

## Goal Reasoning for an Autonomous Squad Member

---

**Kellen Gillespie**

KELLEN.GILLESPIE@KNEXUSRESEARCH.COM

**Matthew Molineaux**

MATTHEW.MOLINEAUX@KNEXUSRESEARCH.COM

**Michael W. Floyd**

MICHAEL.FLOYD@KNEXUSRESEARCH.COM

Knexus Research Corporation, Springfield, VA 22153 USA

**Swaroop S. Vattam**

SWAROOP.VATTAM.CTR.IN@NRL.NAVY.MIL

NRC Research Associate; Naval Research Laboratory (Code 5514), Washington, DC 20375 USA

**David W. Aha**

DAVID.AHA@NRL.NAVY.MIL

Naval Research Laboratory (Code 5514), Washington, DC 20375 USA

### Abstract

Autonomous agents are beginning to play larger roles within team-oriented tasks and missions in various domains. Many reasoning agents are designed to perform in or assist with a single, static goal or task within an environment. Our aim is to design and develop an autonomous squad member that assists a squad with conducting a surveillance mission by identifying and reacting dynamically to changing situations and goals. We present a goal reasoning system for this agent that integrates natural language processing, explanation generation, and plan recognition components to recognize these changing situations and the squad's responses to them. Our system uses goal selection and plan generation to respond to such changes. We describe the architecture we use to integrate these components and provide a case study that demonstrates how they work together to make a robust and adaptive autonomous agent.

### 1. Introduction

Robots are increasingly being added to teams to improve their ability to accomplish specific tasks and missions (ARL, 2011; Kott et al., 2010). Most instances involve a single task or objective for the agent or team to complete that is static and uninterrupted. This may suffice for simple tasks and environments, but in more realistic situations the team's goals are dynamic and can be interrupted or changed at any time.

Our objective is to design and develop an autonomous squad member (ASM) that can accompany and assist a squad of soldiers on military missions. These missions can be, and often are, conducted in hazardous and hostile areas. In such environments, unexpected events (e.g., encountering enemy fire, explosives, and other obstacles) can occur at any moment without warning. Therefore, the ASM must identify and react to highly dynamic and unpredictable environments to coordinate with its team. Furthermore, as its human teammates will typically react quickly and instinctively to such changes, the ASM must also recognize rapid changes in team behaviors.

Meeting these needs requires the ASM to have multiple reasoning capabilities. First, it must recognize any relevant situational and environmental changes that have occurred. We address this need by bolstering standard observational capabilities (e.g., static environmental and map information, a priori mission knowledge) with natural language understanding (NLU). Next, the agent must recognize local events and the actions its squad is performing based on this observational data. We handle this problem by using explanation generation to abductively infer actions and events that may have been responsible for the ASM's observations. Using this information, the ASM can infer the current goals and plans of other squad members by performing plan recognition in situ using the history of recognized actions. Based on the team's inferred goals, the ASM's goal selection algorithm should then choose a new goal for the agent that coordinates with the team. Plan generation then creates a sequence of actions to accomplish this goal. Finally, the ASM should act in accordance with its plan. We detail each of these steps and their supporting techniques in this paper.

In Section 2 we examine related goal reasoning (GR) systems and discuss military programs that promote autonomous agents as teammates. Section 3 provides a more detailed description of the ASM domain and its inherent challenges, while Section 4 describes our GR process and its major components. Section 5 introduces our demonstration scenario and showcases an initial proof of concept in that domain. We conclude and discuss future research plans in Section 6.

## 2. Related Work

Numerous GR agents have been created and applied to control unmanned autonomous systems, some of which adapt to unexpected events and situations. For example, Coddington et al. (2005) describe MADbot, an agent that can change its goals dynamically. It offers insight into supplying agents with underlying drives and (primarily internal) motivations that can initiate goal change. We agree with the importance of motivated goal changes, but aim to generate goals based not only on internal motivations but also external factors such as the actions of team members and exogenous events in the environment.

Other GR agents have been designed to respond to dynamic problems and tasks (e.g., Talamadupula et al., 2011). While these reason about situation changes, they do not always detect and recognize them in intuitive ways. Instead, they often require structured interfaces to communicate goal changes with the agent. For instance, Talamadupula et al. use a "Problem Update" structure to communicate new sensory information and goal changes to the agent, but in real-world situations such information is not always made available so easily and quickly.

