bears close resemblance to that of Harland et al. (2014) and earlier work (Thangarajah et al., 2010). They present a goal lifecycle for BDI agents, provide operational semantics for their lifecycle, and demonstrate the lifecycle on a Mars rover scenario. Work by Winikoff et al. (2010) has also linked Linear Temporal Logic to the expression of goals. Our work differs in that it focuses on teams of robots rather than single agents.

Our approach of coordinating behaviors with constraint-based planning is inspired by much of the work mentioned by Rajan, Py, and Barriero (2013). Our Team Executive leverages the Executive Assistant of Berry et al. (2003).

6. Summary and Future Work

We detailed our implementation of a system, called the SDP, which links hierarchical task planning (i.e., a goal network) and reactive controllers by synthesizing correct-by-construction FSA vehicle controllers. The central contribution of this paper is an interface (i.e., the Coordination Variables) that allows task planning to control and receive feedback from a reactive layer. Our approach saves considerable computation during FSA synthesis. In a small demonstration, we showed that our implementation of the SDP adjusts to notable events (e.g., finding a VIP) or retasks vehicles to continue stalled missions when such events occur.

Future work will consist of further automating portions of the SDP and enriching the domain model. For example, we plan to extend the domain model to fully encode temporal and resource concerns similar to the TREX system (Rajan, Py, and Barriero 2013). We also plan to test the SDP in more challenging environments, which will require allowing vehicles to set their own command sensors autonomously and moving the Team Executive to the robotic platforms with sufficient computational power. Finally, we plan to incorporate richer sensor models and higher-fidelity simulations. Ultimately, we plan to run the SDP on actual vehicles and perform user studies on its effectiveness in helping an Operator coordinate a team of vehicles in Disaster Relief.

Acknowledgements

Thanks to OSD ASD (R&E) for sponsoring this research. The views and opinions in this paper are those of the authors and should not be interpreted as representing the views or policies, expressed or implied, of NRL or OSD. We also thank the reviewers for very helpful feedback that helped improve the paper.

References

Apker, T., Liu, S.-Y., Sofge, D., and Hedrick, J.K. (2014). Application of grazing-inspired guidance laws to autonomous information gathering. Proc. of the Int'l Conference on Intelligent Robots and Systems (pp. 3828-3833). Chicago, IL: IEEE Press.

Balch, T., Dellaert, F., Feldman, A., Guillory, A., Isbell, C.L., Khan, Z., Pratt, S.C., Stein, A.N., & Wilde, H. (2006). How multirobot systems research will accelerate our understanding of social animal behavior. Proc. of the IEEE, 94(7), 1445-1463.

Berry, P., Lee, T. J., & Wilkins, D. E. (2003). Interactive execution monitoring of agent teams. Journal of Artificial Intelligence Research, 18, 217–261.

Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y. (2012). Synthesis of Reactive(1) designs. Journal of Computer and System Sciences, 78(3), 911–938.

Dvorák, F., Bit-Monnot, A., Ingrand, F., & Ghallab, M. (2014). A Flexible ANML Actor and Planner in Robotics. In Working Notes of the Planning and Robotics Workshop at ICAPS. Portsmouth, NH: AAAI.

Geofabrik. OpenStreetMap Data Extracts. (2014) Accessed from http://download.geofabrik.de/index.html.

Ghallab, M., Nau, D., & Traverso, P. (2014). The actor's view of automated planning and acting: A position paper. Artificial Intelligence, 208, 1–17.

Geier, T. & Bercher, P. (2011). On the decidability of HTN Planning with task insertion. In Proc. of the 22nd Int'l Joint Conf. on AI. (pp. 1955-1961). Barcelona: AAAI.

Harland, J., Morley, D., Thangarajah, J., & Yorke-Smith, N. (2014). An operational semantics for the goal life-cycle in BDI agents. Autonomous Agents and Multi-Agent Systems, 28(4), 682–719.

Haque, M., Rahmani, A, & Egerstedt, M. (2010). Geometric foraging strategies in multi-agent systems based on biological models. Proc. of the IEEE Conf. on Decision and Control (pp. 6040-6045). Atlanta: IEEE.

