Deep Green Seedling Language for Investigating Myriad Eventualities (LIME)

FINAL REPORT

Build date: 13:46 November 28, 2007

Approved for Public Release, Distribution Unlimited

Robert P. Goldman (SIFT)
Kyle S. Nelson (Adventium Labs)
David J. Musliner (Honeywell)
Mark S. Boddy (Adventium Labs)

Acknowledgment: This report is based upon work funded by DARPA, administered by the United States Air Force under Contract No. FA8750-05-C-0089. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or the United States Air Force.

TABLE OF CONTENTS

Chapter 1

Introduction

The Language for Investigating Myriad Eventualities (LIME) is intended to serve as a domain modeling and inter-module communication language within Deep Green. Given the representational requirements of DG, LIME needs to be able to encompass uncertain action outcomes, limited information and contingent planning and execution, reasoning about resources, state abstraction, trajectory constraints, temporally extended and overlapping actions, adversary planning and action, dynamically created and destroyed objects and resources, and task decomposition, among other things.

The goal of the seedling is to provide a head start to the Deep Green program, by defining language requirements and a set of design recommendations. This report describes the current status of those design recommendations. To the extent possible, the recommendations are implementation-neutral, because there is more than one way to approach these problems and, more pragmatically, the DG program has not yet started, so exactly how the software gets implemented depends on who gets the contract.

This report consists of three main chapters:

1. Military scenarios and use cases, covering a range of situations relevant to the Deep Green program.

2. LIME language requirements derived from Deep Green system requirements (specifically from the BAA) and the use cases.

3. The LIME design document, describing the current approach to LIME and some accompanying rationale.

Chapter 2

Deep Green Scenarios and Use Cases

2.1 Scenario Overview

The following scenarios were developed to provide a framework within which requirements and use cases for Deep Green's Language for Investigating Myriad Eventualities (LIME) can be created. The three scenarios — mid-intensity combat, counterinsurgency, and humanitarian missions — incorporate a broad and representative sample of aspects of contemporary full-spectrum warfare. Figure 2.1 gives an overview of the scenarios and their key aspects. Each scenario is from the DoD point of view (i.e. dealing with those aspects involving the military), and is focused at Brigade or Battalion level commanders. The three scenarios may not cover every possible situation, but they are sufficiently distinct to demonstrate the major DG concepts and the applicability of these concepts to operations in general.

Each scenario involves one or more US Army, Marine, and/or Coalition land units. By focusing on land units we achieve a similarity across the scenarios, which lets us minimize the number of terms required, and makes it easier to highlight the planning advantages of DG while avoiding the details of the differences in tasks, tactics and procedures (TTPs) of the different services.

2.2 Mid Intensity Conflict (MIC) Scenario and Use Cases

The Mid Intensity Conflict (MIC) scenario shown in Figure 2.2 is adapted from Schmitt's tactics workbook [18]. It is somewhat academic because a) this scenario reflects the kind of enemy forces (mounted and armored forces in open terrain) the US military is well suited to take out and b) DG will not have the benefit of knowing what the enemy is planning. This situation was selected to provide a straight-forward example to focus on DG and LIME and the enemy analysis provides guidance as to the role of the enemy COA. A recommendation for future development is to develop a MIC scenario with more than two combatants (*e.g.*, a coalition of friendly vs. a coalition of enemy) where the sides have differing and asymmetric allegiances, tactics, rules of engagement, and so on. Asymmetry is captured in the other scenarios, but they do not include asymmetric, MIC-level combatants which may more

Scenario	Summary	Terrain	Participant Affiliations	Commander COA Options	Features
Mid Intensity Combat (MIC)	Defeat/Delay Enemy River crossing, 3 bridges	Rural	2-combatants Symmetric	O(10)	Assets outside Commander's scope
Humanitarian Aid (Binni)	Sample Collection Protect Convoy Humanitarian Aid	Lightly Populated	6 groups Asymmetric	O(100)	Mixture of participant types
Counter Insurgency (COIN)	Protect infrastructure Attack insurgent target Support HN, NGOs Deterrent patrols	Urban	8 groups Asymmetric and fluid	O(1000)	"Three Block War" NGO/Diplomatic Intervention

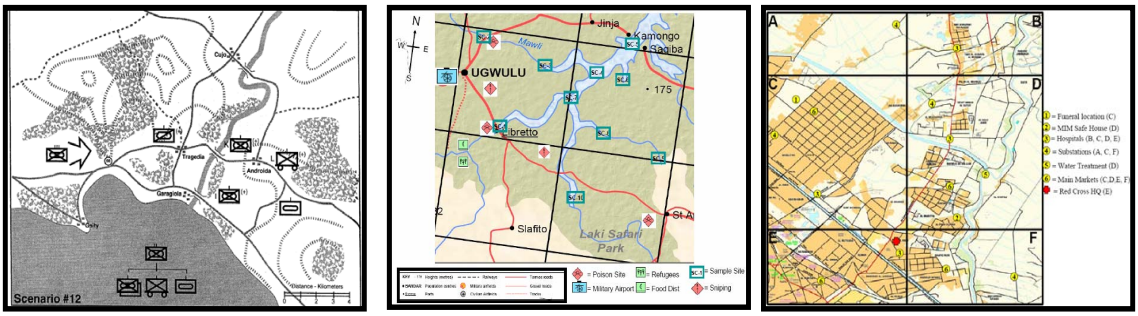

Figure 2.1: Overview of the scenarios used for LIME

accurately reflect potential future Mid Intensity Conflict missions.

2.2.1 Mid Intensity Conflict (MIC) Scenario Description

Mission: Defeat any enemy forces in your zone, or failing that, at a minimum delay and disrupt the enemy advance.

Force Structure: 3d Battalion, 4th Marines consisting of:

- Companies I and K on assault amphibious vehicles (AAVs) located on the Garagiola River.
- Company L on trucks at Androida.
- A light armored reconnaissance (LAR) company west of the river.
- A tank company in reserve east of Androida.
- Anti-tank (TOW) section is moving with Company L.

Operational Situation: Operations have been ongoing for two weeks, facing superior enemy mechanized forces. US has been forced to fall back from west to east, while slowly wearing down the enemy and conserving own combat power. Other notes:

- The Battalion is currently halted along the river.
- LAR company is moving west to reestablish contact through Tragedia.

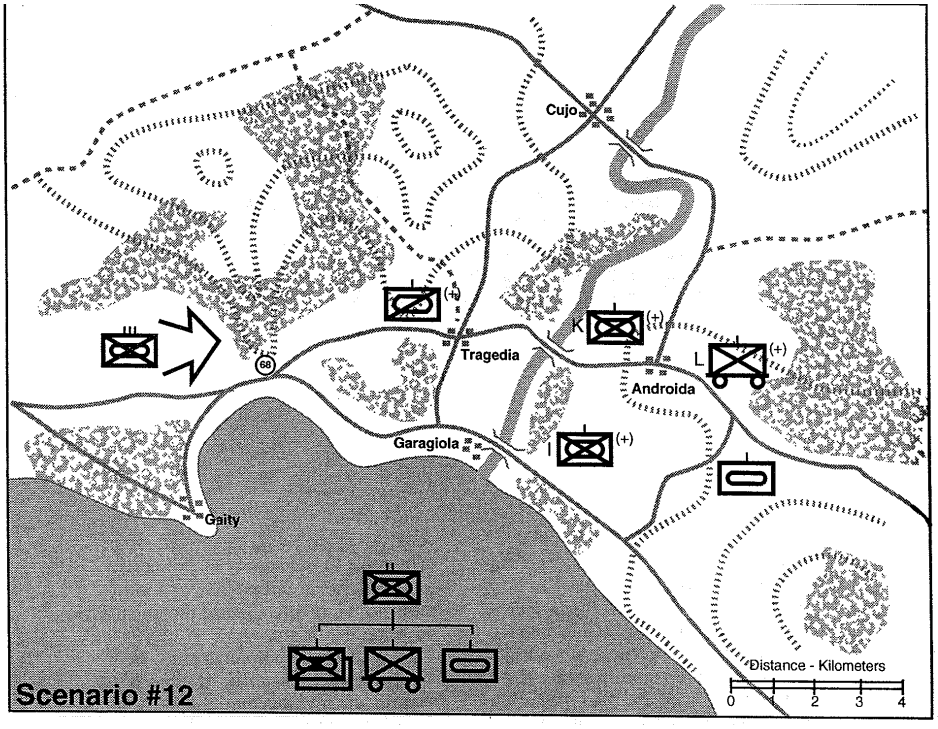

Figure 2.2: Operational map from the friendly point of view.

- Artillery liaison officer assures massed fire support.
- Air officer says he can get you "a couple of sorties of FA-18s."
- Three bridges cross the river: near Cujo, between Tragedia and Androida, and near Garagiola.
- Poor visibility due to heavy fog lately has obscured combatants from each other.
- Garagiola River has steep banks that make it impassable to any vehicles.
- Centrally located Androida is key for several reasons: good observation of nearly the entire battlefield, Androida and the connected high ground curve south to the shore to dominate both southern bridges, and Androida blocks the main route from the Cujo bridge.
- The area immediately east of Checkpoint 68 is a low area devoid of much cover and concealment, penned in and dominated by high ground on three sides and by water on the fourth ("a tailor-made killing zone").
- Somewhere along 4 or 5 kilometers of flatland east of the river mouth there are suitable landing beaches. Further southeast there are not: the high ground appears to descend directly to the waterline.

Intel Situation:

- Learned that the enemy force is a mechanized regiment consisting of a tank battalion and two mechanized infantry battalions.
- Enemy tactics have emphasized attacking aggressively upon contact and maintaining the momentum.
- The enemy will generally try to overrun resistance with tanks, and prefers to dismount infantry only when necessary for close combat.
- LAR company (moving westward) reports a sizable enemy mechanized force to the west approaching Checkpoint 68, and estimates an enemy regiment with a tank battalion in the lead. The LAR commander does not believe he has been detected by the enemy.
- Enemy tanks will be approaching Tragedia or Garagiola in about 30 minutes.
- Believe that the enemy's objective is to force a crossing of the Garagiola River, and facilitate the continuation of their advance.

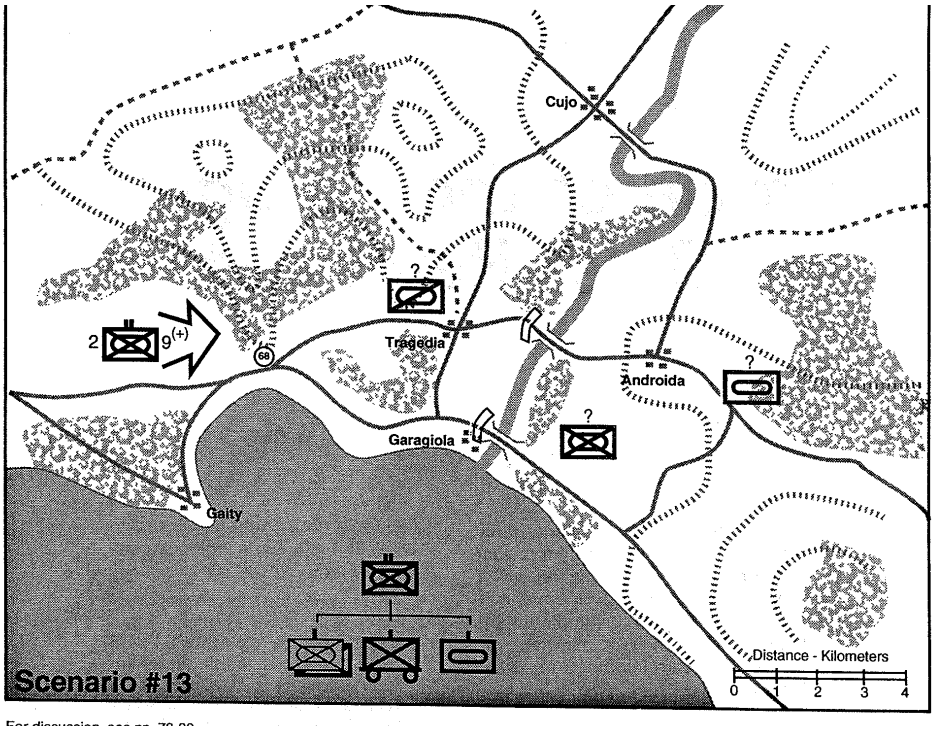

Figure 2.3: Operational map from the enemy point of view.

Enemy Situation: In this scenario, we have the luxury of knowing the enemy situation because Schmitt spells it out in some detail [18]. In reality, we will not know with certainty the enemy force structure or their plans/intentions. Therefore, the following discussion is included to point out some of the challenges that DG will face in accurately accounting for

potential enemy COAs. In fact, it will be very important that DG allow for a wider range of enemy action than may be assumed by the Commander's intel officer (S2).

Schmitt describes the enemy situation as the following:

- **Enemy Mission:** Attack aggressively, force a crossing of the Garagiola River as quickly as possible to facilitate the continuation of the advance, and defeat the US force. The regimental mission (i.e. the superior echelon) is to quickly create a secure bridgehead and an open road to the east.

- **Enemy Force Structure:** 2d Battalion has two companies (E and F) on assault amphibious vehicles (AAVs). Company G on trucks and an antitank (TOW) section on HMMWVs. The Dragons and heavy machine guns from Weapons Company are attached to the companies. 2d Battalion has been moved into the lead and has an attached tank company. Two battalions are held in reserve, the 1st and the Tank Battalion.

- **Enemy Intel Situation:** Believe that the US forces are a tank-mechanized force of battalion strength, they have fallen back through Tragedia and Garagiola, and are preparing positions along the Garagiola River. US tanks have been spotted near Androida and have received a report of US armored reconnaissance (with antitank missiles) at Tragedia. Believe US intends to defend the river line. See Figure 2.3 for the map from enemy point of view.

Two key points of this enemy situation are that:

1. The enemy has in fact spotted the LAR company, despite the LAR commander's opinion to the contrary.

2. Neither side fully understands the capabilities and placement of the other side's forces.

Neutral/Political/Civilian Situation: This scenario is limited to force-on-force, so these factors are not applicable.

Potential Courses of Action and options: Recall the commander's intent: we are to defeat any enemy forces in our zone, or failing that, at a minimum delay and disrupt the enemy advance. The disjunction suggests two fundamentally different courses of action: one prudent and one bold. Each can accomplish the commander's stated intent.

- **Positional Defense (conservative).** The intent of this COA is to keep the enemy on the far side as long as possible, and make it as costly as possible in terms of time and resources for the enemy to cross. Tactics include establishing strong defensive positions along the key terrain, creating a defensive barrier intended to hold up the enemy's advance, and utilizing the river as an obstacle to increase combat power. In this defense we keep some reserves to eject enemy penetrations and restore threatened areas. One option for this defense is to demolish the bridges, but for this option we would need the superior commander's permission.

The risk in this conservative defense COA is that we are surrendering the initiative, and waiting for the enemy rather than actively seeking to destroy them. This raises the possibility that the enemy will be able to find a bold COA by which they may quickly cross the river, and continue their advance.

- **Mobile Defense/Counterstroke (bold).** The intent of this COA is not merely to delay and wear down the enemy, but to destroy him altogether. Tactics include using minimal fixing forces to the enemy's front that would fall back under pressure. Coaxing enemy into committing himself to the southern bridges and drawing the leading enemy elements across to split his forces. In parallel, the bulk of the friendly force, including all tanks, outflank the enemy via the Cujo bridge and attack the enemy in the flank from the north. This approach ignores the natural defensive qualities that the river offers, using it as an obstacle instead. In addition, includes a requirement to have forces outside the control of the current commander, e.g., more air support and additional reserves (tank-infantry battalion).

  One risk of this COA is that executing a flanking maneuver in the field can be difficult, but this also makes it less expected. A larger risk of this COA is the consequences of failure: in the worst case of the conservative COA, the enemy will still cross the river and continue to advance, but only after fighting through all of our forces which remain on the east bank of the river. Some failures of the bold COA — unexpected delays of a flanking force, insufficiently delaying the enemy from reaching the bridge, or simply an outright defeat which we would have brought about more quickly — will allow the enemy to cross the bridge to encounter a far smaller force than the conservative COA would present. In other words, the failure of the bold COA could fall much further short of the Commander's intent to at a minimum delay and disrupt the enemy advance than would the failure of the conservative COA.

Schmitt gives this example of a counterstroke COA [18]:

- Intent: Use two companies to draw the enemy across the river and crush him with a flanking attack from the north, supported by concentrated artillery and air support. This initial focus of our efforts is on fixing the enemy in the killing zone, shifting to the flank attack as it commences.

- LAR company: Delay the enemy along Tragedia-Androida line.

- India Company: Delay the enemy along coast road, from as far forward as possible to a commanding position along the heights east of the river.

- LAR and India: Hurt the enemy, but not so badly they try a flanking maneuver. Keep the enemy's orientation eastward. Must hold for at least 90 minutes.

- Executive Officer (XO): Take Tanks, Kilo and Lima across Cujo bridge and prepare for a coordinated attack on order generally south along Cujo-Tragedia line. Must be ready

in 90 minutes, but sooner is better. Form a tank-infantry company in reserve, and wait for order on whether to commit. Should be ready to attack along either Cujo-Tragedia or Cujo-Androida lines.

- Fire Support Coordinator: Mass fires on-call in the vicinity of Tragedia, Androida and the Tragedia and Garagiola bridges.

- Air: A couple of sorties is not enough: we will need everything that flies. This COA requires that we mass all supporting arms in conjunction with the flanking attack.

- Regimental Commander: Request additional air support, plus the tank-infantry battalion being held in reserve at that level.

2.2.2 Mid Intensity Conflict: Pre-Battle Preparations Use Case

This scenario allows us to examine how DEEP GREEN assists with preparing for a battle. One might imagine that DG could lead to paralysis: the Commander second-guesses himself over and over as DG reveals how every COA, faced with a sufficiently skillful enemy COA, will lead to disaster. Especially considering the enemy's superiority in forces in this particular scenario, for every friendly COA there certainly would exist a disastrous countermove. To the contrary, the intended use of DEEP GREEN allows the Commander to propose both the conservative and aggressive COAs with confidence because DG provides the ability to recover from enemy COA choices which might otherwise thwart the Commander's intent. In this scenario, we have this DG use case:

- The Commander decides to explore two COAs: the conservative defense in depth, and the bold counterstroke.