Work has been done, however, to extend these technologies with natural language processing and understanding techniques. Cantrell et al. introduce a full architecture (DIARC) that is designed to handle natural language and dialogue processing (2012). While this work does integrate natural language into a larger goal reasoning architecture, it mainly focuses on language and capability-related commands. For instance, telling an agent it can open a door to enter a room (Cantrell et al., 2012). We feel it is important for the ASM, in its military mission-related domain, to understand language not necessarily directed at it or in command form. Consider a scenario where one of the agent's teammates encounters an improvised explosive device (IED) near a convoy: the agent may simply sense the teammate yell "*IED*" or "*Explosive*." Therefore,

while there has been notable NLU work applied to GR, we feel that our domain requires handling more simple and overheard styles of language. We respond to this challenge by forming domain-appropriate NLU techniques to incorporate into the ASM.

A related problem to explanation is diagnosis of discrete-event systems (Sampath et al., 1995), also referred to as history-based diagnosis (Gspandl et al., 2011), which finds action or event histories that account for a series of observations including observed events. Aside from representations, our formal model of explanations differs from the standard discrete-event system diagnosis model in the following ways: (1) we distinguish actions from events (which are deterministic), to better understand which actors are responsible for which actions and to predict unavoidable consequences; (2) we provide a formal model of ambiguous occurrences and a way to characterize their correctness, and (3) we assume that only partial states, and never actions or events, are observable. Diagnosis efficiency is considered a major issue in this community; recent efficient systems convert diagnosis problems to satisfiability or planning problems and adopt efficient techniques for solving the new problem (Grastien et al., 2011; Sohrabi et al., 2010).

Aside from more general GR research, several military programs are investigating the use of autonomous agents alongside teams of humans. One of these, the Safe Operations in Urban and Complex Environments (SOURCE) (Kott et al., 2010) Program, aims to develop collaborative autonomous agents to assist warfighters in dynamic environments in a safe and trustworthy manner. Similarly, the Robotic Collaborative Technology Alliance Army Program calls for the research and development of perceptive, intelligent autonomous vehicles that can interact with human teams (ARL, 2011). However, many of the collaborative systems proposed by these programs have not progressed beyond single-objective tasks with little to no external interference or danger. This presents a significant limitation, as more realistic missions like reconnaissance tasks (Section 5) are dynamic and can involve unexpected and dangerous events. A primary objective of our work on the ASM is to address such events.

We extend this diverse body of research on GR and autonomy by laying the groundwork for a more adaptive ASM.

### **3. Challenges in Real World Domains and Missions**

The domain and mission characteristics for our proposed ASM present realistic and difficult challenges to GR and autonomy in general. When teams of warfighters undergo missions, such as reconnaissance, the mission environment is often hazardous and almost always dynamic.

Dynamic environments pose a major threat to autonomous systems that cannot adapt on the fly to changing conditions. For instance, a priori information about the world may be given in the form of maps and satellite images that may be inconsistent with the real-time state of the mission environment. Unforeseen obstacles, such as downed trees and boulders, can block a mission route. For an agent to collaborate effectively with its team it must react to dynamic environments such as these, and plan to act and assist in whatever way it can (e.g., help pull the tree out of the road or calculate a new route to the current destination).

In addition to being dynamic, in these environments a team may encounter unexpected and anomalous situations throughout a given mission (e.g., a squad can encounter enemy mortar fire

while en route to a given destination). An agent cannot always be expected to perceive or sense the physical mortar shells being fired, nor can its teammates be expected to stop what they are doing and tell the agent what is happening (e.g., via some structured user interface). Instead, a robust autonomous agent must understand and explain what is happening around it based on observations and context clues, including those coming from its teammates in the form of natural language. For example, recognizing that its squad leader has yelled “*Mortar fire incoming*” can provide insight into what is currently happening, as well as how the rest of the team is expected to respond. The ASM can process these events without requiring focused operator input.

While we do not claim to have a complete solution to these issues that are inherent in realistic mission scenarios, but our system takes steps towards handling such scenarios.

#### 4. Autonomous Squad Member Goal Reasoning Process

Our GR process includes five primary steps, represented by five distinct components: (1) a *Natural Language Interpreter*, (2) an *Explanation Generator*, (3) a *Plan Recognizer*, (4) a *Goal Selector*, and (5) a *Plan Generator*. Figure 1 displays the ASM’s decision cycle, which involves using these components. Observations originate from the world (or simulation) the agent inhabits, which we label as the *Environment*. Ultimately, the agent’s actions feed back into the *Environment*, completing the decision cycle. We outline and describe each of the five primary components in the remainder of this section.