Ingrand, F., & Ghallab, M. (2014). Robotics and artificial intelligence: A perspective on deliberation functions. AI Communications, 27(1), 63-80.

Jing, G., Finucane, C., Raman, V. and Kress-Gazit, H. (2012). Correct high-level robot control from structured English. Proc. of the Int'l Conf. on Robotics and Automation (pp. 3543-3544), St. Paul, MN: IEEE Press.

Kambhampati, S., Knoblock, C.A., & Yang, Q. (1995). Planning as refinement search: A unified framework for evaluating design tradeoffs in partial-order planning. Artificial Intelligence, 76, 168-238.

Kambhampati, S. & Nau, D. (1994). On the nature of modal truth criteria in planning. Proc. of the 12th Nat'l Conference on AI (pp. 67-97). Seattle, WA: AAAI Press.

Klenk, M., Molineaux, M., & Aha, D.W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. Comp. Intell., 29(2), 187-206.

Kress-Gazit, H., Fainekos, G.E., & Pappas, G.J. (2009). Temporal logic based reactive mission and motion planning. Transactions on Robotics, 25(6), 1370-1831.

Liu, S-Y., & Hedrick, J.K. (2011). The application of domain of danger in autonomous agent team and its effect on exploration efficiency. Proc. of the Am. Control Conf. (pp. 4111-4116). San Francisco: IEEE.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., & Balan, G. (2005). Mason: A multiagent simulation environment. Simulation, 81(7), 517-527.

Marthi, B, Russell, S., & Wolfe, J. (2008). Angelic hierarchical planning: Optimal and online algorithms. Proc. of the Int'l Conf. on Automated Planning and Scheduling (pp. 222-231). Menlo Park, CA: AAAI.

Myers, K.L. (1999). CPEF: A continuous planning and execution framework. AI Magazine, 20(4), 63-69.

Pollack, M.E., & Horty, J. (1999). There's more to life than making plans: Plan management in dynamic, multiagent environments. AI Magazine, 20, 71-83.

Rajan, K., Py, F., & Barreiro, J. (2012). Towards deliberative control in marine robotics. In Marine Robot Autonomy (pp. 91–175). New York, NY: Springer.

Roberts, M., Vattam, S., Alford, R., Auslander, B., Karneeb, J., Molineaux, M., Apker, T., Wilson, M., McMahon, J., & Aha, D.W. (2014). Iterative goal refinement for robotics. In Working Notes of the Planning and Robotics Workshop at ICAPS. Portsmouth, NH: AAAI.

Shivashankar, V., Alford, R., Kuter, U., & Nau, D. (2013). The GoDeL planning system: A more perfect union of domain-independent and hierarchical planning. Proc. of the 23rd Int'l Joint Conference on AI (pp. 2380-2386). Beijing, China: AAAI Press.

Thangarajah, J., Harland, J., Morley, D., & Yorke-Smith, N. (2011). Operational behaviour for executing, suspending, and aborting goals in BDI agent systems. In Declarative Agent Languages and Technologies VIII (pp. 1–21). Toronto, Canada: Springer.

U.S. Department of the Navy. (1996) Humanitarian assistance/disaster relief operations planning, (Technical Report TACMEMO 3-07.6-05). Washington, D.C.: Government Printing Office.

Vattam, S., Klenk, M., Molineaux, M., & Aha, D. W. (2013). Breadth of approaches to goal reasoning: A research survey. In D.W. Aha, M.T. Cox, & H. Muñoz-Avila (Eds.) Goal Reasoning: Papers from the ACS Workshop (Tech. Report CS-TR-5029). College Park, MD: Univ. of Maryland, Dept. of Computer Science.

Winikoff, M., Dastani, M., & van Riemsdijk, M. B. (2010). A unified interaction-aware goal framework. In Proc. of ECAI (pp. 1033–1034). Lisbon, Portugal: IOS Press.

Yilmaz, N.K, Evangelinos, C., Lermusiaux, P., & Patrikalakis, N.M. (2008). Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming. IEEE Journal of Oceanic Engineering, 33(4), 522–537.