- The Commander's staff proposes a number of possible enemy COAs.

- With the aid of DG, a Futures Graph is built to model how the scenario might evolve.

- In some of these situations, there are catastrophic outcomes; in others, wildly successful outcomes; in still others, more mixed outcomes.

- The Commander then explores the Futures Graph to reveal *branching points* separating the bad and good outcomes. Factors behind branches to catastrophic outcomes may be detectable before the branching point, allowing the bad branch to be avoided. Likewise, examination of branches to good outcomes reveal the key factors behind those successes.

In the following sections we examine these COAs in more detail, and discuss how this scenario has influenced LIME's design. Note that in this scenario, engagement with the enemy has been occurring for some time, so DG will have access to previous COAs and their outcomes. Seeding DG with such contextual information will be important to its effectiveness.

Positional Defense COA

In this COA the Commander's intent is to delay the enemy's advance across the river, he has about 30 minutes to complete defenses.

- Believe the enemy will attack via Tragedia and Cujo.
- K Company prepares fighting positions at Tragedia Bridge.
- I Company prepares fighting positions at Garagiola Bridge.
- L Company takes up defensive positions East of Cujo bridge, reinforcing LAR company.
- LAR Company falls back to Cujo, and joins with Company L.
- Tank reserve takes up a position midway between Cujo and Androida.
- Request air sorties to patrol Northern trail, and strike the enemy at checkpoint 68.

DG iterates with the Commander to determine the constraints and timing between the various entities. For example, should LAR fall back when Company L is at Cujo, or before? What should it do the meantime? What coordination is required to prevent L from firing on LAR company? Interactions must be sufficiently detailed to capture Brigade or Battalion planning issues.

Once specified by Commander's Associate, the COAs are supplied to Crystal Ball. Crystal Ball first augments these COAs with other information about the entities, such as current knowledge of weapons, fuel, etc., including uncertainties. Crystal Ball then provides Blitzkrieg with the starting state(s), plus guiding parameters such as how long Blitzkrieg has to generate futures. Note that the starting state could be the best assessment of the current state(s), or could represent states at a future time: for example, we might start at the future states one hour from now. Based on the knowledge within Crystal Ball, Blitzkrieg is guided to consider at least the following contingencies:

- Whether the heavy fog clears, impacting movement and air strike effectiveness.
- Whether LAR Company has been spotted, impacting the factor of the enemy's surprise.
- Whether LAR Company moves directly to Cujo, or takes longer. In the latter case, LAR Company may engage the enemy while moving to Cujo — these engagement may turn out to be a good or a bad thing, an assessment which DG can help the Commander to make.
- Whether we are allowed the requested air sorties, or fewer.
- Variations of enemy strength and reserve options, and of a shift of emphasis between Cujo and Tragedia.

One issue for the DG program to address is the extent to which the input COA from the Commander includes options to consider as opposed to deriving them from the Automated Option Generator and/or directly from the Futures Graph (for example based on uncertainty recorded with the items in the starting state(s)), or possibly a mixture of both.

Aggressive Counterstroke COA

DG will also enable the Commander to explore the aggressive option. Here the Commander's intent is to destroy the enemy.

- LAR Company executes a delay along Tragedia-Androida line.
- I Company executes a delay along the coast road from as far forward as possible.
- LAR and I delay until Tank reserve, Kilo and Lima are in position, which must take no more than 90 minutes. They must use caution to prevent the enemy from trying a flanking maneuver.
- Tank reserve, K and L Companies cross the Cujo bridge, and prepare for a coordinated attack along the line from Cujo to Tragedia, or from Cujo to Androida, pending the Commander's order.
- Form a tank-infantry company in reserve, waiting for Commander's order before committing.
- The fire support commander mass fires on-call in vicinity of Tragedia, Garagiola, and Androida.
- The air controller requests as much air support as possible.
- Request that the Regimental Commander commit additional tank-infantry battalion now being held in reserve.

DG will iterate with the Commander to clarify the COA: determining what is meant by *as much air support as possible*, the desired structure for the reserve unit and corresponding changes to the units from which it is derived, coordination details between the companies, what the Commander would do with the extra battalion if his request is approved. The Commander's staff draws up several enemy COAs for crossing at Cujo and Tragedia, including at least one that anticipates the flanking maneuver the Commander is planning.

Once specified and detailed by Commander's Associate, the COAs are supplied to Crystal Ball. Crystal Ball performs the same processing to this COA as for the conservative plan: it augments the COA with additional entity information, and provides Blitzkrieg with guiding parameters. Here, the contingencies which the knowledge within Crystal Ball guides Blitzkrieg to consider include:

- Variations in the air support provided. Analysis of the Futures Graph could identify the minimum sorties required to achieve a certain likelihood of success. DG will present this as a decision point for the Commander, expecting him to confirm that the support can or cannot be provided.

- Whether Tank reserve, K and L attack along Cujo/Tragedia line, along the Cujo/Androida line, or along both, along with different timing options for each.
- Whether LAR and I Companies delay the enemy for more or less time than expected.
- Whether Tank reserve, K and L are delayed in reaching the jumping-off point for their attack.
- Whether the Commander receives the additional tank-infantry battalion.

One question which the DG program must resolve is how information about the newly requested tank-infantry battalion would be provided to DG. If there is a higher-echelon DG in operation, then the information may already be available to the lower-echelon DG. Otherwise, the lower-echelon DG must be able to retrieve it from somewhere — either from a larger database, or directly from the Commander. Similar situations occur when a defined COA references forces or other objects not already known to DG. In some cases the unknown entity will simply be an error, either on the Commander's part or on the handwriting recognizer. In other cases the unknown entity will be completely new, like a newly defined coordination point or key terrain.

Mid Intensity Conflict: Courses of Action Analysis

Given possible COAs — here, conservative or bold — it is the Commander's responsibility to determine the one which our forces will follow. With or without DG, the Commander's assessment must include several considerations:

- The mission itself.
- The terrain.
- The enemy situation.
- The capabilities of our own forces.
- The fit with the superior Commander's overall strategy.
- The success of the various preparations, which in this scenario include:
  - Fire support planning for the coordination and massing of air and ground fires.
  - Operations by combat engineers to prepare defensive works, obstacles, and minefields to canalize and temporarily halt enemy forces to make them vulnerable to fire.
  - Coordinating the barrier plan with the fire plan.
  - Selecting withdrawal routes for fixing forces and directions of attack for flanking forces.
  - Mission rehearsals (at least with key leaders) to get the timing and coordination down.

What DG will bring to this decision process is the ability to analyze well-developed futures to determine the best COA, or to devise a hybrid of good COAs. In this scenario, aspects of the COAs that DG can help the Commander to discover include:

- *Flexibility.* Are there options within the aggressive COA which also provide sufficient flexibility to revert to a variant of the conservative COA options? Alternatively, can the conservative COA be executed in a way which allows a quick shift to the aggressive posture in reaction to developments which give the aggressive COA a more acceptable chance of success?

- *Low payoff.* The risks of the bold COA are higher than the those of the conservative COA, and the Commander may not consider the odds of success of the bold COA to be high enough to justify this risk. DG can provide insight into which of the aggressive options show such an inadequate chance of success.

- *Make-or-break points.* Similarly, DG can help identify go/no-go decision points (such as adequate air support) for different parts of the plan, where an external event will limit the Commander's subsequent options.

LIME Implications

The MIC scenario points out a collection of basic items that will need to be represented in LIME in order to provide the type of analysis required:

- Friendly and enemy forces, their capabilities (including equipment, weapons and skills), and our uncertainty of our knowledge of them (especially of the opposing forces). LIME must be able to express reconfiguring them on the fly, for example forming a tank-infantry company in reserve.

- The terrain. Some aspects of the terrain correspond to actual physical structures, such as the bridges. Other aspects such as zones, areas, lines and routes are conceptual; others such as Checkpoint-68, the Androida-Tragedia line, "the killing zone" and "the heights" may be ephemeral.

- Environmental features: the steep river banks, the lake, etc.

- Weather features and their potential variations, including durations and other constraints. This scenario included heavy fog of uncertain expected duration.

- Means to refer to objects across all DG components. For example, Checkpoint-68 needs to mean the same thing across DG. Furthermore, the key terrain will need to be explicitly named — for example, each of the bridges will need to have an explicit name like "Cujo," and DG must associate this name with the correct terrain feature and its properties.

- Past actions and their effects on the current situation need to be represented. In this scenario we know that the enemy battalion strength was reduced in the last encounter,

we may have made certain defensive preparations, we may have rehearsed or otherwise prepared for certain maneuvers, and we may be aware of past tendencies of participants (here we consider only the enemy, but in other scenarios we will also consider neutrals, for example).

- A COA may set forth explicit constraints on actions within it. In this scenario we noted actions happening "on-order," and actions delayed for 90 minutes. In fact, this latter constraint is actually a constraint between two actions; the underlying issue was to allow the other units to get into place.

- Implicit (or operational/doctrinal) constraints arise as a consequence of explicity directed actions. Here we directed forces to move across the Cujo bridge, but this motion is not instantaneous; DG must infer the constraint of the time these actions require.

- A common terminology needs to be established for the actions that the various units can take, for example, movements, seizing, preparing attack positions, coordinating, and so on. Furthermore, the Commander terms will need to be fleshed out into more detail to conduct the planning. For example, preparing fighting positions in a particular location requires that first the unit move to that location.

- Representational differences from the earlier COA include more constraints between the units, a heavy dependence on adequate air support, requesting forces (the tank-infantry battalion) not currently under the control of the Commander, and the judgment call on when to issue the flanking attack order (various timing options could be explored with DG).

The MIC scenario also highlights some things that do *not* need to be represented in LIME.

- Details of how different objects behave can be limited. For example, knowledge that a tank operates only on land while an amphibious vehicle can operate on land or water is important, but need not be explicitly modeled in the language. It needs to be considered by the Commander, to some extent in Blitzkrieg (to correctly unroll futures), the Automated Option Generator (to generate reasonable options), and OneSAF. It would probably be a mistake, however, to assume exactly the same dynamics in each of these components. Otherwise, DG might have a very shallow analysis.

- Fairly static items like cities and roads and such, even if they are tactically important. To the extent the are referenced in a COA, these items would be indicated using a separate symbology — e.g. using Cujo as a rendezvous point (rather than it being explicitly represented as a city). Likewise, roads need not be explicitly representable. Blitzkrieg will need to have an internal model that matches the map the Commander is using, but LIME need not include that sort of detail.

2.2.3 Mid Intensity Conflict Enemy COA Options and Analysis

In addition to friendly COAs, the Commander's staff (S2) draws up several enemy COAs. In this case, there is a richer set of Enemy COAs than there are for Friendly forces and the success of the Friendly force COAs depends greatly on which bridges the Enemy targets. The first enemy COA would be for crossing at Cujo and Tragedia because this is where the Commander *thinks* the enemy will attack. However, with support from DG other COAs should also be included — perhaps COAs developed by the Automated Option Generator. These include the following (note that we refer to the bridges by initials: C for Cujo, and so forth):

- Attack C: Avoids the natural killing zone and possibly takes our main positions in the flank, because the Cujo bridge is somewhat isolated from the other two and beyond the range of US mutual support. Can move a unit along the trail north of the ridge to the Cujo road, it is the only move masked from US observation from Androida.

- Attack G: The Garagiola bridge offers the fewest advantages. An attack across the river at Garagiola would not directly threaten US lines of communication, nor the forces defending the other bridges. And even after crossing they would still be in a precarious position, surrounded by dominating terrain.

- Attack T: Attacking directly toward Androida may crack open the US position in general and cut off US forces defending the Cujo bridge in particular. Attacking near Tragedia is more likely to prevent both the US's center and left from reinforcing Cujo.

- Attack CG: Attacking Garagiola fixes only the far left of the US's front, leaving his center free to reinforce his right at Cujo. This attack keeps US forces busy enough to allow the enemy to cross the river at Cujo.

- Attack CT: Being closer to the Cujo bridge makes it easier for two wings to reinforce and complement each other. This attack protects their left flank against a US attack from that direction.

- Attack TG: Schmitt did not consider this option in detail, mainly because it has significant negatives. DG would model this attack but rank it as unlikely.

- Attack CTG: These attacks provide the greatest number of possible options, but would disperse forces across the greatest frontage, making it difficult to concentrate quickly at any one spot. These attacks also leave less reserve to exploit whichever attack shows promise.

- Another option is for the enemy to use their AAVs. The problem is that these would be visible to the US during the entire assault, as they move slowly through the water on exterior lines. The US, motorized and using interior lines, would simply have to face left to meet the threat, and would have plenty of time to do so. Envelopment would not be deep enough to pose much of a threat to the US position and the enemy would

still be in front of the commanding terrain at Androida and the Androida heights and would not directly threaten US lines of communication.

Each of these options can be further split into a Hasty Crossing or a Deliberate Crossing

- **Hasty:** A hasty crossing is where the enemy launches directly into the crossing without loss of momentum, hopefully seizing one or more bridges intact and preventing the US from disengaging again. The enemy would want to strike in more than one place, so that they have multiple options, with the flexibility to reinforce whichever action shows the most promise. The main risk is that if US has made solid preparations, a hasty crossing may not be an option.

- **Deliberate:** Complex operation requiring thorough planning and preparation. Like above the enemy would want to strike in more than one place with the flexibility to reinforce whichever action shows the most promise. The risk for the enemy is that the US will have the option of withdrawing on their own terms, and the enemy would probably have to wait until later for the decisive battle.

The combinations have different pros and cons, but in general the single bridge attacks and the three-bridge attack would not rank as high for the enemy, although DG should still model them. Because the perceived enemy intent is to destroy US forces, not merely secure the bridges, one would expect that a Hasty Crossing would be a higher likelihood enemy COA than Deliberate Crossings. If the goal were just to secure the bridge, the deliberate one might be better.

LIME **Implications**

Some implications for LIME are:

- Enemy COA timing constraints have operational impacts. Here, the slower deliberate options allow for increased preparations. In another context/mission, this same situation might result in different, even opposite, operational impacts. Hence, there is not necessarily a canonical impact on operations for any given constraint, but instead it depends on the context of the state.

- Rather than including detailed unrolling of the enemy COAs into the Futures Graph, it may be more useful to use the simulation information to weigh the likelihood of certain outcomes occurring, for example, ranking the probability of the enemy attacking at any particular bridge by the outcome of simulating those COAs for the enemy.

2.2.4 DG During the Battle

To this point we have considered how DG will assist the Commander during preparations. DG should also prove beneficial as the battle unfolds. Suppose the Commander selected the conservative COA, perhaps after a bolder COA was overruled by the superior Commander. Let us assume that during the battle it becomes clear that the main enemy force is crossing at Tragedia in a hasty, focused assault. Kilo company is having difficulty holding the bridge. LAR and Lima Company are holding at Cujo. Garagiola appears to be uncontested.

DG informs Commander that the Future where the Enemy successfully crosses Tragedia is becoming more likely. The Commander draws up several more COAs:

- I company moves directly North to reinforce K.
- I company crosses Garagiola bridge and attacks the enemy's right flank.
- Commit tank reserve to support K Company.
- Destroy the Tragedia bridge with air strikes.

A question for the DG program to address is what to do if some of these new COAs indicate a different Commander's intent. Does that get flagged to the Commander (e.g., to confirm the change)? Do those initial Futures get pruned (because they don't match Commander's current intent)? Or do they cease to be annotated as intended (desired) future states of the Commander and therefore become less likely?

Futures involving certain enemy COAs should be considered more likely and some others pruned away as the enemy commitments become known. Perhaps additional COAs should be added: for example, what happens if the enemy has reserves in the woods around Garagiola? Commander selects from these, issues orders, and so on.

LIME Implications

Some implications for LIME are:

- DG needs to simultaneously account for different configurations of objects, for example, different force configurations (enemy and friendly), different weather outcomes and changes, different status of key terrain (bridge out or not).
- DG must distinguish *real* information (the COP) from "what-if" information postulated as part of some Futures. DG must also accommodate changes to information classification: speculated information could very well become real information as the battle unfolds.

2.2.5 Mid Intensity Conflict: Multi-Echelon Deep Green Use Case

The MIC scenario has a request to the upper echelon regimental Commander for additional air support and access to a battalion being held in reserve. In this section we explore how DG systems operating at both the battalion and regimental levels might interact to capture these requests. Figure 2.4 illustrates the relationship between the instances of DG: DG-B3 is assumed to be the Deep Green at the lower 3d battalion, DG-B alone refers generically to a battalion-level DG, and DG-R is the higher, regimental DG.

Suppose that the Commander issues his intent to all of the battalions, and that this is also transmitted to the respective DG-B. The COAs that are already modeled (or the futures that correspond to this new intent) would be made available to the battalion Commander as he adjusts plans to meet this new COA.