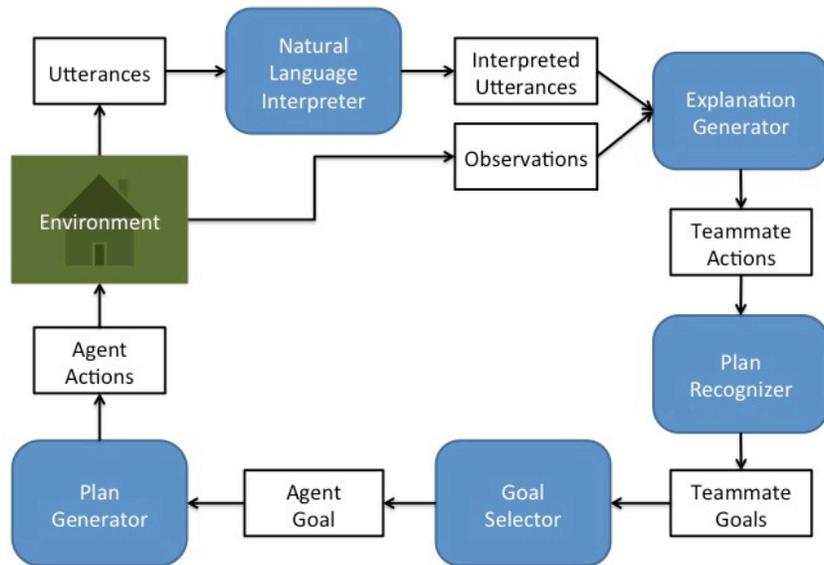


Figure 1. The Autonomous Squad Member’s Goal Reasoning Process.

#### 4.1 Natural Language Interpreter

The ability to interpret natural language uttered by teammates is a key capability for understanding nearby events, both expected and anomalous, in many domains. We begin the interpretation process by parsing spoken utterances using a domain-specific grammar and a chart parser from the Natural Language Toolkit (NLTK) (Bird, 2006). We keep the grammar small by considering only semantic categories related to the mission and potential anomalies that could be encountered (e.g., utterances regarding the recon mission, sniper encounters, teammate status). While a grammar is inherently limited in its vocabulary, the structured and well-known terminology of the military domain (e.g., NATO phonetic alphabet, device and role terminology) allows us to cover large portions of standard mission-related conversation and phrases in a compact manner. This grammar, coupled with a chart parser from the NLTK, produces parses for the raw utterances heard by the ASM. It reformulates these parses into semantically tagged structures that denote utterance types (e.g., mission-related, anomalous-event-related, squad-related) and content. Finally, these semantic structures are mapped to ontological information representing the implications and meanings of the utterance(s) being processed. We refer to this generated information as *interpreted* utterances.

Table 1 displays a few example utterances that could be overheard during a mission, as well as some interpretations that could be generated from each. For example, if the ASM senses a teammate saying “*Enemy Sniper*,” it can assume the teammate means that there exists an enemy in the world that is holding a sniper rifle. Variables in Table 1 are indicated by a leading question mark (e.g., ?enemyA and ?teammateA), and are implicitly existentially quantified. As output, the language understanding process generates semantic interpretations in the form of facts about the world, which are passed to the Explanation Generator for further processing.

Table 1. Example semantic interpretations for several sample utterances.

Utterance	Example Interpretations
“ <i>Enemy Sniper</i> ”	(exists ?enemyA) (holding ?enemyA sniper-rifle) (is-self ?self) (on-team ?self ?teamA) (not (on-team ?enemy ?teamA))
“ <i>Incoming Mortar</i> ”	(exists ?enemyB) (holding ?enemyB mortar) (fired ?enemyB mortar)
“ <i>Man Down</i> ”	(exists ?teammateA) (on-team ?teammateA ?teamA) (is-self ?self) (on-team ?self ?teamA) (status ?teammateA injured)

#### 4.2 Explanation Generator

For the ASM to understand what its teammates are doing at any given moment, it needs to monitor its environment to recognize their individual actions. The Explanation Generator is

responsible for this task; it accepts as input state information about the world in the form of observations and interpreted utterances from the Natural Language Interpreter.

To perform this task, we use a modified version of the DiscoverHistory algorithm (Molineaux, Kuter, & Klenk, 2012). DiscoverHistory is appropriate because it is designed to function in a partially observable environment and infer exogenous events and actions based on static observations (Molineaux & Aha, 2014). Unlike most other algorithms for explanation generation, DiscoverHistory operates efficiently by incrementally constructing explanations, and by maintaining explanations as new observations arrive rather than processing large batches. Additionally, it recognizes the difference between actions and events, and can ascribe particular world changes to the actions of known actors. This is important to support plan recognition in circumstances where individual actors may have different and even competing goals.