Presume that the regimental Commander's intent was as described above and the 3d battalion Commander developed COAs also as described above. As these are unrolled by

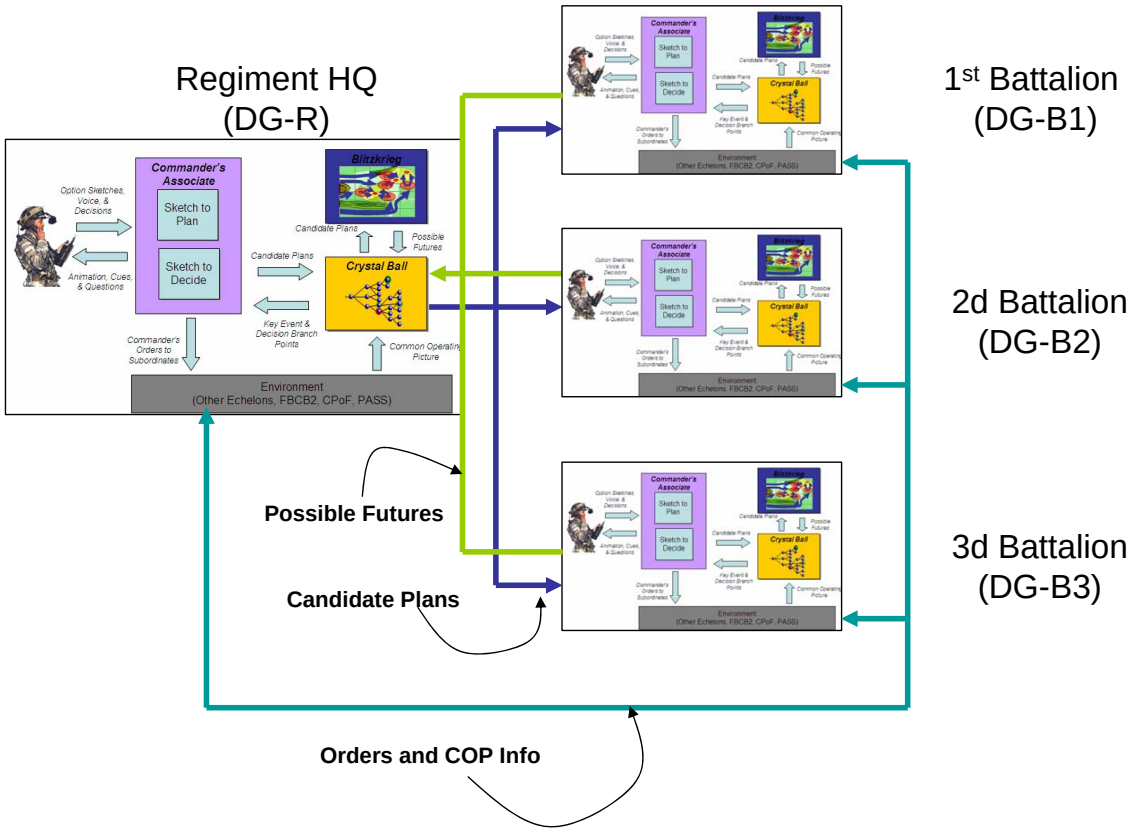

Figure 2.4: Notional example of upper and lower echelon DG systems interacting

DG-B3, they should also be reflected at DG-R as inputs into its possible futures. In effect, the DG-B systems serve as additional Blitzkrieg systems for the DG-R. This means that Crystal Ball should be able to accept multiple subgraphs, including those from lower echelon DG.

From the perspective of the DG-R, the DG-Bs in this example serve as additional Blitzkrieg systems, accepting plans from DG-R, enhancing them with Battalion Commander inputs, unrolling them (like normal), and then feeding them back up to DG-R as additional potential futures to be merged in to the DG-R Futures Graph.

Some observations:

- Each DG-B needs to reflect a Commander's intent that is consistent with that reflected in DG-R. Note that it would likely be necessary, however, to distinguish between intent that is being used for planning and intent that has been ordered. When hooked together, the individual DG systems could (and probably should) be able to access the speculative directions each of the Commanders is heading.

- In deciding what orders to give the battalion commanders, the regimental commander would need to consider what each of these battalions are capable of doing and what threats or other constraints they face. Integrated DG should make this easier.

- Each of the DG-B3d COAs would be folded into the COA being considered by DG-R, as would the enemy COAs developed at the DG-B3d. DG-R's Crystal Ball will need to

merge the lower echelon Futures Graph (from all of the DG-Bs) into its own Futures Graph at an appropriate level of abstraction.

- The DG-B3d COAs that utilize other regimental forces need to be reconciled with those COAs developed by other battalion commanders that also use those forces. Helping the regimental Commander to make appropriate tradeoffs. For example, it is possible that more than one Commander is requesting access to the reserve battalion. DG-R can help discriminate between these different options.

- DG-B for the reserve battalion would benefit from knowing the futures of the other battalions (i.e., the battalions it may potentially be supporting with contingency response) that would require their services. This would help the reserve battalion Commander to pre-position forces to provide this response.

LIME **Implications**

Some implications for LIME are:

- Each COA needs to be marked as being speculative or selected (ordered by the Commander) because this impacts several aspects, not the least of which is likelihood, if the Commander has selected and ordered a COA then it becomes much more likely.

- Generally there is a single Commander'sintent and all competing COAs must satisfy it. In the multi-echelon DG case, however, a battalion Commander may be supporting (directly or indirectly) multiple other Commander's, each of whom will likely have differing (although related to their common superior Commander's) intent. As futures are unrolled for what could be a large number of complete and partial COAs, it will be important to maintain a tie to the specific Commander's intent(s) that the COA is intended to meet. Further, during speculative planning, a Commander may be exploring with different intents and so may, in fact, have multiple intents. This means that LIME should allow for a COA or part of COA to map to multiple intents. In fact, COAs that meet multiple intents might be particularly interesting to the Commander.

- Although this is not a LIME issue per se, it is worth noting that options that improve the overall utility for the regiment need to be traded off against the battalion-level utilities.

- By providing visibility to the different COAs, including the enemy COA, the combined DG should be able to make the aggressive DG-B COA more likely and also be able to trade off the effects of executing that COA against ones from other battalions.

- Objects need to be distinguishable across DG systems. It is possible that two battalions might specify the same name for a different locally scoped object. For example, more than one battalion may have a Company A or specify "Objective Able." Conversely, battalions could also specify different names for the same terrain (Cujo Bridge or Bridge A). At DG-R, these variations need to be reconciled and a mapping maintained so that no errors are introduced into interpretations of the subordinate systems.

2.3 Humanitarian Aid Scenario and Use Cases

This humanitarian scenario is adapted from the Binni Scenario [16] found at www.binni.org. The following is a brief historical summary of the events leading up to the subsequent tactical scenario. For brevity, items not directly relevant to the tactical scenario are not mentioned (see full scenario for details).

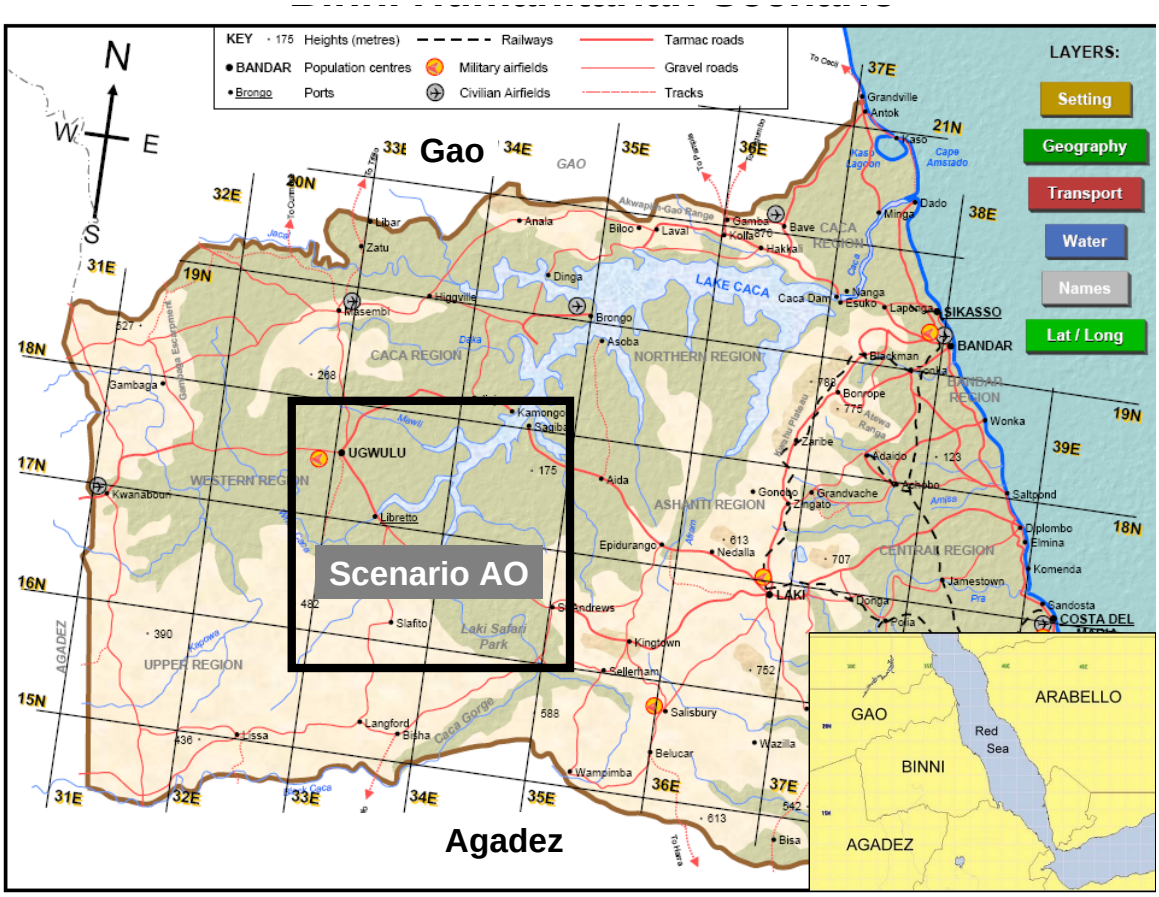

Figure 2.5: Binni and the area of interest for this scenario, pictures are from www.binni.org

- Two fictitious, neighboring countries, Gao and Agadez, in the Sudanese plain have seen an environmental change that resulted in vast tracts of land which were uninhabited/uninhabitable becoming arable and itinerant people returned to their homelands to become farmers and landowners in an agrarian society.

- Infrastructure and port access are major issues, particularly for Gao which is essentially landlocked, with Agadez controlling the only deep water port in the area. Note the map above is *after* the Gao action that created Binni out of terrain that used to be part of Agadez.

- Tensions rose with boundary disputes frequently leading to vigilante actions, increasingly open hostility between communities due to the absence of an adequate law enforcement infrastructure.
- Agadez has a substantial international loan from the World Bank to develop Port of Sikasso and the road and rail infrastructure.
- Agadez applied a significant tax on the transportation of food across its boundary from Gao, this seriously impacted Gao's economics of production and they threatened military action.
- Fragile agreement on access to the sea was not sufficiently comprehensive and was broken when disruptions occurred to the flow of goods from Gao to the port of Sikasso (caused by repeated terrorist attacks on road and rail convoys, widely suspected of being conducted by clandestine agents of the Agadez Government). Gao products that did reach the port were often delayed by bureaucratic regulations, causing perishables to rot.
- Gao launched a successful preemptive strike to open up a corridor to the sea. Caught Agadez by surprise and was accomplished with little local resistance because the indigenous people were sympathetic.
- Gao declared the annexed area to be the independent country of Binni.
- Agadez has launched repeated guerrilla activities to dislodge the Gao forces from Binni.
- The Provisional Government of Binni was established and sought the protection of the UN in order to secure its stability.
- A UN coalition peacekeeping force was established to enforce the Binni borders.
- Agadez guerrillas poisoned the watercourse in the region of conflict at several points leading to the fertile lowlands and the industrialized coastal region of Binni.
- A serious concern to the World Organization for Health (WOH), the cause of the deaths is a toxin of unknown origin. There is an immediate need for expert analysis by the WOH that can only be conducted on-site at substantial risk to the investigators. There are only two known specialists with the experience necessary to deal with the field issues surrounding the problem. Their access to the affected regions and their protection during the subsequent UN operation is of paramount importance.

With this context, the UN passed a resolution to create and deploy a UN War Avoidance Force for Binni (UNWAFB). This force is composed of military resources from four UN member nations, supplemented by advisors and personnel such as language/dialect and cultural specialists from the international community. The relevant portions of the UN mandate are to:

1. Stop the Gao forces in the rout of the Agadez forces.
2. Survey the area of contamination by the unknown toxin, identify its dispersion characteristics and assess population/wildlife environmental effects.
3. Establish a total exclusion zone (TEZ) between opposing forces.
4. Attend to the sick/injured and arrange safe enclaves for the displaced in the local population. Prepare the way for medical and humanitarian aid organizations.
5. Collect evidence for subsequent international criminal proceedings and reconciliation of the distressed population.

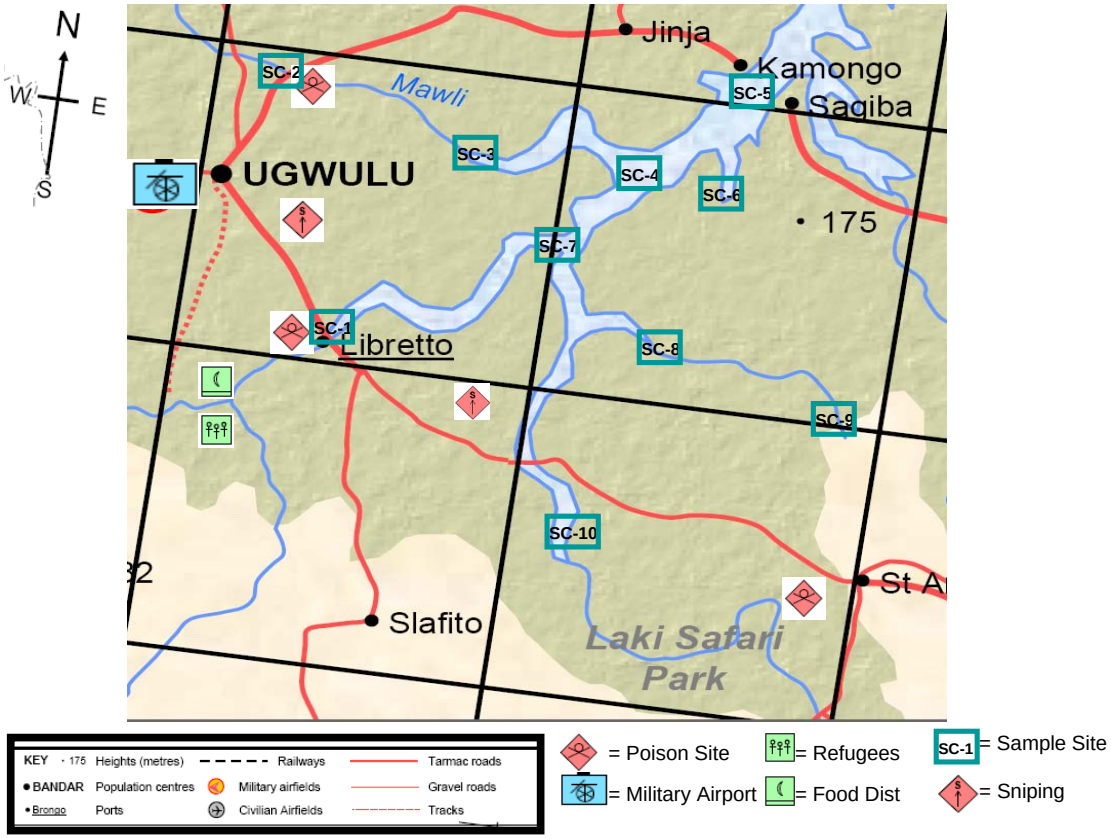

Figure 2.6: Map of the area of operations for this scenario showing current situation

2.3.1 Humanitarian Aid Scenario Description

This scenario focuses on a particular battalion of the UNWAFB, comprised of forces from two countries, the US and Baz (yet another fictional country).

Mission: The primary mission is to protect experts surveying the area of contamination and secure road from Ugwulu airport to Libretto. The secondary mission is to provide humanitarian assistance to the local populace. In the background, forces should be collecting evidence for subsequent international criminal proceedings and reconciliation of the

distressed population. The Commander's intent is to continue the sample collection survey (targeting one, two, three, or possibly more in a single day or multi-day missions), protect a humanitarian aid convoy from Ugwulu, and provide security for construction of a well for the refugees.

Force Structure: The UNWAFB 2nd Land Battalion headquarters is in Libretto, approximately twenty miles from UGWULU military airport. Comprises units from two of the four coalition countries, the US and Baz.

- Company A: US Army mechanized infantry unit with armored personnel carriers and several tanks, organized into four platoons of roughly equal strength.
- Company B: Baz mechanized infantry and logistics supply contingent, primarily responsible for convoy operations. Arabic is the primary language, which is also the most prevalent language in Binni.
- Company C: US Marines amphibious assault unit, squadron of Blackhawk helicopters, hazmat unit, and a small special forces unit with skills in indigenous populations. Also attached are the two civilian experts embedded to deal with the unknown toxin.

Operational Situation:

- The Battalion based in Libretto have set up HQ and a lab, conducting searches and other investigations to collect evidence and understand poisoning.
- Three sites where initial poison was delivered have been established and experts have identified 10 sample collection sites. To date, one was easily collected (site 1) and three have been successfully visited by helicopter (sites 6, 8, 9). The last helicopter mission (9) drew small arms fire southeast of Libretto (see map) on return flight.
- Convoys to/from Ugwulu have come under sniper fire in last two weeks.
- Humanitarian aid organization has set up food and water distribution site upstream from the poisoning. This site provides good water, but a contractor has been hired to drill a well as a contingency against more poisoning, or needing more water.
- Refugee population is growing quickly, and likely has been infiltrated by Agadez guerrillas.
- Concerns the terrorists will re-poison streams, upstream from refugee camp.
- Small arms fire at helicopters has been increasing, particularly in area north of Libretto.
- Guerrilla activity is picking up in the corridor between Libretto and Ugwulu and areas to east of Libretto.
- Trash is increasing on the airport road, increasing risk of IEDs.

- Guerrilla's weapons include small arms, shoulder fired SAMs, IED and demolition capability. They are well-stocked with ammunition and well trained. Thus far, they are not inclined to suicide tactics.

- Small arms fire on site 9 mission may indicate Agadez guerrillas have picked up a pattern, or have a forward observation post.

- Presume guerrillas know about the supply shipment arriving in the next couple of days.

- A Baz-based contractor is on location to construct new wells for the area. They have well drilling and other light construction equipment. Safety and welfare is responsibility of UNWAFB Commander.

- Lakes and rivers in Libretto and downstream are not safe for drinking, but refugees are pouring in.

- Refugee distress has been exacerbated by the poisoning of watercourses by Agadez terrorist elements, and the consequent death of cattle and other domestic animals.

- Some 90 staff from the UN aid agencies are being transported by land from Gao to set up the initial humanitarian aid support camps across Binni. Ten staff are in Libretto. Their safety and welfare will also be the responsibility of the UNWAFB Commander.

- Gao forces in area are providing rear guard to Gao forces and serving as Binni law enforcement. Three units, totaling approximately 100 infantry soldiers armed with modern portable weapons including rocket launchers and heavy mortars.

Read as: Row is <cell> to column	US	Baz Contractor	Refugees	Agadez Guerillas	Gao Forces	Civilians
US	Friendly	Friendly	Friendly	Hostile	Friendly	Neutral
Baz Contractor	Friendly	Friendly	Neutral	Hostile	Friendly	Friendly
Refugees	Neutral	Friendly	Friendly	Neutral	Friendly	Friendly
Agadez Guerillas	Hostile	Hostile	Hostile	Friendly	Hostile	Hostile
Gao Forces	Friendly	Friendly	Friendly	Hostile	Friendly	Friendly
Civilians	Neutral	Neutral	Hostile	Hostile	Neutral	Friendly

Figure 2.7: Matrix of affiliation assumptions of the scenario, including asymmetric affiliations.

Figure 2.7 shows the affiliations between the participants. The asymmetries are highlighted as gray boxes. In practice, these affiliations may not be known with certainty, may shift, or may not apply to the entire participant set.

Temporal/Physical/Environmental Situation:

- It is early winter and storms are forecast for the next two weeks.
- Considerable rain is forecast in the region of conflict, and the terrain is becoming increasingly difficult.
- Low level flying and high level reconnaissance missions will be limited.
- Terrain comprises a mix of hills with steep escarpments, swamp lands and dense forest/jungle growth.

Potential Courses of Action and Options: The Commander needs to consider many things in determining how to array forces. Again, the primary mission is to continue the sample collection missions without loss of the experts. The Commander must worry about predictability and safety at the remaining sample mission sites 2, 3, 4, 5, 7 and 10. COAs can attempt 0, 1, 2 or more of these missions. In addition to helicopters, some are accessible by amphibious vehicles (4, 7, 5) and one is also accessible by road (2).

The refugee situation is of concern. The Commander needs to protect them, and prevent riots or other discontent. Construction of alternate water supplies is important. The road is seeing increasing guerrilla activity. The Commander needs to protect the convoy due to arrive today from airport.

Possible COAs for the different mission objectives for the day include these four:

1. **Collection Mission (6 Remaining)**
   - Company C assets to 1, 2 or 3 sites. Note that there are over 150 combinations of sites meeting this criterion.
   - C + B Assets (use B to access Sample Collection (SC) Site 2 via road)
2. **Convoy Protection Mission**
   - Company B (logistics unit only, self protect)
   - B (logistics + infantry)
   - B plus some C (e.g., helicopters)
   - B plus A (tanks and infantry)
3. **Security of Well Construction Mission**
   - Company B (infantry)
   - B, plus some C (e.g., SOF)
   - B, plus A (tanks, infantry)
   - A or C in combination

4. **Force Protection Mission**
    - Company A, B or C assets only
    - A, B, C in combination

While these COAs must adhere to the Commander's intent, complications that may arise that change the likely possible future states include:

- Weather changes.
- Ever present uncertainty, fog of war, etc.
- Enemy actions such as a new poisoning.
- Discovery of new, actionable intel.
- Neutral and Civilian actions.
- Constraints between units.

2.3.2 Humanitarian Aid: COA Options and Abstraction

Figure 2.8 maps the COAs specified by the Battalion Commander which we discuss in the following use case and in several others in this section. The COAs assigned to the companies are:

- **Company A:**
  - Secure Sample Collection (SC) Sites SC-1, SC-4, SC-10. Clear and secure landing zones (LZ) at each location. Protect survey party and gather all required evidence
  - Secure and hold Bridge B7.
  - Provide platoon sized reaction force. Primary mission support of Company B.
- **Company B:**
  - Secure Route Ugwulu, provide convey protection.
- **Company C:**
  - Secure Sites SC-2, SC-4, SC-5 by ground or sea.
  - Secure Sites SC-3 by air assault. Clear and secure landing zones (LZ) at SC-2, SC-5.
  - Protect survey party and gather all required evidence.
- **Coordinating Instructions:**
  - Survey Team will be transported by air; SC sites must be secured prior to their arrival.
  - The sequence of SC site investigation will be: SC-3, SC-4, SC-2, SC-5, SC-7, SC-10.
  - Be prepared to hold survey sites for further investigation if there is evidence of contamination.

Here we consider only the sample collection mission aspects of the COA above. The Commander listed a specific ordering of these missions. At a high level of abstraction, these can indicate the Commander's desired end states, i.e., the state where SC-n has been successfully collected. One layer down, each of these states is broken into two other main states:

1. Site secured and landing zone created.
2. Sample collection completed by survey team.

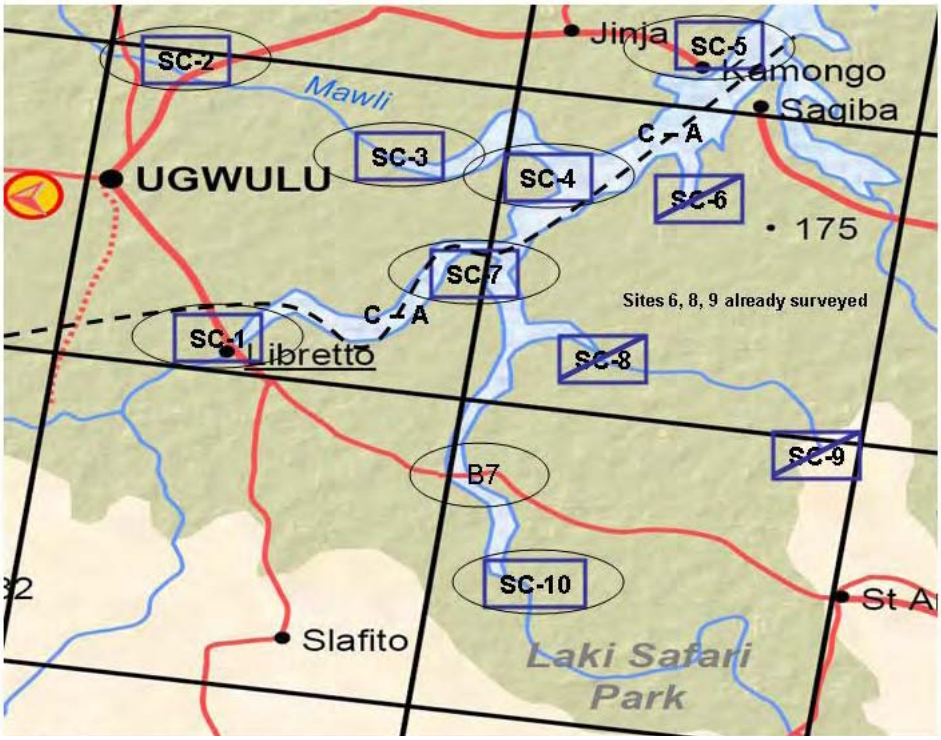

Figure 2.8: Map of the COAs generated by the Battalion Commander for the Binni scenario which we discuss in our use cases. Sample sites that have been collected are noted; areas of responsibility of Co-A and Co-C are indicated by the dotted line along the river.

At this level, a third state that represents the securing unit remaining behind if evidence is found could also be specified, but is omitted here to avoid unnecessary complexity in this discussion (DG would need to handle it, of course). Figure 2.9 shows the eight possible ways to achieve the Commander's intent with respect to the sample collections, given the options for maneuvering to the site. Figure 2.10 shows how this tree of eight state transitions can be abstracted into a simpler set of distinct high-level states.

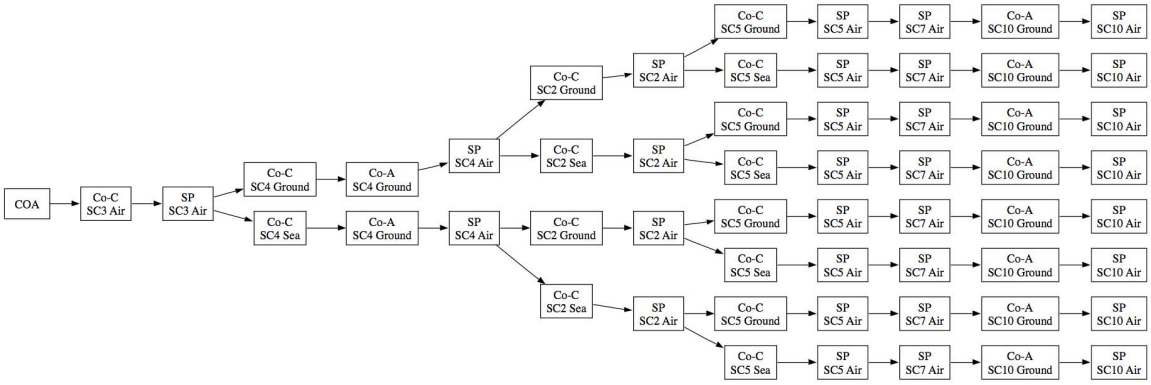

Figure 2.9: Eight distinct courses of action are possible from the Commander's specified COA above. Each box above is a desired state. For example, "Co-C SC3 Air" means the state where Co-C has secured and cleared SC-3, created and secured LZ, and is awaiting arrival of the survey party.

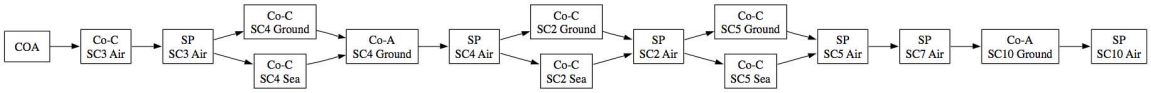

Figure 2.10: The states from Figure 2.9 can be collapsed into a smaller set of states based on common states, generally the portions where the survey party is present and collecting the samples.

The tasks involved in achieving each of the higher level states can be considered sub-states that must be achieved. Figure 2.11 shows, at a notional level, each of the sub-states within the Co-C-SC3-Air and SP-SC-3 states as described by the Commander. As in the earlier use case, they reflect the state being completed: so "Co-C Maneuver (air) to SC-3" means the state where Co-C has flown to SC-3. In DG these would be added by the Detail Adding Planner or some other aspect within Commander's Associate, provided to Crystal Ball, unrolled by Blitzkrieg into Futures Graph subgraphs and then merged back into Crystal Ball.

Each of the sub-states has a time period when it is active, and constraints among the states. For example, the "Co-C Secure SC-3" state requires that first the "Co-C-Clear-SC-3" state be entered. Several of these states may be active at the same time: for example, the site remains secure for the entire duration of the mission. In addition to a duration for each of the states, other constraints between them are required, as Figure 2.11 shows. Some

are implicit, for example Co-C cannot secure a landing zone until it is created; others are explicitly stated by the Commander, for example that the site must be secured *before* the survey party arrives.

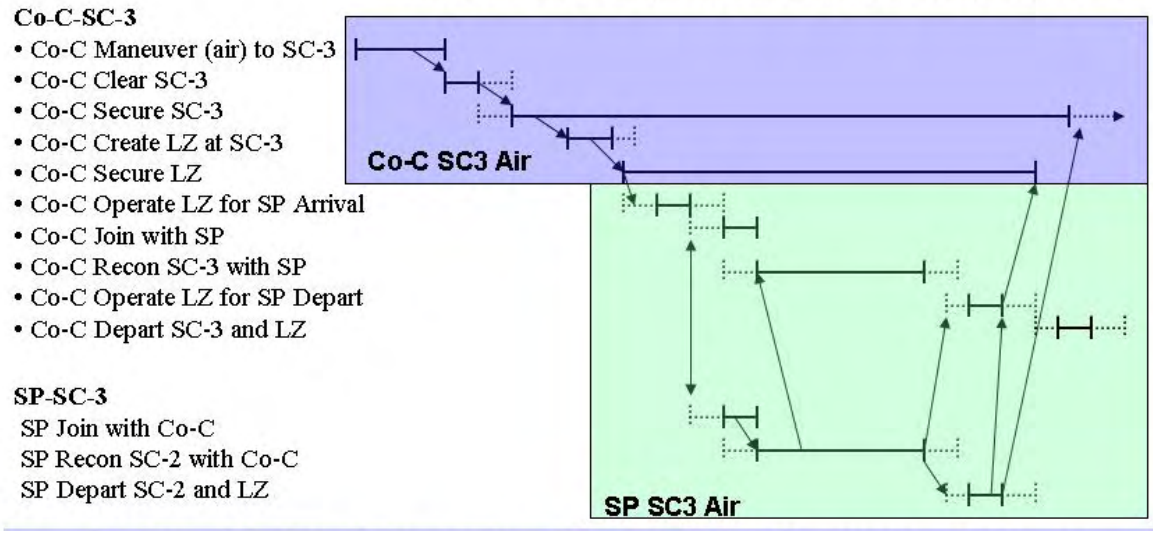

Figure 2.11: Constraints are specified between the intended states, represented as lines between each of the states. The sub-states are those specified by the Commander's COA.

LIME Implications

As in previous scenarios the use case carries implications for LIME:

- The COA specified by the Commander may well have a set of branches and options with it. For example, the Commander may opt to maneuver by air or by sea to several of the sites.
- The COA may not initially cover all of the missions. In the COAs above, the contractor constructing wells is not protected.
- The Commander will specify constraints, but their actual meaning may be subject to interpretation. For example, humans know that the intent of clearing and securing the sample sites is to keep them secure for the duration of the survey party's visit, but this relationship is not explicitly stated.
- Contingencies are considered but may not be specifically addressed. For example, the Commander COA specified Co-A provide a platoon sized reaction force, but the contingencies for which it should prepare (riot at refugee camp, attack on convoy, attack on survey party, etc.) are not enumerated.
- DG will be assumed to know about all of the following:

    1. The individual makeup of each of the companies that comprise the Battalion.

2. All of the participants, including refugees, the Gao, guerrillas, and so forth.
3. Key terrain, such as Bridge B7 and each of the SC sites.
4. Referenced mission tasks (SC, clearing a LZ, securing sites, etc.) including preconditions, postconditions, durations and other such constraints on planning.

This information must be already in DG, or there must be a way to add it. For example, Commander's Associate could notice that Bridge B7 is indicated, and create a new key terrain to represent it, adding it to the model.

- Abstraction is an important part of DG, starting right at the basic Commander-specified COA.
- COAs indicate the intended states that the Commander wants to achieve, e.g., each of the sample sites visited, as well as some of the important intermediate states (such as creating an LZ, and where importance is determined by the Commander).
- Constraints specified in the COA and unspecified operational constraints must be represented and it may not be the case that the Commander will identify all of the constraints: some will need to be added.
- Key terrain features may be created as the result of executing a COA, e.g., the landing zone become a piece of key terrain once it has been secured. Further, this key terrain only lasts as long as the sample mission lasts, once the teams leave the area, the landing zone reverts to just regular terrain since the LZ then no longer functions.
- An entirely natural language interface for capturing COA is beyond the scope of DG, but an overly contrived vocabulary for specifying the COA is also out of scope. LIME needs a straightforward set of tasks that can be described by the Commander (and/or expanded upon within Commander's Associate) that mirror as closely as possible the way that the Commander expresses COAs.

2.3.3 Humanitarian Aid: Unrolling COA

Thus far, the use cases have only considered what the Commander *wants* to happen, but have not covered what *might* happen. Again looking at the Sample Collection tasks, the arc moving from one state to another may result in entering the next intended state: for example, maneuvering by air may result in the unit arriving at the site. In real life, however, any number of things may occur. In this example, the unit attempting to clear the site may come under fire. Figure 2.12 shows a notional example for the SC-3 mission. This representation is notional because in real life, the outcomes would be modeled via a directed acyclic graph (DAG) without the cycles depicted above. The point, however, is that it is not a simple matter of moving from state to state when there is an adversary involved. Further, this only considers a small set of outcomes, others are possible such as fog or other weather preventing safe flight operations.

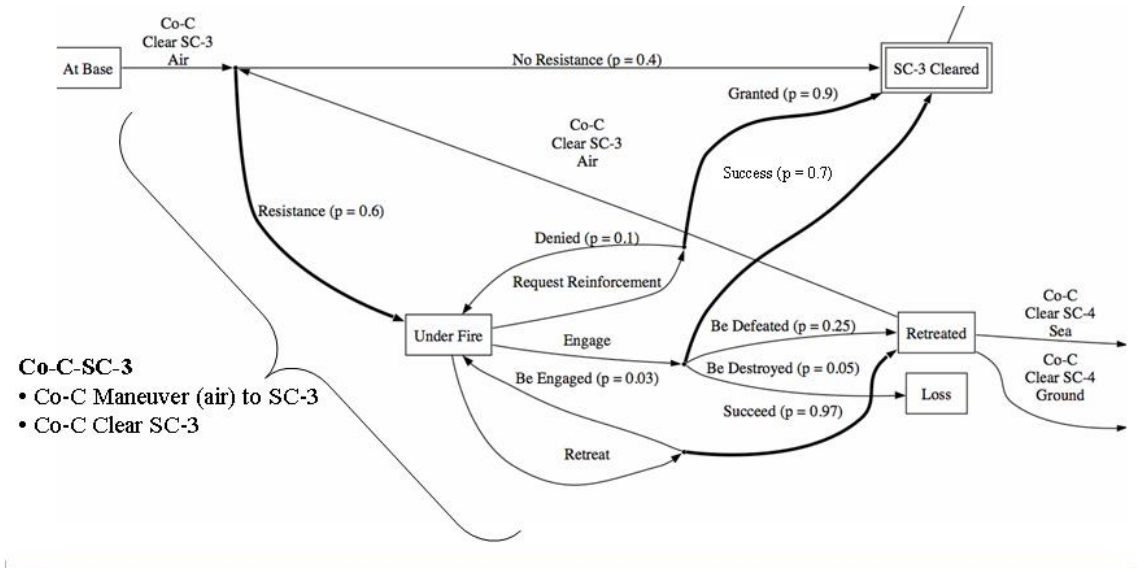

Figure 2.12: A notional example of the outcomes possible when moving by air to sample collection (SC) site 3.