DiscoverHistory detects and resolves inconsistencies in existing explanations by applying one of the following inconsistency resolution methods to update the explanation:

- Hypothesize a new event or action
- Bind an unbound variable
- Remove an event or action
- Assume a property of the initial state
- Constrain the ordering of two unordered occurrences

DiscoverHistory conducts a search through explanation space; at each step of the search, it selects an inconsistency and applies all possible inconsistency resolutions to find successor nodes in the search space. This search terminates when a pre-specified number of explanations are found with only *ambiguous inconsistencies*, which are inconsistencies that can be resolved by binding a variable in multiple ways, and are considered trivial because the required conditions are met by several existing objects.

To properly integrate the partial knowledge discovered by the Natural Language Interpreter, we extended DiscoverHistory with the ability to incorporate existential quantifiers in its observations, which can in turn be bound during the explanation generation process. This resolves, or *grounds*, the semantic interpretations generated by the Natural Language Interpreter. Binding steps in the explanation process unify the variables in the interpreted utterances with existing (or newly discovered) entities in the environment. To illustrate this process, let's revisit the example interpretation in Table 1, namely the "*Enemy Sniper*" utterance. Suppose the ASM is aware of an enemy sniper nearby, and has labeled it `person1`. DiscoverHistory would recognize that the assertion (`holding ?enemyA sniper-rifle`) is inconsistent with existing knowledge, and resolves that inconsistency by binding the variable `?enemyA` to the value `person1`. This reconciles the inconsistency, as (`holding ?enemyA sniper-rifle`) is supported by prior observations. This contextualizes the remaining information, allowing the inference that `person1`, who holds a `sniper-rifle`, is not on the same team as `self` (i.e., the robot). Thus, the situation-agnostic interpretation is integrated with other observations of the current environment.

The modified DiscoverHistory algorithm also uses the predicted plan from the Plan Recognizer (Section 4.3) to help infer actions. Assuming that these predictions are correct, this

speeds up the explanation generation process by removing several otherwise unnecessary search steps to find correct actions.

The primary output of the Explanation Generator is the set of actions that were inferred to be performed by the ASM’s teammates. These actions are then given as input to the Plan Recognizer.

### 4.3 Plan Recognizer

Once the ASM has inferred the actions of its teammates, it can attempt to identify the higher-level plans and goals they are trying to accomplish via these actions. The task of identifying an ASM’s plans based on their actions is referred to as *plan recognition*.

We extend and use a plan recognizer called the *Single-Agent Error-Tolerant Plan Recognizer (SET-PR)* (Vattam, Aha, & Floyd, 2014; 2015) to infer the plans (and goals) of the ASM’s teammates. SET-PR’s case-based approach to plan recognition is designed to robustly tolerate observational input errors, including missing, mislabeled and extraneous actions. This capability is especially useful when partial and imperfect observability conditions exist that are typical in the simulations and scenarios that we use to test our GR agent.

SET-PR learns to recognize plans from a given plan library  $\mathcal{C}$  (i.e., a set of cases), where each *case* is a tuple  $c = (\pi_0, g_0)$ ,  $\pi_0$  is a known plan, and  $g_0$  is its corresponding goal. Each case’s plan  $c.\pi_0$  is modeled as an *action-state sequence*  $\mathfrak{s} = \langle (\mathbf{a}_0, \mathbf{s}_0), \dots, (\mathbf{a}_n, \mathbf{s}_n) \rangle$ , where each action  $\mathbf{a}_i$  is a ground operator in the planning domain, and  $\mathbf{s}_i$  is a ground state obtained by executing  $\mathbf{a}_i$  in  $\mathbf{s}_{i-1}$ , with the additional caveat that  $\mathbf{s}_0$  is an initial state,  $\mathbf{a}_0$  is null, and  $\mathbf{s}_n$  is a goal state. Plan  $c.\pi_0$  does not store the propositional representation of  $\mathfrak{s}$ . Instead,  $\mathfrak{s}$  is encoded as an *action sequence graph*  $\mathcal{E}^{\mathfrak{s}}$  and then stored in  $c.\pi_0$ . Vattam et al. (2014; 2015) introduce the action sequence graph representation for plans and discuss the reasons for using this representation in SET-PR. Inputs to SET-PR are sequences of actions performed by an observed agent and the resulting states. Like plans in cases, these observed sequences are modeled as action-state sequences and represented as action sequence graphs. Input graphs are then used to retrieve matching cases from the case base. SET-PR uses approximate graph matching techniques, described to compare an input against each case’s plan  $c.\pi_0$  and assign a score to  $c$ . The case with the highest score is returned; SET-PR predicts that the agent is following the plan  $c.\pi_0$  to achieve the goal  $c.g_0$ .