LIME Implications

Some implications for LIME are:

- Moving out of a state may result in any number of end states.
- There are various ways by which we might discover outcomes: two examples are by explicit and *a priori* modeling, and additions during the unrolling by Blitzkrieg. But regardless of how we discover outcomes, the probability or likelihood of attaining them will be dependent upon other information in DG.
- The outcomes are dependent, to some extent, on the other COAs (for all of the "forces" in the AO, including weather and neutrals) that have been specified. For example, if the enemy COA for ambushing the arriving unit at a SC site is not represented in DG, then it is unlikely that it will be adequately considered in unrolling Friendly COAs. Of course Blitzkrieg may have some canonical outcomes for various actions, but these will necessarily be limited, and will probably not adequately take into account the specifics of the situation.
- When taken together with all of the possible outcomes of unrolling the COA, this will become a complex graph: dealing with this complexity requires abstraction.

2.3.4 Humanitarian Aid: Critical Point Analysis

Once the Futures Graph is created via unrolling and abstraction, it can be used to identify critical points where Commander input is required. This use case explores how that might occur in the context of COAs defined for the sample collection missions.

Suppose that Crystal Ball has reviewed the Futures Graph and determined two points that require the attention of the Commander. Figure 2.13 presents these points in abstract form. At these points Crystal Ball has identified problems found in the state:

- **Co-A-SC4-Ground:** where Company A is at SC-4 via ground assault.

  In this case, DG (i.e., some combination of Crystal Ball, Commander's Associate and Blitzkrieg) determined that there was a chance of Co-A coming under fire by Co-C, who arrived at SC-4 prior to the arrival of Co-A. In looking back at the COA, there was no explicit coordination specified between these two companies. The COA as specified, however, does meet the Commander's intent that the collection sites be secured prior to arrival.

- **SP-SC-7-Air:** when the survey party conducts the tactical reconnaissance at SC-7.

  Here the opposite of the above is true: the survey party is sent to SC-7 without an advance unit securing the site and establishing the landing zone. This violates one of the coordination constraints delineated by the Commander (specifically, that the sites be secured prior to survey party arrival). This should be caught in Commander's Associate, perhaps in the Detail-Adding Planner, but in the event it was not, then this plan flaw would be revealed in Blitzkrieg, assuming that a COA for guerrillas ambushing the survey party was generated and input into DG. The Blitzkrieg simulations would show an inordinately high risk for the SC-7 team, and the Commander, upon investigating, would see immediately that his Intent was violated by the COA.

The Commander can address these issues in several ways, the most obvious being the assignment of either Co-A or Co-C to protect the survey party at SC-7 instead of SC-4.

LIME **Implications**

Some implications for LIME are:

- DG will need to reason about the likelihood or probability of action outcomes other than those identified by the Commander.
- Analysis at lower, more detailed levels of the Futures Graph will need to be percolated up to higher levels of abstraction for presentation to the Commander.
- COAs for participants other than Friendly forces need to be represented.
- Critical point checking needs to be supported at multiple points in DG.

2.3.5 Humanitarian Aid: Neutral and non-combatant COAs

This use case considers the inclusion of neutral or non-combatant COAs in addition to the friendly and enemy COAs that have been considered thus far. In the scenario, there are many refugees and it is believed that the refugees have been infiltrated by the guerrillas. The friendly COAs will assign forces to secure and protect the refugee camp as above.

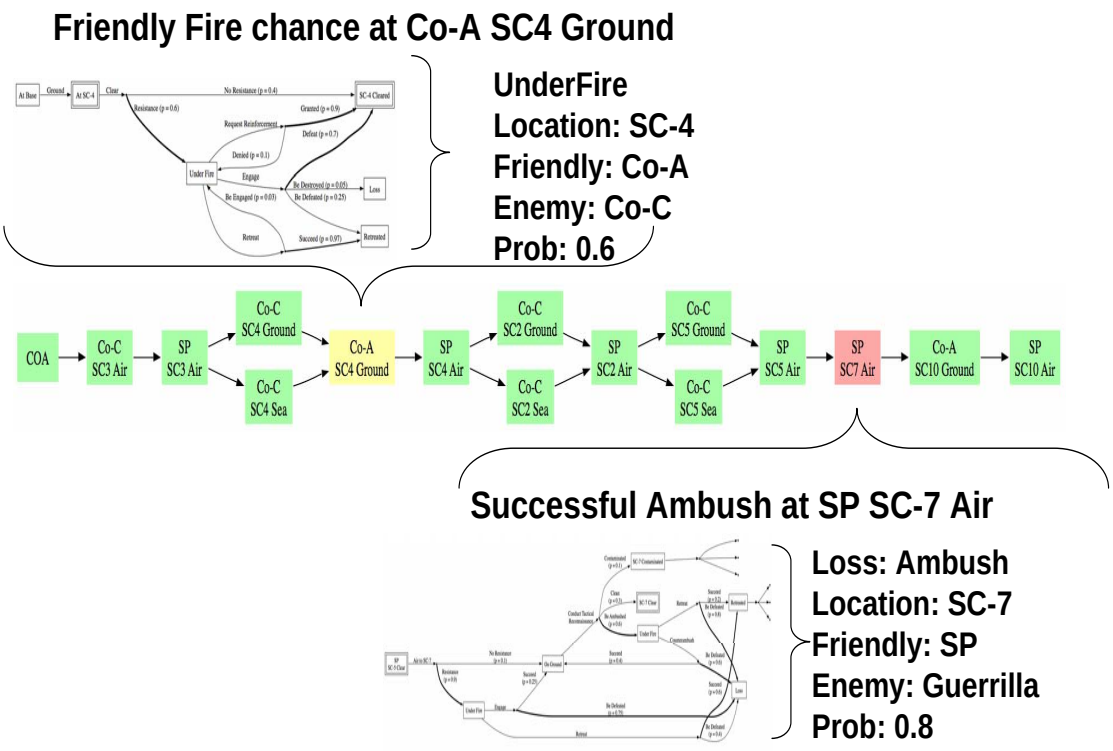

Figure 2.13: Notional example of analysis of the Futures Graph and presentation at an abstract level for the Commander. The details underlying the analysis are based on the potential outcomes of the actions taken.

In order to determine the potential futures, however, the Commander's staff will need to create COAs for the neutral parties. Suppose two COAs are considered for the refugees: either refugees remain peaceful, or they riot. Obviously, the former is the desirable COA.

For broadest generality, assume that these COAs are developed to occur at any time during the timeframe being simulated, say over several days. The unrolling of friendly COAs will need to take into account the likelihood of a riot given other information about the situation, both as defined in the candidate COAs and also as the current operational situation indicates. The pre-COA analysis will try to minimize the likelihood of rioting, while the operational analysis will address contingencies.

Relationships between the various participants (friendly, refugee, guerrillas, etc.) become important. For example, the likelihood of the rioting scenario depends greatly on the effectiveness (or numbers) of guerrillas that have infiltrated, the opinion of the non-aligned toward the US and the guerrillas, and the conditions in the camp. For example, if the camp is sanitary and the refugees' needs are being met, the people will be less likely to riot; on the other hand, if supplies or water is disrupted then the likelihood goes up. Of course, we have no way of knowing what *will* happen, but DG must consider the range of what *might* happen and then abstract these possibilities into qualitatively different states.

The affiliation matrix should be used to drive (and track) force allocation decisions. In

this case, DG might play out these various COAs and determine that the friendly COA with best utility is having Co-B in charge of refugee security. At least one reason would be that the refugees perceive Co-B better (friendly) than the US (neutral), even though Co-B is only neutral to the refugees while the US is friendly.

Given this asymmetry, however, it would be wise for the Commander to develop contingencies if the viewpoint of the refugees change, e.g., Co-B commits some act of violence. In this case, perhaps the Commander specifies that the Co-C special forces unit with skills in indigenous populations be part of the contingency response force for the convoy protection, food distribution and refugee camp protection.

LIME **Implications**

Some implications for LIME are:

- Asymmetric relationships need to be considered during the unrolling, including relationships that are not currently present: relationships can change over time and the plan needs to be ready for the more likely changes.

- The relevant affiliation relationship is the one that might cause problems with a plan. For example, it may not matter that the US is friendly to the refugees if the refugees end up being hostile to the US.

- It is likely that the neutral COAs will play a bigger role in affecting the likelihoods and utility of unrolling the friendly COAs rather than specifically unrolling the neutral COAs into their own Futures Graphs.

2.3.6 Humanitarian Aid: Addressing Timing Issues

The Commander identified a specific ordering of sample collection missions and units to fulfill those missions. He did not specify a total duration within which they should be accomplished, the time between landing zone being secured and survey party arrival, timing differences of sea or air assaults, timing and resource issues among his forces in accomplishing the other missions, nor any other such constraints. There are several possibilities here:

- DG, namely the Automated Option Generator, could relax the Commander's ordering constraints and look at permutations of the sample collection missions.

- DG will be expected to provide timing feedback to the Commander, and in particular to determine the time range in which the specified COA is achievable. This means that if the COAs cover multiple days, Blitzkrieg will need to address operational issues like adequate sleep.

- Operations may be indefinitely ongoing, or they may be of fixed duration. For example, we may unroll futures where the units need to secure the sites indefinitely, which brings up staffing issues. This example contrasts with the assumption that an operation is wrapped up at the conclusion of the sample collection.

- Include timing of the convoy, using estimated arrival times of the supplies, ranges of travel along the road, and other applicable factors.

- As other COAs are added, conflicts may be created in the Futures Graph where those units cannot be in two places at the same time. But at an aggregate level (for example, with a company), an object *can* be in two or more places at the same time. It is possible even that a platoon can be in two places at the same time.

LIME Implications

Some implications for LIME are:

- Representations will be incrementally changed, and there may be different versions of the Commander's COA being evaluated simultaneously.

- The difference between Commander analysis activity, options and *orders* provided to the units. Analysis activity includes exploring different COAs, for example. Once orders are issued many of the options of the analysis would become moot, while other COA possibilities will continue to be evaluated — but those with portions that match the orders should be given a higher likelihood.

- COAs will be added incrementally to the system, and these new COAs must be integrated with previous COAs. In particular, the system must determine whether the new COAs are additions, alternatives, or replacements for COAs already in place.

- The Commander might decide to split into two survey parties and see how those unroll. Thus, the representation needs to handle multiple views of unit composition.

- LIME may need an *indivisibility property* in its representation of the domain that indicates items that cannot be further broken up. This will likely vary based on the echelon of a particular DG instance.

- COAs that restructure troops will need to be considered. This restructuring can be done via orders but there are obvious timing issues: for example, when two units to be combined are 100 miles apart.

2.4 Counterinsurgency (COIN) Scenario and Use Cases

This scenario describes a counterinsurgency operation. This is a fictional situation drawn from several resources describing recent/ongoing insurgencies [8, 6, 15, 17] and approaches to COIN operations [10, 3, 15, 12]. While this scenario is based on real-world insurgencies, it does not nearly serve as a comprehensive survey of the nuances and variations of insurgent and counterinsurgent tactics. But this scenario does attempt to capture aspects of these operations that are distinct from combat or humanitarian operations. Another key difference is that this scenario is in an urban setting, while the others are in lightly populated areas. A COIN primer follows in Section 2.4.3.

Scenario background. An ethnic people, the Macs, allege that the Ghatistani government has long denied them political, cultural and linguistic rights. Mac resistance to Ghatistani rule has encompassed both a peaceful movement to assert Macish civil rights and autonomy within Ghatistan and an armed struggle, advanced by the Macistan Independence Movement (MIM), a nationalist organization working to incite a revolution in Ghatistan with the aim of establishing an independent Macish state. Intermittent discussions between MIM and the Ghatistani government have not led to lasting agreements. In fact, the Ghatistani government waged an intense military campaign against the MIM, targeting MIM members and those believed to support or sympathize with their cause, affecting many Mac families. The government deployed security forces, established militias in the region, and declared martial law. The Ghatistani military forcibly evacuated numerous villages, in many cases burning them in a scorched-earth campaign to prevent them from harboring MIM militants.

Many of the resulting refugees migrated to the already crowded city of Pomme, a large city of over 10 million people, straining its social services and resulting in turf battles between a large number of different factions including ethnic enclaves, religious groups, influential families, criminal organizations and MIM operatives.

Complicating matters, in the last few years the MIM altered its ideology to better accommodate Linic (a prominent world religion) in an effort to win more support from the population. This has resulted in a loose and sometimes contentious confederation between MIM and other violent radical Linici organizations.

The level of violence in Pomme and the country in general has been escalating at an almost exponential rate, threatening civil war and the sovereignty of Ghatistan, an important US ally. The Ghatistani leader requested and received assistance from the US in quelling the growing insurgency. Among other issues, Pomme has seen degradation of services, establishment of MIM-operated roving checkpoints, increased extortion and violent acts by criminals, and efforts to close the city markets.

2.4.1 COIN Scenario Description

This scenario focuses on the US Forces consisting of the 2d Light Infantry Brigade[1] assigned to the southern sector of Pomme, an area of approximately 100 square kilometers that has experienced the most violence.

Mission: The overall US mission is focused on several logical lines of operations (LLOs, see Section 2.4.3): the dimensions of COIN operations that guides military involvement, and helps coordinate other participants in quelling or otherwise responding to the insurgency. In an actual COIN operation there would likely be more LLOs than listed here, but this set is used to give the flavor and keep things tractable. We consider four LLOs:

- *Combat operations/civil security operations.* Secure the populace continually, separate insurgency from populace, counter crime (organized and petty), isolate insurgency,

[1]Future COIN operations will very likely make heavy use of the Stryker Brigades, the accelerated deployment of which has been partially motivated by the need to create forces which could deploy more rapidly and also have a smaller footprint in urban areas. Light infantry brigades are also used in COIN operations now and in the future.

integrate with host nation (HN) security forces (hand over responsibility on a case-by-case basis), and rebuild trust in HN security forces.

- *Essential services.* Sewage treatment plants operating, trash collected regularly, potable water available, electrical power maintained, transportation network maintained, schools and colleges opened, medical clinics and hospitals opened.
- *Economic development.* Mobilize/develop local economic activity, initiate contracts with local businesses to stimulate/maintain economy, restore/maintain medical clinics and hospitals
- *Information operations.* Support and enhance operations along all LLOs by highlighting the successes along each.
- *Diplomacy.* The military has a role both in *facilitating* diplomacy (*e.g.*, securing state department representatives) and also their role in *performing* diplomacy (*e.g.*, brokering an agreement between two factions in the Commander's AO).

We describe the events of the day related to these LLOs in the COA section below, after we first present details of the various participants.

Force Structure: 2d Light Infantry Brigade consisting of:

- One Field Artillery Battalion for indirect fire support (about 16 towed howitzers).
- Two Infantry Battalions (about 350 Infantrymen per battalion).
- An Armor Battalion for COIN operations (about 50 Tanks).
- Combat Support Battalion for sustainment operations (about 500 soldiers).
- A Headquarters and Headquarters Company for a light infantry brigade (about 100 soldiers).
- An attack/recon aviation battalion (about 30 OH-58D helicopters).
- Finally, one battalion of HN Ghatistani Forces infantry forces are assigned to work with US Forces.

Operational Situation:

- Ghatistani forces have a low profile because of recent civil unrest, bombings and other attacks against them. Part of the US mission is to build trust in these forces among the populace.
- Pomme Law Enforcement is believed to be infiltrated by MIM supporters, and is a frequent target of MIM attacks. Its paramilitary unit has been accused of human rights abuses, most recently for detaining a British human rights researcher.

- MIM militias set up roving checkpoints in the city, often using them to extort money. In one case, an argument erupted and a rocket was fired at the vehicle, injuring at least 2 or 3 people.
- Public lynchings of alleged Ghatistani supports have terrorized the populace.
- Favored MIM tactics include rioting, infrastructure attacks, abductions, sniper fire, executions and IEDs.
- Non-government agencies operating in the AO include the Red Cross, which provides humanitarian aide and which must maintain its neutrality; the United Nations Food Program, which receives and distributes food to local citizens; and the Cool World Program, which is distributing air conditioners to the local citizens.
- Unfortunately, despite a stated policy of peacekeeping the population views the US military as an extension of the Ghatistani government.
- A firefight between US Forces and MIM insurgents killed the only child of a prominent Linici family caught in the crossfire. Both sides blame the other, and the incident has eroded civilian trust in US Forces.
- MIM units have raided Cool World supplies to distribute air conditioners themselves, and take credit within the population.
- The civilian population does not trust Pomme law enforcement. An earlier protest became violent, with protesters throwing Molotov cocktails and stones at law enforcement officials. In response, law enforcement officials appear to have used excessive and disproportionate force in responding to the rioters. At least 13 people are believed to have been killed. Many other protesters as well as law enforcement officers were injured during the clashes.

Temporal/Physical/Environmental Situation:

- The AO contains over a million people, four hospitals, a water treatment plant, electrical substations, at least a dozen schools, four main and myriad smaller commercial areas, numerous parks, churches, restaurants, major highways, bridges, and other typical urban features.
- Some neighborhoods are relatively stable, while others are very contentious.
- It is summer, so air conditioning is particularly important. Tempers have been flaring more than usual because of the high heat and humidity.

Figure 2.14 summarizes the relationships between the major players. As with the humanitarian situation, it is very likely that affiliations are really some sort of probabilistic spread instead of the absolute statements we make in this matrix. Furthermore, in this scenario these affiliations are fluid and highly dependent upon future events.