We adapted SET-PR, a general-purpose plan recognizer, to fit into the ASM agent architecture. First, to operate in a multi-agent domain, we modified it to recognize a team’s plan (containing individual team members’ actions) and a team’s goal (containing individual team members’ goals). In particular, we modified SET-PR’s plan representation by adding an actor to each action as its first argument. Furthermore,  $c.g_0$  now represents a team goal and contains a set of goal propositions, one per team member. Second, the input to SET-PR is no longer provided by the environment, but is instead provided by the Explanation Generator. Explanations contain the set of all inferred actions and events, from which the subset of team members’ actions are extracted and used as input to SET-PR.

From the Plan Recognizer we obtain the most likely plans and goals of the ASM’s teammates. To illustrate, suppose that the output of the Explanation Generator consists of the

following three observed actions of the ASM's teammates: (follow MEMBER2 MEMBER1), (follow MEMBER3 MEMBER1), and (move MEMBER1 ROUTE1). Using this an input, SET-PR returns the following recognized team goal:

```
MEMBER1 -> (investigate-route MEMBER1 ROUTE1)
MEMBER2 -> (investigate-route MEMBER2 ROUTE1)
MEMBER3 -> (investigate-route MEMBER3 ROUTE1)
```

SET-PR also returns the following recognized plan, which contains the predicted actions of team members:

```
((follow MEMBER2 MEMBER1), (follow MEMBER3 MEMBER1),
 (move MEMBER1 ROUTE1),
 (DIRECT MEMBER1 (DIRECTIVE investigate-checkpoint)),
 (gesture MEMBER1 POINT),
 (move MEMBER2 LOCA), (move MEMBER3 LOCB), ...).
```

#### 4.4 Goal Selector

The Goal Selector determines a goal for the ASM based on the goals of its teammates. If the robot has  $n$  teammates, the goal tuple  $T = (g_1, g_2, \dots, g_n)$  contains the currently recognized goal  $g_i$  of each teammate. Our current implementation uses a static goal (i.e., no goal selection is performed) but we are currently investigating several alternative goal selection strategies.

As future work, we plan to implement a goal selection strategy where the ASM selects its goal based on its teammates' current goals (i.e.,  $goalSelection: \mathcal{T} \rightarrow \mathcal{G}$ , where  $\mathcal{T}$  is the set of all goal tuples and  $\mathcal{G}$  is the set of all goals). However, we also plan to investigate a strategy that examines how the teammates' goals have changed over time. The ASM would use the currently observed goals  $T_t$  at time  $t$  and the sequence of previously observed goals  $\langle T_{t-1}, T_{t-2}, \dots \rangle$  to select a new goal (i.e.,  $goalSelection: \mathcal{T} \times \mathcal{T} \times \dots \times \mathcal{T} \rightarrow \mathcal{G}$ ). For example, the ASM could observe a teammate that historically has *scouting* goals but now has a goal reserved for squad leaders. This would allow the robot to infer that something has happened to the original squad leader and act accordingly (e.g., inform central command).

Goal selection strategies require criteria governing when the ASM's goal should be changed. Our initial plan is to use selection criteria provided by a domain expert. We prefer this in our domain because it avoids introducing additional error into the ASM (e.g., if it attempted to learn goal selection criteria) and prevents switching to an incorrect goal. If the ASM switches to an incorrect goal, it could impact the squad's ability to complete their task or mission. However, we also plan to explore strategies for learning goal change criteria in situations where criteria from an expert are not available or incomplete. The majority of our Goal Selector remains future work.

The selected goal is then sent as input to the Plan Generator, which uses the goal to generate a plan containing the ASM's future actions.

#### 4.5 Plan Generator

Plan Generation takes the intended goal of the ASM, changed or unchanged, and determines the best way of accomplishing it. Whenever the goal is changed during goal selection, a new plan is

generated for the selected goal. Replanning also occurs when an existing plan becomes inadmissible due to unexpected environment changes. In this situation, the generated plan attempts to fulfill the goal of the prior plan. In both cases, new plans are generated by the continuous time HTN planner SHOP2-PDDL+ (Molineaux, Klenk, & Aha, 2010). This enables the ASM to react to both the changing goals of the squad, and unexpected changes to the environment, as guided by task decomposition knowledge.

SHOP2-PDDL+ is ideal for use as the Plan Generator in the ASM domain for several reasons. First, it can represent continuous time and exogenous events that predictably occur after a time, such as a requested air strike or arrival of other agents at a location. Most planners can represent continuous changes to an environment only through durative actions, which don't properly represent how the choices of others affect the world. Second, SHOP2-PDDL+ is fast relative to other continuous-time planners, due to the task decomposition knowledge it uses internally to guide search and eliminate large areas of the search space. This is important due to the need for frequent replanning introduced by goal changes and unexpected environment change. Third, SHOP2-PDDL+ is effective in partially observable domains. The integration of SHOP2-PDDL+ with DiscoverHistory, both of which use the same knowledge representation, has been demonstrated in past studies as an effective means of understanding hidden information and creating plans that rely on it (Molineaux et al., 2012; Wilson et al., 2013; Molineaux & Aha, 2014). To support this, agent beliefs about hidden states generated by the explanation system are combined with observations from the environment. Together, these are sent to the Plan Generator as the initial state.