From/To	US	Ghatistani Forces	MIM Insurgen	Civilian	Cool World	Red Cross	Radical Linics	Pomme Law Enforc
US	Friendly	Friendly	Hostile	Friendly	Neutral	Neutral	Neutral ->Hostile	Friendly ->Neutral
Ghatistani Forces	Friendly	Friendly	Hostile	Neutral->Hostile	Neutral	Neutral	Hostile	Friendly
MIM Insurg	Hostile	Hostile	Friendly	Neutral	Hostile	Hostile	Friendly	Hostile
Civilians	Neutral	Neutral->Hostile	Neutral	Friendly	Friendly	Friendly	Neutral	Neutral
Cool World	Neutral	Neutral	Neutral	Friendly	Friendly	Neutral	Neutral	Neutral
Red Cross	Neutral	Neutral	Neutral	Neutral	Neutral	Friendly	Neutral	Neutral
Radical Linics	Neutral -> Hostile	Hostile	Neutral	Neutral -> Hostile	Hostile	Neutral -> Hostile	Friendly	Friendly->Neutral
Pomme Law Enforc	Friendly	Friendly	Hostile	Friendly	Neutral	Neutral	Friendly->Neutral	Friendly

Figure 2.14: One potential affiliation matrix, showing changing affiliations and asymmetries.

Course of Action Options: In addition to generally moving the LLOs forward, the Commander must address the following specific events of the day. These items refer to the map[2] in Figure 2.15.

- Planned funeral today for a Linici child killed during a recent firefight between MIM and US forces, with each side blaming the other. Pomme law enforcement has the lead in providing security. Location of funeral is marked ① in Figure 2.15. (LLO = Civil Security)
- Protect distribution of air conditioners in densely populated sector F by Cool World. (LLO = Civil Security)
- Protect trash collection operations by City Services that cross sectors E and F. Note that these run from 0530 until 1400 daily with predictable routes (high potential for bombing, ambush, or other attacks). (LLO = Essential Services)
- Attend and secure a meeting with a Macish elder, representatives from the Red Cross, and the United Nations to discuss medical aid and food distribution. Meeting to be held at Red Cross HQ. (LLO = Diplomacy and Economic Development)
- Raid a reported MIM safe house at Location ② in joint operation with Ghatastani forces. (LLO = Combat Operations)
- Maintain contingency response, for example to MIM checkpoint reports. (LLO = All of them)

[2]Readers may notice that this is actually the public domain NIMA map of Baghdad. A fictional insurgent scenario is used, however, to make absolutely clear that this fictional scenario was fabricated for research purposes.

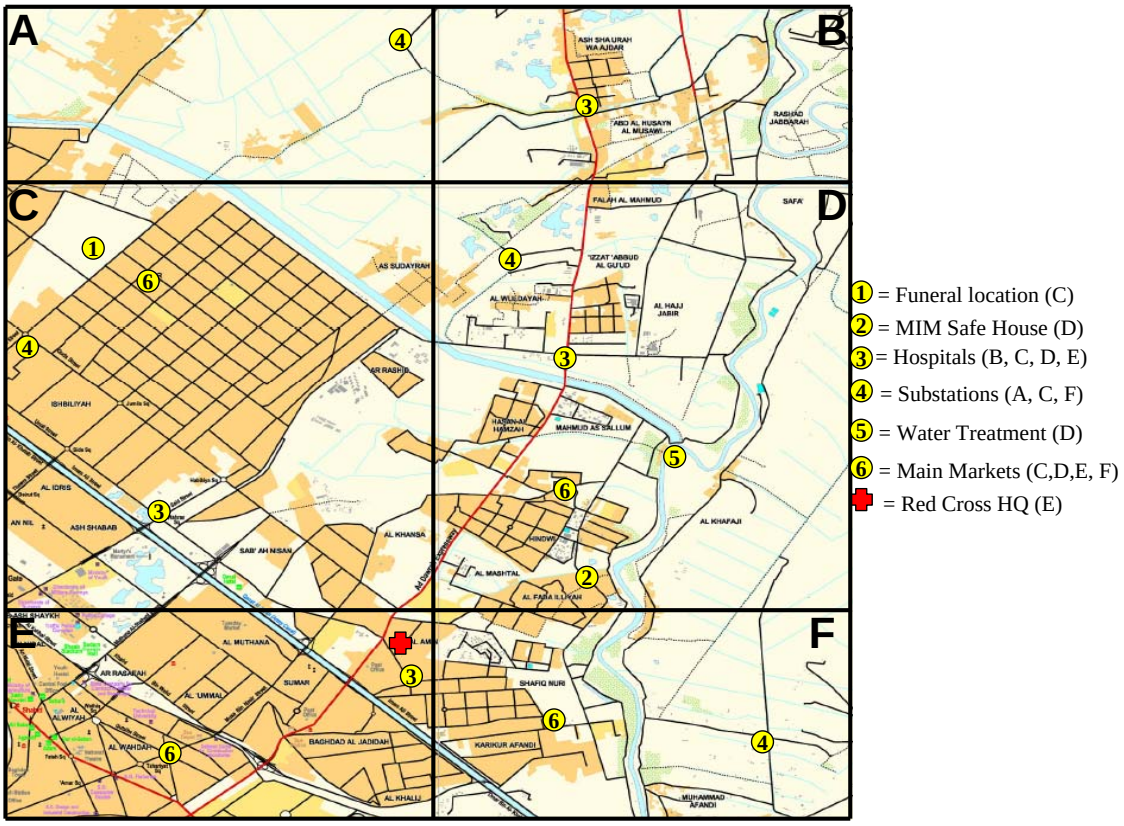

Figure 2.15: Map of the area of operations for this scenario showing the current situation. The yellow/orange shading reflects population density (orange is higher density)

- Conduct deterrent patrols across AO, including a visible presence in markets. Markets in Sectors C and D have historically seen recurring attacks. (LLO = Combat Operations and Economic Development)

- There are rumors of an attack on one or more critical infrastructure facilities, but no consensus as to the target. US forces need to protect the critical infrastructure: bridges, hospitals, electrical power and water treatment. (LLO = Essential Services and Economic Development)

These options entail thousands of potential courses of action: which targets to protect and how, where to deploy which patrols, timing of these with other participant COAs. One potential high-level assignment of duties might be the following:

- 1st Infantry Battalion conduct saturation patrols in sectors C and E with a focus on sector E, taking one company to vicinity of funeral. Provide indirect security to Cool World operations in Sector C, they do not want to be associated with the US or Ghatistani forces.

- 2d Infantry Battalion conduct saturation patrols in D and F, take down MIM safe house after dark. Consider a blue-green approach (see Section 2.4.3). Coordinate with Ghatistani forces and give them lead where appropriate. Maintain one reserve company for contingency operations.

- Armor Battalion divide tanks to provide direct support to 1st and 2d infantry battalions to protect key infrastructure areas, keep 25% available for contigency operations and maintenance.

- Combat Support Battalion conduct food purchase in each of the main markets over next week, coordinate with Infantry Battalions. Assist and provide security to the trash collection operation, provide visible security presence at Red Cross meeting, secure Commander's travels throughout AO. Deliver medical supplies to hospital in Sector C.

- Company-HQ secure Brigade headquarters, prepare information operation to publicize food purchases by US forces.

- Aviation battalion conduct aerial patrols over the city and maintain contingency response. Provide one unit to support 2nd Infantry Battalion taking down MIM. Steer clear of the funeral and markets unless needed for contingency.

- Field Artillery Battalion (indirect fire support) prepare firing positions to cover less populated areas in sectors A, B, D and F for targets of opportunity and contingency operations.

2.4.2 Counterinsurgency: Contingency Focus

More than any of the other scenarios, COIN concerns contingency and reacting appropriately as events unfold. Much of this occurs at levels below the Battalion and Brigade Commanders, at the level of the Strategic Corporal [11]. Of course, the higher-echelon Commanders have a role in arraying and guiding forces to best meet the unexpected events that will arise during each day's operations. Three types of contingency are present:

- *Pre-planned contingencies,* events that the Commander thinks might happen, and puts plans in place for dealing with them. We have addressed these contingencies in other use cases (*e.g.*, Section 2.3.5), but the difference with COIN is that there will be more participant groups, that is, more non-US COAs. Moreover the affiliations and other relationship matrices may be more fluid, and will very possibly have finer granularity, by neighborhood, or sector, or subgroups, or subsubgroups.

- *Tactical contingencies,* events that occur in field that are dealt with in the field, either because the field units are empowered to address them directly or the time scale is too fast for involving higher echelon Commanders. DEEP GREEN will not be able to help very much with these *while* they are happening, but information from them, how they were handled, and their result represent a learning opportunity across DG. For

example, Crystal Ball should update affiliation matrices and likelihood assessments, possibly detecting different critical points; Blitzkrieg simulations should learn the new potential outcomes and possibly become sensitive to local conditions (treating different zones differently); Commander's Associate can improve its ability to fill in plan details and generate additional COAs; and so on. It is up to the program to decide how to do this, whether automatically, under human initiative only, or (most likely) under mixed initiative. Note also that this learning should be coordinated with other DG systems running at different echelons, as we have already seen in Section 2.2.5.

- *Strategic contingencies,* events that require higher echelon Commander support. Examples include minor events that grow larger, or a major surprise where major replanning and adjustment is needed. Although strategic contingencies are issues in other operations, they are magnified in COIN, and so are the subject of the remainder of this example.

Suppose that shortly after the distribution of air conditioners in Sector F a significant blackout occurs, shutting out the lights in all of Sector F and part of Sector E, including Red Cross HQ and the nearby hospital in Sector E. The local electrical utility company informs the Commander that the blackout will last at least 6 hours, maybe longer, and is due to unknown cause. The blackout began two hours prior to the planned meeting at Red Cross HQ with the Macish elders, the UN, and the Red Cross. Any number of things could have caused the blackout, including:

- Hostile action on the substation by any number of malcontents.
- Accidental actions of the US forces protecting the substation.
- The increased load from the existing and newly distributed air conditioners.
- Tree falling at an inopportune time and location.
- Probably many others.

Although the cause is not immediately known, people have already jumped to conclusions and are reacting. In particular,

- Radical Linicis in the vicinity of Sector F (who recall are hostile to Cool World) are blaming Cool World, and inciting a crowd that has surrounded Cool World vehicles, now in Sector E.
- The MIM insurgent safe house is near the location of the blackout and, given the rumors of an infrastructure attack, US and Ghatistani forces immediately assume this was MIM doing and are on a heightened alert status.
- A group of civilians that live in Sector F saw US and Ghatistani vehicles heading toward the substation shortly before the blackout, so they are spreading the belief that the US did it.

- Everybody, even different US forces units, has varied knowledge of the blackout and its extent.

In this situation, DG can help sort out various options, and most importantly help prepare plans as things unfold. The Commander establishes his priorities in those sectors:

- Secure and meet with the elders, UN, and Red Cross as planned. This is a key diplomatic mission. Bring a fuel tanker to refuel the hospital's backup generators to bolster trust.
- Complete the distribution of medical supplies in Sector C, which he views as key to establishing trust with the Macish elder.
- Defuse the situation with the Cool World personnel and prevent it from becoming a riot, lynching or hostage taking. Assign to subordinate commander with diplomatic skills in his organization to pull this off
- Continue to support ongoing trash collection in the sectors affected by the blackout.
- Assist the utility company in restoring power, and determining the cause of the blackout.
- Assist the Pomme police directing traffic in the area.
- Prevent looting and civil strife in the blackout area, especially in the market.
- In conjunction with Cool World, set up cooling areas, powered with generators from Combat Support, for the elderly and others at risk from the heat. Help restore Cool World reputation.

Initially, DG can help the Commander find out what he does not know by identifying critical points in the Futures Graph where additional information would enable better quality of analysis. For example, the status of the substation is a key piece of information. The Commander needs to know if there is evidence of an attack or serious accident, or if the problem is minor or fixed. The Commander must also figure out where all of the forces and equipment are going to come from to accomplish his priorities. DG can unroll futures with different affiliation matrices: for example, suppose the civilians (at least in this area) become hostile to the US forces. How would that effect the Commander's plans? Which forces can be left alone to continue their present missions, and which should be retasked?

If the Commander pulls forces from other sectors, how does that impact the other missions for the day? Several options are available for the MIM safe house: use the confusion to increase surveillance of it to identify more insurgents, take it down immediately (with the risk of even more people thinking the blackout was intentional), or put more forces in the area of the safe house to provide security. Presuming it might be terrorist activity, how do things need to change to ensure the safety of the remainder of the power grid? We should note that protecting the substations may not be enough if the attack was perpetrated some

other way, such as by knocking down key transmission lines. Finally, which activities should have the local media or other participants involved? The Commander may be able to use the media to increase the profile of the Ghatistani forces, or to put the US efforts in a positive light.

LIME Implications

The implications for LIME are:

- Affiliation and other relationship matrices may be very narrowly scoped, and may change in a fluid way.
- Providing a means to identify what is not known is important.
- Tracking the impacts of changing sub-COAs on the other COAs is necessary, especially if those other COAs have already been sent out in orders.
- The line between what the Brigade/Battalion Commander needs to worry about, and what can be handled by the field Commander is blurry. LIME, and DG in general, will need some flexibility here.
- Previously ordered COA may be changed as events occur.
- Although not specifically a LIME issue, the simulation used to unroll futures will need to incorporate details such as diplomatic capability. A challenge here is that the unit best qualified to conduct any given diplomatic mission (in this case, defusing the situation with Cool World) depends on that mission. Further, the best *person* to lead the effort should be the one most qualified to do it, *i.e.*, it may not be the Commander or even any of his subordinate Commanders, but rather a 2nd Lieutenant who has been working in that specific section for the past three months.

2.4.3 Insurgency and Counterinsurgency Primer

Because COIN operations are different from conventional military operations, the following is an introduction to COIN operations as described in the Army Field Manual on Counterinsurgency [3]:

- An insurgency is defined as an organized, protracted politico-military struggle designed to weaken the control and legitimacy of an established government, occupying power or other political authority while increasing insurgent control.
- COIN is military, paramilitary, political, economic, psychological and civic actions taken by a government to defeat an insurgency.
- COIN operations combine offensive, defensive and stability operations to achieve the stable and secure environment needed for effective governance, essential services and economic development.
- Context is extremely important. The nature of the conflict and its focus on the populace make civilian and military unity a critical enabling aspect of a COIN operation.
- Insurgencies are local. They vary greatly in time and space. The insurgency one battalion faces will often be different from that faced by an adjacent battalion.
- The mosaic nature of insurgencies, coupled with the fact that all soldiers and marines are potential intelligence collectors, means that all echelons both produce and consume intelligence. This situation results in a bottom-up flow of intelligence. This pattern also means that tactical units at brigade and below require considerable support for both collection and analysis, as their organic intelligence structure is often inadequate.
- Participants in COIN operations include the following: US military forces, multinational (including host nation forces, US Government agencies, other governments' agencies, non-government organizations (NGOs), intergovernmental Organizations (IGO), multinational corporations and contractors, and HN civil and military authorities, including local leaders.
- Effective operations are shaped by timely, specific and reliable intelligence, gathered and analyzed at the lowest possible level and disseminated throughout the force.
- The US government prefers that US military forces operate with other nations' forces and not alone. Although the missions of multinational partners may appear similar to those of the US, rules of engagement, home-country policies, and sensitivities may differ among partners.

Killcullen [10] gives some insight into the challenges faced by commanders in COIN operations, phrased as specific observations. His article is interesting because of its relevance to COAs, the ways that operations might be run, and shows the smaller scale of COIN operations. His points are excerpted/summarized below:

- *Know the turf.* The Commander's task is to become the world expert on his district — its people, topography, economy, history, religion and culture, every village, neighborhood, road, field, population group, leader and ancient grievance.

- *Diagnose the problem.* Who are the insurgents? What drives them? What makes local leaders tick? Counterinsurgency is fundamentally a competition, between each side, to mobilize the population in support of its agenda. So the Commander must understand what motivates the people and how to mobilize them.

- *Organize for intelligence.* Operations will be intelligence-driven, but intelligence will come mostly from the Commander's own operations, not as a *product* prepared and served up by higher headquarters. So the Commander must set up a company intelligence unit and possibly platoon level as well, considering carefully where best to employ any linguist resources available. There may be one less rifle squad, but the intelligence section will pay for itself in lives and effort saved.

- *Organize for inter-agency operations.* Almost everything in counterinsurgency is inter-agency. And everything important, from policing to intelligence to civil-military operations to trash collection, will involve the Commander's company working with civilian actors and local indigenous partners he cannot control, but whose success is essential to the operation.

- *Travel light.* US Forces will be weighed down with body armor, rations, extra ammunition, communications gear, and a thousand other things. The enemy will be far less encumbered. The Commander should lighten this load, while making sure forces can *reach back* to call for firepower or heavy support if needed.

- *Harden your Combat Service Support (CSS).* The enemy will attack the weakest points. Most attacks on coalition forces in Iraq in 2004 and 2005, outside pre-planned combat actions, were against CSS installations and convoys. CSS assets need to be hardened, have communications, and be trained in combat operations.

- *Find a political/cultural adviser.* In a force optimized for counterinsurgency, the Commander might receive a political/cultural adviser (POLAD) at company level: a diplomat or military foreign area officer, able to speak the language and navigate the intricacies of local politics. Back on planet Earth, the Corps and Division commander will get a POLAD, but the COIN operation will not, so the Commander must improvise. Key to this is finding a political/cultural adviser from among the Commander's soldiers — perhaps an officer, perhaps not.

- *Train the squad leaders, then trust them.* Counterinsurgency is a squad and platoon leader's war, and often a private soldier's war. Battles are won or lost in moments: whoever can bring combat power to bear in seconds, on a street corner, will win. The Commander should train squad leaders to act intelligently and independently without orders.

- *Rank is nothing, talent is everything.* The Commander should find the "naturals" at counterinsurgency and put them into positions where they can make a difference. Rank matters far less than talent.

- *Have a game plan.* The final preparation task is to develop a game plan: a mental picture of how the Commander sees the operation developing. It will likely change, but having a plan is essential.

- *Be there.* The most fundamental rule of counterinsurgency is to be there. US Forces can almost never outrun the enemy. If forces are not present when an incident happens, there is usually little that can be done about it.