The plan generated by the Plan Generator is the decision output by the decision cycle, and replaces any prior plan generated in a previous cycle. Plans are enacted by sending actions to the environment at appropriate intervals. Enacting a plan fulfills the goal found in goal selection, which in turn relies on the environment interpretation generated by the first three components in the cycle. This entire cycle is continuously repeated, generating and enacting new plans as necessary, for as long as the agent inhabits its environment.

## 5. Integration Proof of Concept

In this section, we describe our proof-of-concept system, focusing on our demonstration scenario and a discussion of the system's performance.

### 5.1 Integration Scenario

To demonstrate the effectiveness of our GR agent, we created a scenario that encompasses both the standard mission-based objective that our domain normally contains and a characteristic anomalous event that disrupts mission execution.

The ASM accompanies a squad on a typical reconnaissance mission. Figure 2 shows an example mission map with checkpoints and routes labeled. A squad performs this mission by moving between checkpoints on a pre-specified route (e.g., *Start Area*, *CP Alpha*, *CP Bravo*). At each checkpoint, the squad splits into smaller teams or *sub-squads* that perform investigations of nearby areas, each of which follows a pre-determined investigation route (e.g., *RA1*, *RA2*, *RBI*,

*RB2*). The mission is complete when the final checkpoint on the main route has been investigated. Our scripted scenario occurs as follows:

1. Squad receives a reconnaissance mission
2. Squad moves to checkpoint *Alpha*
3. Squad members investigate routes *RA1* and *RA2*, and then return to *Alpha*
4. Squad begins moving to checkpoint *Bravo*
5. Squad encounters an enemy sniper
6. Squad members eliminate the enemy sniper
7. If casualties have occurred, mission is aborted
8. Squad continues on to checkpoint *Bravo*
9. Squad members investigate routes *RB1* and *RB2*, and then return to *Bravo*
10. Mission complete

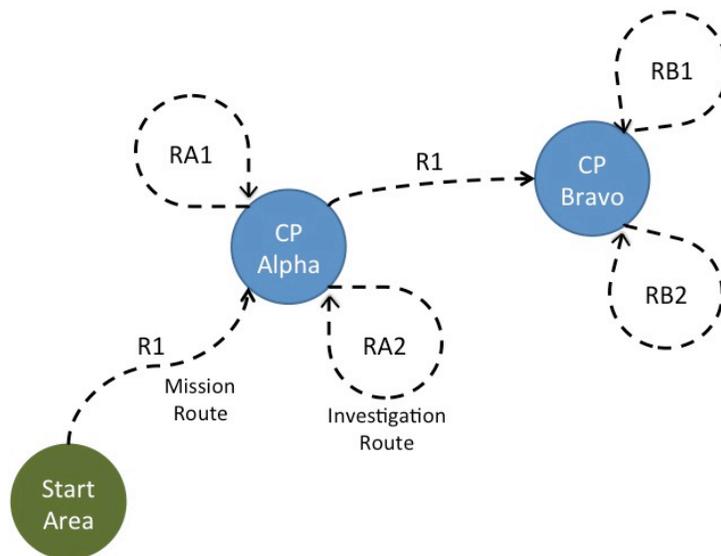


Figure 2. Reconnaissance Scenario.

The sniper encounter is an unplanned event that causes the squad to abandon their existing goals in favor of new ones, such as running for cover and attempting to locate the enemy. The autonomous agent must recognize these changes based on observations and natural language and act accordingly. Further, a member of the team will announce that the enemy has been neutralized, which the agent must recognize and use to update its understanding of the situation. Finally, the squad members continue on their mission and complete it, resuming their previously interrupted goal.

## 5.2 Demonstration Walkthrough

Below we provide a walkthrough of our demonstration in the form of data snapshots (for one cycle through the GR process) as we progress through the scenario. Explanation generation was requested to generate two explanations at each opportunity.

The initial goals of the team members are given as follows:

```
Goal, member1: (investigate-route route-1)
Goal, member2: (investigate-route route-1)
Goal, member3: (investigate-route route-1)
```

At this point, `member1` and `member3` are at checkpoint alpha and `member2` is about to investigate `route-alpha1`.

The Natural Language Interpreter receives the following utterance as input:

```
Utterance: "Man down" Time: 600.013 Speaker: member3
```

...and generates the following interpretation as output:

```
...
(is-person ?teammate)
(on-team ?teammate ?myTeam)
(person-has-property ?teammate injured)
(is-utterance ?utterance)
((utterance-text ?utterance) "Man down")
((utterance-type ?utterance) statement)
```

This interpretation tells us that there is a person `?teammate` on the robot's team `?myTeam` that is injured, along with some information about the utterance `?utterance`. It is then passed along as input to the Explanation Generator.

Before the utterance was made, the Explanation Generator had the following explanation about the environment:

```
(observe-environment s=392)
(move route-alpha1 location-alpha member2 s=393)
(gps-observe-location member2 location-alpha s=394)
(observe-environment s=415)
(shoot-toward location-alpha #v795 s=416)
(observe-environment s=438)
```

In this explanation, it is believed that teammate `member2` was starting to move along `route-alpha1` when an unknown person `#v795` fired a shot towards his/her location.

Adding the utterance interpretation from the Natural Language Interpreter (along with other observed information) to this existing explanation results in the following inconsistencies:

```
inconsistency
  condition: person-has-property (#v958 injured)
  prior: <none>
  next: (observe-environment s=461)
inconsistency
  condition: utterance ("man down" 600.013 member3)
  prior: (observe-environment s=438)
  next: (observe-environment s=461)
inconsistency
  condition: sound-occurs (600.013 speech location-alpha)
  prior: (observe-environment s=438)
  next: (observe-environment s=461)
```

The inconsistencies indicate that the new observations include information not explained by prior events:

- A previously unknown person #v958 was injured
- member3 uttered a sentence
- A sound occurred at location-alpha

It also generates the following ambiguities:

```
ambiguity
  condition on-team (#v958 #v957)
  prior: (observe-environment s=438)
  next: (observe-environment s=461)
ambiguity
  condition: on-team (robot1 #v957)
  prior: (observe-environment s=438)
  next: (observe-environment s=461)
```

The ambiguities indicate that new observations include information not bound to prior knowledge:

- A previously unknown person #v958 is on some team #v957
- The robot is on team #v957

The Explanation Generator attempts to resolve these inconsistencies and ambiguities by deriving two possible explanations (new information is shown in bold).

**Explanation 1**

This first explanation indicates that a previously unknown person at `location-4` was shot.

```
(assume-initial-value (object-location #v958) location-4 s=1)
...
(observe-environment s=392)
(move inv-routea1 loca member2 s=393)
(gps-observe-location member2 inv-routea1 s=394)
(observe-environment s=415)
(shoot-toward loca #v795 s=416)
(person-clipped #v958 s=(interval :start 417 :end 460))
(observe-environment s=438)
(speak-aloud "man down" member3 s=439)
(human-hears member3 "man down" s=440)
(human-hears member1 "man down" s=440)
```

**Explanation 2**

The second explanation indicates that someone was injured earlier, and `member3` is commenting on it now.

```
(assume-initial-value (person-has-property #v958 injured) s=1)
...
(observe-environment s=392)
(move inv-routea1 loca member2 s=393)
(gps-observe-location member2 inv-routea1 s=394)
(observe-environment s=415)
(shoot-toward loca #v795 s=416)
(observe-environment s=438)
(speak-aloud "man down" member3 s=439)
(human-hears member3 "man down" s=440)
(human-hears member1 "man down" s=440)
```

The Explanation Generator maintains these plausible explanations, and a single most-plausible explanation is given as input to the Plan Recognizer.

At this time, the goals of `member1` and `member3` have changed. However, the Plan Recognizer is unable to immediately detect these changes. Further observations lead to refinements of these explanations. Subsequently, the Plan Recognizer is provided with the following history of actions and events:

```
...
(speak-aloud "man down" member3 s=439)
(human-hears member3 "man down" s=440)
(human-hears member1 "man down" s=440)
(assume-defensive-position tree1 member3 s=462)
```

...and is able to recognize the following changed goals:

```
Goal change, member1: (respond-to-sniper loca4)
Goal change, member3: (respond-to-sniper loca4)
Goal detected, enemy1: (snipe team1)
```

These statements reflect a correct recognition, namely that an enemy is sniping the team, and two team members are now responding to the sniper instead of their previous goal (i.e., to investigate the route).

## 6. Conclusions and Future Work

We have demonstrated how a goal reasoning agent, the autonomous squad member (ASM), cooperates with teammates during a squad mission. The ASM recognizes and reacts to unexpected and anomalous events, processes natural language, infers and assumes unobserved facts about the world (via explanation generation), and recognizes the plans of its own teammates. In future work, goal selection will allow the agent to choose collaborative goals that are appropriate for its changing circumstances. Plan generation will then determine a means to accomplish these selected goals.