- *Avoid knee jerk responses to first impressions.* Don't act rashly, get the facts first. The violence the Commander sees may be part of the insurgent strategy, but it may also be various interest groups fighting it out with each other, or settling personal vendettas.

- *Prepare for handover from Day One.* The Commander should start handover folders (lessons learned, details about the population, village and patrol reports, updated maps, photographs, etc.), in every platoon and specialist squad, from Day One. Ideally, the Commander would have inherited these from his predecessors, but if not they should still be started.

- *Build trusted networks.* The key task is to build trusted networks. If successful, they will grow like roots into the population, displacing the enemy's networks, bringing him out into the open to fight, and seizing the initiative. These networks include local allies, community leaders, local security forces, NGOs and other friendly or neutral non-state actors in the AO, and the media. Actions that help build trusted networks serve the Commander's cause. Actions, even killing high-profile targets, that undermine trust or disrupt the Commander's networks help the enemy.

- *Start easy.* Start from secure areas and work gradually outwards by extending the Commander's influence through the locals' own networks.

- *Seek early victories.* This is not necessarily a combat victory, rather it may be something like resolving a longstanding issue the Commander's predecessors have failed to address, or co-opting a key local leader who has resisted cooperation with our forces.

- *Practice deterrent patrolling.* There are many methods for this, including "multiple" patrolling where US forces flood an area with numerous small patrols working together. Each is too small to be a worthwhile target, and the insurgents never know where all the patrols are, making an attack on any one patrol extremely risky. Other methods include so-called *blue-green* patrolling, mounting overt daylight humanitarian patrols, which go covert at night and hunt specific targets. A reasonable rule of thumb is that one to two thirds of forces should be on patrol at any time, day or night.

- *Be prepared for setbacks.* Setbacks are normal in counterinsurgency, as in every other form of war. The Commander will make mistakes, lose people, or occasionally kill or detain the wrong person. The Commander may fail in building or expanding networks. Local commanders should have the freedom to adjust their posture to local conditions. This creates elasticity that helps survive setbacks.

- *Remember the global audience.* Beware the *scripted enemy*, who plays to a global audience and seeks to defeat US Forces in the court of global public opinion. The Commander counters this by training people to always bear in mind the global audience, assume that everything they say or do will be publicized, and befriend the media.

- *Engage the women, beware the children.* Most insurgent fighters are men. Co-opting neutral or friendly women, through targeted social and economic programs, builds networks of enlightened self-interest that eventually undermine the insurgents. Win the women, and you own the family unit. Own the family, and you take a big step forward in mobilizing the population. But conversely the Commander should stop his people fraternizing with local children: it puts the children, the forces and the mission at risk. Similarly, forces should not throw candies or presents to children: it attracts them to our vehicles, creates crowds the enemy can exploit, and leads to children being run over.

- *Take stock regularly.* The Commander should develop metrics early in the tour and refine them as the operation progresses. They should cover a range of social, informational, military and economic issues. Virtually nothing is a snapshot, it is trends over time that help the Commander track progress in the AO.

- *Exploit a "single narrative".* This is a simple, unifying, easily-expressed story or explanation that organizes people's experience and provides a framework for understanding events. This narrative is often worked out by higher headquarters, but only the Commander has the detailed knowledge to tailor the narrative to local conditions, and generate leverage from it.

- *Local forces should mirror the enemy, not ourselves.* Local indigenous forces need to mirror the enemy's capabilities, and seek to supplant the insurgent's role. This does not mean they should be "irregular" in the sense of being brutal, or outside proper control. Rather, they should move, equip and organize like the insurgents, but have access to the Commander's support, and be under the firm control of their parent societies. Combined with a mobilized population and trusted networks, this allows local forces to "hard-wire"the enemy out of the environment, under top-cover from the Commander.

- *Practice armed civil affairs.* Counterinsurgency is armed social work, an attempt to redress basic social and political problems while being shot at. This makes civil affairs a central counterinsurgency activity, not an afterthought.

- *Small is beautiful.* Another natural tendency is to go for large-scale, mass programs. This is usually a mistake. Often programs succeed because of specific local conditions of which we are unaware, or because their very smallness kept them below the enemy's radar and helped them flourish unmolested.

- *Fight the enemy's strategy, not his forces.* If things proceed well, the insurgents will go over to the offensive because the Commander creates a situation so dangerous to the insurgents, by threatening to displace them from the environment, that they have to attack US Forces and the population to get back into the game. It is best to attack the enemy's strategy: where he is seeking to recapture the allegiance of a segment of the local population, co-opting them against him. Where he is trying to provoke a sectarian conflict, going over to "peace enforcement mode." The permutations are endless but the principle is the same: fight the enemy's strategy, not his forces.

- *Build your own solution - only attack the enemy when he gets in the way.* The Commander should not be distracted, or forced into a series of reactive moves, by a desire to kill or capture the insurgents. The aim should be to implement the *game plan* developed early in the campaign, and then refined through interaction with local partners.

- *Keep extraction plan secret.* The temptation to talk about home becomes almost unbearable toward the end of a tour, but the Commander must protect the specific details of the extraction plan, or the enemy will use this as an opportunity.

- *Whatever else you do, keep the initiative.* In counterinsurgency, the initiative is everything. If the enemy is reacting, then the Commander controls the environment. If we are reacting to the enemy, even while killing or capturing him in large numbers, then he is controlling the environment and we will eventually lose.

COIN Operations guided by Logical Lines of Operations (LLO)

One difference between COIN operations and strictly military/combat missions is the concept of logical lines of operations (LLO). This description of LLOs is summarized from the COIN field manual [3]. Figure 2.16 shows an example of LLOs that might be used in a COIN operation.

- LLOs help commanders identify missions, assign tasks, allocate resources, and assess operations when positional reference to enemy forces has little relevance. LLOs are appropriate for synchronizing operations against enemies that hide among the populace.

- A plan based on LLOs unifies the efforts of joint, interagency, multinational and HN forces toward a common purpose. LLOs are closely related. Successful achievement of the end state requires careful coordination of actions undertaken along all LLOs.

- Each LLO represents a conceptual category along which the HN government and COIN force commander intend to attack the insurgent strategy and establish HN government legitimacy.

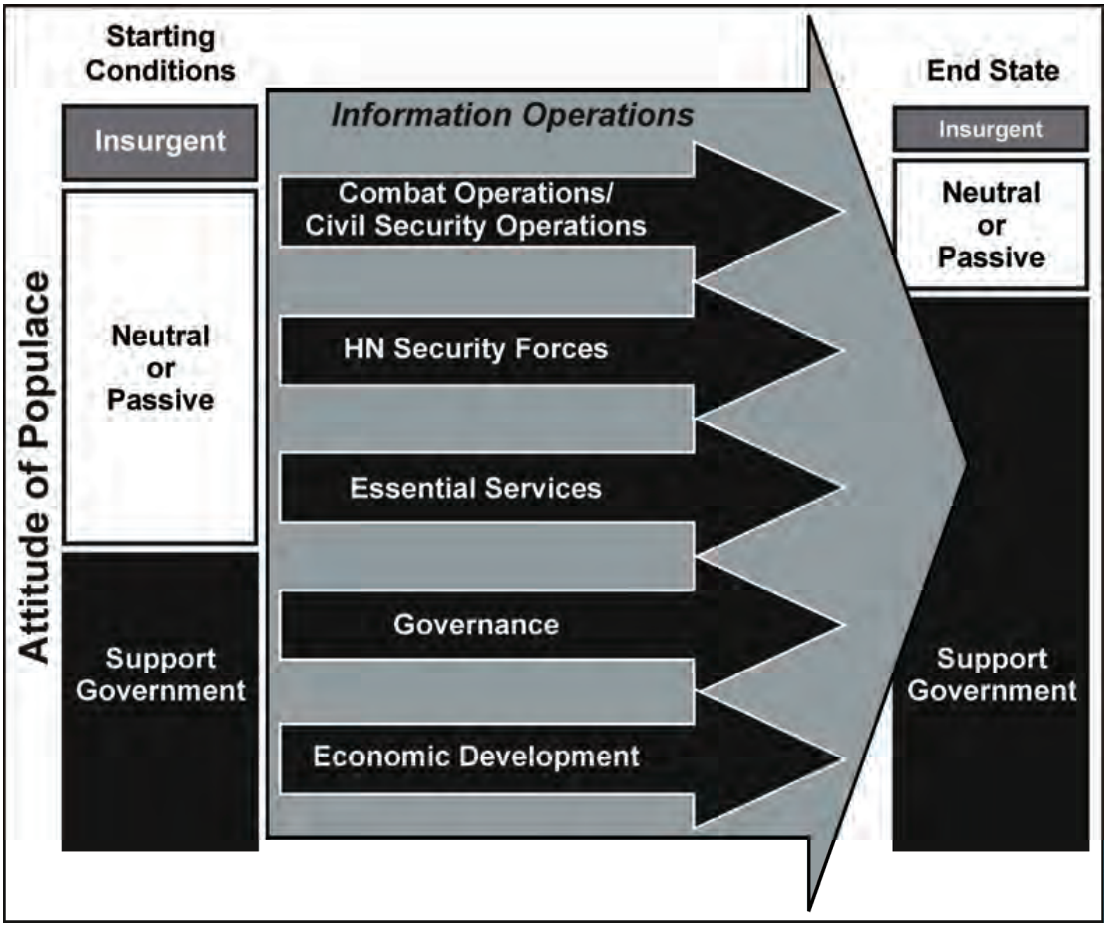

Figure 2.16: Example logical lines of operations that might be used in COIN operations

- There is no list of LLOs that applies in all cases. Commanders select LLOs based on their understanding of the nature of the insurgency and what the COIN force must do to counter it. Commanders designate LLOs that best focus counterinsurgent efforts against the insurgents' subversive strategy.

- Commanders at all echelons can use LLOs. Lower echelon operations are nested within the higher echelon's operational design and LLOs; however, lower echelon operations are conducted based on the operational environment in each unit's area of operations (AO).

- Operations designed using LLOs typically employ an extended, event-driven timeline with short-, mid- and long-term goals. These operations combine the effects of long-term operations such as neutralizing the insurgent infrastructure, with cyclic and short-term events like regular trash collection and attacks against insurgent bases.

Commanders determine which LLOs apply to their AO and how the LLOs connect with and support one another. For example, commanders may conduct offensive and defensive operations to form a shield behind which simultaneous stability operations can maintain a se-

cure environment for the populace. Accomplishing the objectives of combat operations/civil security operations sets the conditions needed to achieve essential services and economic development objectives. When the populace perceives that the environment is safe enough to leave families at home, workers will seek employment or conduct public economic activity. Popular participation in civil and economic life facilitates further provision of essential services and development of greater economic activity. Over time such activities establish an environment that attracts outside capital for further development. Neglecting objectives along one LLO risks creating vulnerable conditions along another that insurgents can exploit. Achieving the desired end state requires linked successes along all LLOs.

Chapter 3

LIME Requirements

3.1 Requirements Introduction

This report describes the requirements for the Language for Investigating Myriad Eventualities being developed for the DARPA Deep Green program. LIME will allow DG components to communicate about military situations, Commander's intent, planned COAs, and possible future states. LIME will continue to be modified as the overall Deep Green program vision matures and the motivational scenarios are evaluated in detail.

Deep Green will build a battle command decision support system that interleaves anticipatory planning with adaptive execution. Deep Green must be capable of addressing the full spectrum of joint and combined arms capabilities available to the modular Brigade Commander, drastically increasing the option and future space. This will allow the Commander to think ahead, identify when a plan is going awry, and help develop alternatives "ahead of real time." The Commander and his support staff are involved in two major asynchronous functions: generating options and making decisions. The goal of the DG program is to create a Commander-driven system to assist the Commander and his support staff in generating options or Courses of Action (COAs). DG will aid in battle command and Commander's visualization by creating technologies that make it easier for the Commander to articulate options to consider, and to anticipate the possible futures that result from those options. This proactive analysis will help predict which possible futures are becoming more likely — before they occur. Given that information, the Commander can make better decisions and focus planning efforts (the generation of future branches and sequels) on where they can be the most useful.

This document presents the requirements on the LIME language that the seedling team has identified from DG design documents, discussions with experts, and scenarios with DG use cases. We begin with a brief glossary of LIME/DG terms, and then move into requirements proper. The requirements discussion is structured top-down: we begin with DG requirements, drill down to individual DG components, and conclude with the requirements on the LIME language itself (Section 3.8). Please note that we have *not* attempted to generate a complete set of requirements for DG. The DG and component requirements are here only to provide structure and traceability for the requirements on LIME. In particular, critical requirements

for the components may be omitted unless they have an impact on representational issues.

3.2 LIME / DEEP GREEN Terms

Abstraction — An abstraction is a concept that generalizes from (subsumes) a more specific concept by omitting some information or detail/precision, usually to retain only information that is relevant for a particular purpose. In the DG context, an abstraction is a compact description of a set of other objects. Abstraction is often applied to states, or to object properties that may vary from state to state. So, an abstraction of fuel level might map from a numeric range to a set of qualitative values (*e.g.* Full, Mostly Full, Half Full, Mostly Empty, Empty)

We can also aggregate states directly, for example over a range of times, or relative to some set of events (all states in which one event has happened and another is pending, for example).

Goal statements tend to be abstractions because they are only partially specified. They may be further abstracted in that they refer to properties not directly represented in any state description. Examples: have we captured the objective, how far are we along the corridor, have I established the food distribution location in this neighborhood.

Action — A behavior taken by a Taskable Object that results in changes to one or more Domain Objects, leading to one or more Outcomes.

Action Instance — A uniquely identified representation of an action in which all of the references to Domain Objects are bound to specific objects. When we refer to an Action, by default we mean an Action Instance. Note that there may exist multiple different Action Instances with exactly the same Domain Object bindings, contained in different COAs or different places within a single COA.

Activity Matrix — A depiction of the activities to be undertaken by units involved in a COA. The activity matrix indicates mission phases, tasks, branches, and coordination points.

Action Class — A representation of an action that is not tied to a specific set of Domain Objects. Might also be called an Action Type or Action Schema.

Affiliation — The relationship between two Taskable Objects, represented as an enumeration such as Hostile, Neutral or Friendly.

AO — Area of Operations (qv.).

Area of Operations (AO) — The area covered by the military operation on which DG is focused.

Attachment — Temporary (although possibly indefinite within the scope of DG) assignment of control authority of a Taskable Object to another Taskable Object that is not its superior command.

COA — Course of Action (qv.).

Commander — Battalion- or Brigade-level Commander *and his staff*. In this document, "Commander" is approximately the same as "user."

Commander's Intent — A statement of the goal the Commander is trying to achieve through execution of some Course of Action (qv.).

Configuration — Specifies a consistent/coherent set of values for one or more object features or sets of objects.

Course of Action (COA) — A COA is a high-level plan of action for one side in a scenario. The COA may include contingencies. During the process of COA construction, the COA may contain (partially specified) Action Schemas, but the final COA submitted to Blitzkrieg must contain only Action Instances.

Current State(s) — The most likely State (or States) in the Futures Graph. DG may never really know what State in the Futures Graph is the best representation of the real world at the current time. Having the Current States be a set allows DG to consider how a COA selection may have different effects depending on which of the possible individual States is actually true, and also allows DG to reason about COAs that may reduce the uncertainty of what State the world is actually in.

Deep Green— The overall system being developed on the Deep Green program as described in DARPA BAA07-56 DG.

DG — Deep Green (qv.).

Direct Support — A unit Alpha is in "direct support" of another unit Beta when supporting Beta is its (unit Alpha's) primary mission or priority. In contrast, "indirect support" indicates a lower priority. In resource-limited situations where support can only be given to one other unit, the "direct support" responsibility will win.

Domain Object — Items needed to model the Operation for DG (*e.g.*, Taskable Objects (qv.) and Environment Objects (qv.)).

Effect — A statement defining the changes to be made to the world state or to processes as a result of a COA's execution.

Environment Object — A Domain Object (qv.) that can only be acted upon, which cannot be tasked (*e.g.* a hill, bridge, etc.). Environment Objects include permanent items (*e.g.* hills) as well as temporary items (*e.g.* a temporary landing zone).

First Class Object — Object which can be used in programs without restriction (when compared to other kinds of objects in the same language). In LIME this means being expressible as an anonymous literal value, being storable in variables, being storable

in data structures, having an intrinsic identity (independent of any given name), being comparable for equality with other objects, being passable as a parameter to a procedure/function, being returnable as the result of a procedure/function, being constructible at runtime [4].

Future — An anticipated future State.

Futures Graph — The projection of a COA or set of COAs, played out against likely other side COAs, into the set of possible future states. Nodes in the Futures Graph capture the State and arcs capture the Actions, processes, and outcomes that led to those States. Arcs are annotated with likelihood information to support assessment of risk, utility, etc.

Indirect Support — A unit Alpha is in "indirect support" of another unit Beta when supporting Beta is its (unit Alpha's) non-primary, lower-priority mission. In contrast, "direct support" indicates a higher/primary priority. A unit may be tasked with indirect support of multiple other units.

LIME — Language for Investigating Myriad Eventualities.

Operation — The overall military activity being addressed by DG.

Outcome — The State entered as a result of executing an Action.

Participant — A member of the Commander's staff that participates in COA construction, *e.g.*, the logistic and intel officers. In this document, participants plus the commander are generally referred to as Commander.

Precondition — A logical statement that defines a set of world states (Configurations) in which the action may be executed.

Note that for temporally-extended actions, the notion of "precondition" is generalized to "condition" because there are some conditions that must hold over the entire duration of the action, not just at the beginning of the action. For example, to move at full speed down a movement corridor, the movement corridor must be unobstructed over the entire duration of the action, not just at the start of the action.

Qualitatively Different —

In this document, we use the term "qualitatively different" informally to mean "different in an interesting way." For example, two states, one in which a military unit has a great deal of fuel on hand, and one in which it has only a little, are not interestingly different if there are no circumstances under which we might want to actually *use* that unit's fuel.

Ideally, a qualitative difference is one that has a significant impact on outcomes. Discerning the effect of differences between states or COAs may be difficult enough that in some cases we fall back on a generic definition.