Our demonstration scenario suggests that the integration of these components is effective. In future work, we will conduct a formal empirical study to test this claim. These experiments will be distinguished into two categories, integration testing and individual component testing. In these experiments, we will measure the precision and recall of natural language interpretation, plan recognition, and explanation generation, as well as mission success criteria for determining overall performance. Integration testing experiments will measure whether the ASM performs best with all components present. For example, in an ablation study we will compare the precision and recall of explanation generation and plan recognition with and without natural language interpretation. Our objective is to show that the ASM performs better as a whole than any subset of its components. The other category, individual component testing, will test whether our implementation for each component is a good choice for the overall system. For example, we will swap out our Explanation Generator for a simpler deductive reasoner and compare the performance of the overall system using each.

## Acknowledgements

We would like to thank OSD ASD (R&E) for sponsoring this research, which is being conducted in a project led by the US Army Research Laboratory (ARL).

## References

ARL (2011). ARL Robotics Collaborative Technology Alliance (RCTA): FY 2011 Annual Program Plan. [arl.army.mil]

- Bird, S. (2006). NLTK: The natural language toolkit. *Proceedings of the COLING/ACL on Interactive Presentation Sessions* (pp. 69-72). Stroudsburg, PA, USA: Association for Computational Linguistics.
- Cantrell, R., Talamadupula, K., Schermerhorn, P.W., Benton, J., Kambhampati, S., Scheutz, M. (2012). Tell me when and why to do it!: run-time planner model updates via natural language instruction. *International Conference on Human-Robot Interaction*. Boston, MA, USA. ACM.
- Coddington, A.M., Fox, M., Gough, J., Long., D., & Serina, I. (2005). MADbot: A motivated and goal directed robot. *Proceedings of the Twentieth National Conference on Artificial Intelligence* (pp. 1680-1681). Pittsburgh, PA, USA: AAAI Press.
- Grastien, A., Haslum, P., & Thiébaux, S. (2011). Exhaustive diagnosis of discrete event systems through exploration of the hypothesis space. *Proceedings of the Twenty-Second International Workshop on Principles of Diagnosis* (pp. 60-67). Murnau, Germany.
- Gspandl, S., Pill, I., Reip, M., Steinbauer, G., & Ferrein, A. (2011). Belief management for high-level robot programs. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence* (pp. 900-905). Barcelona, Spain: AAAI Press.
- Kott, N.J. III, Welfare, M., van Lierop, T.K., & Mottern, E. (2010). Safe operations of unmanned systems for reconnaissance complex environments Army technology objective. *Proceedings of SPIE 8045, Unmanned Technology XIII*. SPIE.
- Molineaux, M., Kuter, U., & Klenk, M. (2012). DiscoverHistory: Understanding the past in planning and execution. *Proceedings of the Eleventh International Conference on Autonomous Agents and Multi-Agent Systems* (pp. 989-996). Valencia, Spain: International Foundation for AAMAS.
- Molineaux, M., Klenk, M., Aha, D.W. (2010). Planning in dynamic environments: Extending HTNs with nonlinear continuous effects. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. Atlanta, GA, USA: AAAI Press.
- Molineaux, M., & Aha, D.W. (2014). Learning unknown event models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. Quebec City (Quebec), Canada: AAAI Press.
- Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., & Teneketzis, D. (1995). Diagnosability of discrete-event systems. In *IEEE Transactions on Automatic Control*.
- Sohrabi, S., Baier, J., & McIlraith, S. (2010). Diagnosis as planning revisited. *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning* (pp. 26-36). Toronto (Ontario), Canada: AAAI Press.
- Talamadupula, K., Schermerhorn, P., Benton, J., Kambhampati, S., & Scheutz, M. (2011). Planning for agents with changing goals. *Twenty-First International Conference on Automated Planning and Scheduling: Proceedings of the System Demonstrations* (pp. 71-74). Feiburg, Germany: AAAI Press.
- Vattam, S.S., Aha, D.W., & Floyd, M. (2014). Case-based plan recognition using action sequence graphs. *Proceedings of the Twenty-Second International Conference on Case-Based Reasoning* (pp. 495-510). Cork, Ireland: Springer.
- Vattam, S.S., Aha, D.W., & Floyd, M.W. (2015). Error tolerant plan recognition: An empirical investigation. In *Proceedings of the Twenty-Eighth Florida Artificial Intelligence Research Society Conference*. Hollywood, FL, USA: AAAI Press.

Wilson, M., Molineaux, M., & Aha, D.W. (2013). Domain-independent heuristics for goal formulation. In *Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference*. St. Pete Beach, FL, USA: AAAI Press.