For example, two COAs may be considered qualitatively different if one is much more aggressive; two Futures Graph states may be qualitatively different if the balance of Red/Blue forces is reversed.

A quantitative difference can be evaluated for whether or not it constitutes a qualitative difference through abstraction: the difference between a full tank of gas and a mostly-full tank is not qualitative (unless you know that the mission is infeasible without a completely full tank!), but the difference between a mostly-full tank and a mostly-empty one is.

Resource — Items that are required, consumed or produced by Taskable Objects.

RoE —Rules of Engagement (qv.).

Rules of Engagement (RoE) — A set of constraints on behavior that govern what kinds of COAs are acceptable and how COAs may be implemented. Together with Commander's Intent, Rules of Engagement define the space of possible COA modifications, if the original COA provided by the Commander is not feasible.

The difference between CI and RoE is that CI is mission-specific and may in fact change during the mission, while RoE is assumed to be more static.

Side — A Taskable Object representing a major player in the current operation including Commander-controlled forces, other friendly forces, coalition, neutrals, enemy, and so on.

State — A collection of Domain Objects distinguished by process states and value of the time-varying properties of all the Domain Objects in the collection. Note that a State may be an *abstract* state, characterized by an *abstraction* of the raw values of properties and processes.

Strength — We explicitly leave the concept of a unit's strength undefined, as this will be mission-dependent. Some representation of strength is needed, so that it can be passed to the simulation.

Taskable Object — A Domain Object (qv.) that can be tasked to carry out some function (*e.g.* a military squadron).

3.3 Global design guidelines and DEEP GREEN Requirements

The following are requirements and guidelines that are relevant to LIME. Many are design guidelines, not requirements *per se*. The DG requirements are the top-level drivers for LIME. Note that not all of the DG requirements are listed, just those that impact the design of LIME. References from the BAA [20] are included by BAA page number.

3.1. DG shall support the Battalion- and Brigade-level Commander.
DG will reside at both the brigade and battalion level and coordinate between those echelons.
BAA Ref: p.7

3.2. DG shall support Commander recognizing and seizing opportunities.
The DG program level objective is look for opportunities to exploit advantages and better prepare for contingencies. By focusing on creating options ahead of the real operation rather than repairing the plan, DG will allow commanders to be proactive instead of reactive in dealing with the enemy.
BAA Ref: p.6

3.3. DG shall represent *all* sides in its scenarios.
DG must not be restricted to two-player, zero-sum conflicts. It must handle situations with multiple sides and account for situations where opposing forces, NGOs, neutrals, and others have objectives which are not the opposite of the Commander's.
BAA Ref: p.5

3.4. Represent constraints presented by military doctrine.
The future state generators and the simulators will utilize the constraints as part of their simulators. Some basic constraints and relationships would be represented in LIME (*e.g.* attachment information) and relationships between actions (preceding, concurrent, etc.) in COAs. The Allen temporal relations [1] are used in JC3IEDM, so then can be used in LIME as well. Note that representing doctrinal constraints may use the same vocabulary as representing other "harder" forms of constraint such as the more-rigid structure of a COA.
BAA Ref: p.12

3.5. Accommodate the "One up, Two down" rule.
Commander thinks one level up, plans two levels down. The Commander also thinks across to adjacent/supporting units. This is a management principle and does not impact LIME. Left as a design guideline.
BAA Ref: p.Guidance provided during SME conversations.

3.6. Tie to battle command systems and simulations.
Capture ground truth (from COP) and bubble up to abstractions. For DG, the COP is assumed to be correct; that is, this effort is not working to improve timeliness or accuracy of the COP.
BAA Ref: p.8

3.7. DG must address outlier situations.
There are important situations where either Red or Blue, or some other side, does something that is not foreseen. Note that this suggests that conventional, minimaxing game solutions will not work for DEEP GREEN.
BAA Ref: p.4

3.8. DEEP GREEN shall support all types of military operations, including Operations Other than War (OOTW), Counterinsurgency (COIN) and Mid Intensity Conflict (MIC).
These are the focus areas defined in the BAA.
BAA Ref: p.7,8

3.9. DEEP GREEN shall support battlefield functional areas, including Intelligence, Maneuver, Fire Support, Mobility/Countermobility/Survivability and Combat Service Support.
The first four are specifically named in the BAA; all five are related to COIN and the "Three-block war" [11] mission areas which are also identified in the BAA.
BAA Ref: p.7

3.10. Provide means to differentiate key terrain from other terrain.
Only certain areas of map (or area) are used for planning; these are usually labeled or otherwise indicated, *e.g.* Checkpoint Alpha, Landing Zone 4, Hill 21, etc.
BAA Ref: p.20

3.11. Provide ability to reason in space and time.
DG will need to reason about the spatial and temporal aspects of the COA.
BAA Ref: p.6

3.12. DG system will project and maintain the Futures Graph.
Support updating Futures Graph to reflect the current situation and set of COA defined by the Commander. DG products from lower levels of the hierarchy should provide further projective information to the higher levels of the hierarchy.
BAA Ref: p.8

3.13. DG shall be able to simultaneously represent multiple alternative futures
Need to be able to represent multiple initial states together with multiple COAs (including multiple COAs for the same side) to generate and track multiple resulting potential futures. (what-if capability).
BAA Ref: p.5

3.14. DG shall be able to represent and reason about COAs including contingent branches.
COAs provided by the Commander may have contingent branches with different sets of actions. Branches will be chosen depending on the state at execution time.
BAA Ref: p.5

3.15. DG shall be extensible.
As new tactics, new equipment, new doctrine, new heuristics become available, DG must extend to deal with them without any significant rewrite. This requirement is here partly because of the whole issue of whether or not the language includes the ontology. Ideally it will not, but formal language and natural language differ on this point, and MSDL is much closer to natural language in this regard.
BAA Ref: p.7

3.4 Blitzkrieg Requirements

4.1. Given a Futures Graph state, expand a selected set of COAs for all sides to generate many qualitatively different possible futures.
This requirement is the key role of Blitzkrieg, achieved presumably using some sort of qualitative simulation.
BAA Ref: p.19,22

4.2. Simulation shall operate within the constraints specified by the input State and COA.
Represent constraints of the physical world. Simulations shall unroll States based on constraints, such as timing and properties of the Actions and Domain Objects.
BAA Ref: p.18

4.3. Leadership qualities shall be considered
A Commander may be bold or conservative, which DG would observe in terms of the risk and reward properties of selected COAs. These tendencies should influence the calculation of probabilities.
BAA Ref: p.18 and Use Case Ref: 2.2.2.

4.4. Blitzkrieg shall consider the potential for unexpected futures.
Designed to generate a broad set of possible futures. These futures should be feasible, even if not expected by human users.
BAA Ref: p.18

3.5 Crystal Ball Requirements

5.1. Identify situations with insufficient information for decisions.
Crystal Ball will compute the likelihood of different outcomes given what is known about the state in which an action is taken, meaning that Crystal Ball is managing the merging of knowledge-equivalent states. The initial state displayed to the Commander can include unknowns as well. Goal is to make it clear what information is missing, not to automatically generate tasks to collect that information.
BAA Ref: p.21

5.2. Support reasoning from the future state backwards and current state forward.
The Commander can pick an interesting future state and then review the decisions and actions that led to that state.
BAA Ref: p.18

5.3. Maintain the Futures Graph.
The Futures Graph has the potential to grow without bound unless it is managed. Such management will include pruning branches that become unlikely and only adding futures that are qualitatively different from futures that are already in the Futures Graph.
BAA Ref: p.22

5.4. Assess degree of match between a Futures Graph state and the relevant Commander's goals/intent.
The ability to measure partial matches to Commander's intent will be important for tracking progress since states may never *exactly* match the stated intent.
BAA Ref: p.22

5.5. Provide a means to assess direction of the operation.
Goal is to allow planning assets to be concentrated in the direction the actual operation is heading. Base this on at least three metrics: value (the quality of the current situation in terms of progress towards achieving the Commander's intent), flexibility (options), and likelihood. These metrics will be updated during the operation. Partial information and partial observability will also factor in to these calculations.
BAA Ref: p.21

5.6. Provide a means to "draw the line" between states.
Wrapped up in abstraction, Crystal Ball needs to determine when a new state is required vs. change within a state.
BAA Ref: p.22

5.7. Provide ability to compare states.
Recognize which states match the current state of affairs as derived from the COP and identify when states are similar to a COA's intended States.
BAA Ref: p.22

5.8. DEEP GREEN shall calculate the likelihood that any given State can be reached from a given starting State via COAs input to DG.
DG system will annotate states in the Futures Graph with likelihood metrics. Both Blitzkrieg and Crystal Ball have a role in determining the likelihood of a state– Blitzkrieg to compute the likelihood given a particular initial state and set of COAs, and Crystal Ball to merge those results into a unified Futures Graph. State transition probabilities will change as the situation evolves. Note that DG's coverage of the COA space (especially regarding actions of the other sides) will not be exhaustive. Consequently, since the full universe of states that might be merged (under some abstraction) into a given state is not known, calculating *real* likelihoods is not possible and depends on the quality (and number) of input COA, and the accuracy of Blitzkrieg and the COP data input to the system.
BAA Ref: p.22

5.9. DG shall calculate the flexibility from a given State in the Futures Graph.
Flexibility is a measure of how many branches from a future lead toward better utility. Generally this will be calculated for the Commander, but enemy flexibility might be an interesting metric as well. Furthermore, different staff roles/responsibilities have different perspectives on the qualities of a COA.
BAA Ref: p.22

5.10. Identify decision points.
DG shall identify when the Commander's intervention is required.
BAA Ref: p.21

5.11. States derived from an existing State shall be merged into the Futures Graph.
In particular, the Commander shall be able to postulate different initial conditions.
BAA Ref: p.22

3.6 Integrator Requirements

The main integrator requirement is to refine and implement LIME during the DG program. Therefore, there are not any specific integrator requirements that impact LIME.

3.7 Commander's Associate Requirements

7.1. Identify unknown information for the Commander.
Commander's Associate needs to be able to highlight information that is currently unknown, but it cannot automatically assign the task of collecting it. The decision to collect further information is left to the Commander.
BAA Ref: p.17

7.2. Present the Futures Graph based on different friendly, enemy and other side COAs.
Different members of the Commander's staff will need to see the Futures Graph from different perspectives.
BAA Ref: p.17

7.3. Permit the Commander to enter and modify COA options.
The DG goal is to have an expressive interface that captures the COA, and then embed machinery in Commander's Associate that translates (expands) the COA into LIME for use across the DG components.
BAA Ref: p.13,14

7.4. The Automated Option Generator shall be guided by the Commander's COA and intent.
The module that generates alternatives from Commander's intent lives in the sketch to plan module. DG should mix up parameters to generate a broader range of future states. This feature is expected in later phases of DG.
BAA Ref: p.24

7.5. Permit the Commander to choose a possible future state, and a path to it.
Commander's Associate shall be able to present pre-analyzed options to the Commander, and allow the Commander to choose options based on predictions, review of the Futures Graph, and other information.
BAA Ref: p.17

7.6. Present decision points to the Commander at different levels of abstraction.
This is a UI-level activity. When decision points are identified by Crystal Ball, Commander's Associate must present them at the levels of abstraction appropriate for each current situation.
BAA Ref: p.16

7.7. Commander's Associate shall be able to interface with an external C3 system (*e.g.*, CPOF) to supply orders in the appropriate form once the Commander selects a COA.
This requirement supports the overall need to integrate with battle command systems. Once the Commander selects a COA, it will get turned into orders supplied to the C3 systems. While it is not necessarily an issue for LIME directly, it does represent an area for potentially reusing LIME representations.
BAA Ref: p.17

7.8. Commander can explore Futures Graph at any level of abstraction.
DG shall allow exploration of the Futures Graph by the Commander. Abstraction is a necessary reasoning tool, both for automated reasoning as well as human-based reasoning. Hence the Commander should be able to examine states at any level of abstraction in the Futures Graph.
BAA Ref: p.16, 17

3.8 LIME Requirements

The following LIME requirements are derived from the DG requirements above and the scenario use cases. The LIME requirements are organized into two levels. The first level is serves an organizational function, with more detailed requirements following at the next level. At the more detailed level, the DG requirement(s) and/or the use case(s) from which the requirement is derived are indicated.

8.1. LIME shall capture the framing information of the Operation.
DG must know about the entities involved in the Operation.

8.1.1. Environment Objects shall include those representing key terrain and features of the AO.
LIME will provide a means to relate terrain items to plan activities to support notions of occupation, dominance, fields of fire, etc.
Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.2, Use Case Ref: 2.4.2, Requirement 3.1, Requirement 3.2, Requirement 3.10

8.1.2. Domain Object relationships shall be supported including affiliation, attachment, operational, and other dependencies.
Relationships, both actual and possible/future, between the domain objects need to be represented so they can be tracked and modified over time. Unit relationships might

encompass Attachment, Direct Support, and Indirect Support. Furthermore, these relationships may change over time. For example, a platoon can be attached to several different taskable objects over the course of the operation. By the same token, these relationships do not necessarily have the same echelon level, *e.g.* not every subordinate unit under a brigade is necessarily a battalion.

Remark: Most ontology languages cannot accommodate changing unit-to-unit relationships. Their decompositions must be stationary because of the inferences to be drawn from them. Since we do not care about subsumption inference, we should avoid such limitations.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.2, Use Case Ref: 2.4.2, Use Case Ref: 2.3.5, Requirement 3.8, Requirement 3.5, Requirement 3.11

8.1.3. Domain Objects represented shall include those of all Sides.
Need to represent all sides involved in the Operation including friendly, enemy, each neutral party, weather, and probably others.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.5, Use Case Ref: 2.4.2, Requirement 3.3, Requirement 3.2

8.1.4. Domain Objects shall be sufficient to cover Brigade and Battalion-level planning.
This is the near-in target for the DEEP GREEN effort and includes coordination between those echelons.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.2.5, Use Case Ref: 2.3.2, Use Case Ref: 2.4.2, Requirement 3.1, Requirement 3.9

8.1.5. Taskable Objects will support calculation of capability to perform a particular action.
Unit capability is calculated based on the mission (or action) that is being considered. For example, offensive against armor, operating range on roads, cross country mobility. Items that allow these to be calculated are the object's properties like amount of tank ammo on board, fuel, maintenance status, etc.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.4.2, Requirement 3.9

8.1.6. Taskable Objects' affiliations shall be represented. Affiliation may be asymmetric or symmetric.
Affiliation and affinity depends on the internal state of each party being related, so it is directional and not bidirectional. For example, Jack may like Jill, but Jill might hate Jack.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.5, Use Case Ref: 2.4.2, Requirement 3.9

8.1.7. Taskable Objects shall indicate the value of its properties, including if that value is unknown.

Objects may have both quantitative and qualitative valued properties that may be used in action descriptions to conditionalize outcomes or to cause Blitzkrieg simulations to behave in appropriate ways. The granularity is dependent upon what DG needs, but the idea is to "bin" things together to facilitate reasoning, for example, different generalized/qualitative types of ammunition may be appropriate (*e.g.*, anti-tank ammunition vs. small-arms ammunition). An important feature of these is that some property values may be unknown. The status of a bridge, for example, might be working, destroyed, or unknown.

Traces Back To: Use Case Ref: 2.3.2, Use Case Ref: 2.3.5, Use Case Ref: 2.3.6, Requirement 5.1, Requirement 7.1

8.1.8. Taskable Objects shall represent the qualities of their leadership.
The qualities (or potential) qualities of the leaders is a factor in assessing potential futures

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.4.2, Requirement 4.3

8.1.9. Domain Object information shall come from external sources.
The precise systems will be identified over the course of the DG program, so the requirement is silent on the source of the domain information. It will include configuration or other setup files (*e.g.* something that defines the characteristics of a patrol unit, how they can behave, what it belongs to, etc.), standard military C3I systems and standard military simulation systems. It will likely cover existing military vocabularies, including AUTL, JC3IEDM, 2525B, although this coverage may be conceptual — it is not necessary in the short term that the full set of enumerations be imported, but rather that LIME's expressive power be adequate for this task.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.2.4, Use Case Ref: 2.2.5. Requirement 3.6, Requirement 5.3

8.2. Domain Objects shall facilitate manipulation and reasoning by DG components.
A key part of DG is reasoning about the objects in the domain, including *potential* configurations of them.

8.2.1. Domain Objects shall be First Class Objects.
These will need to be directly referenced and reasoned about.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.2, Requirement 3.10, Requirement 3.14

8.2.2. LIME representations shall be extensible.
The standards on which DG and LIME are based are still changing, and there will certainly be additional items needed as DG evolves.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.2, Requirement 3.15

8.2.3. Domain Objects shall be hierarchically decomposable
Hierarchy and abstraction is necessary to facilitate many of DG's functions. This requirement is particularly important (or at least self evident) for levels of military hierarchy.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.2, Requirement 3.5

8.2.4. Domain Objects shall optionally permit alternate Configurations.
The Commander may be exploring COA with alternate object configurations (*e.g.* different force structures) that are not yet reflected in reality.

Traces Back To:Use Case Ref: 2.2.4, Use Case Ref: 2.3.2, Use Case Ref: 2.4.2, Requirement 3.2

8.2.5. Domain Objects shall have zero or more static properties.
Some items of a Domain Object should not be modifiable. LIME shall provide constructs for explicitly noting this fact.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.4.2, Requirement 3.11

8.2.6. Domain Objects shall have zero or more time-varying properties.
Some properties of a Domain Object will be modifiable.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.5, Use Case Ref: 2.3.6, Use Case Ref: 2.4.2, Requirement 3.11

8.2.7. Domain Objects shall include dynamically-created location objects.
In planning, it is often necessary to create a named location entity in order to establish control measures, objectives, etc. The JC3IEDM has a rich vocabulary for geometric entities for this purpose.

Traces Back To: Use Case Ref: 2.3.2, Use Case Ref: 2.4.2, Requirement 3.11

8.3. LIME shall represent Actions.

8.3.1. Taskable Objects shall have zero or more potential Actions they can perform.
Each of the Taskable Objects will most likely be able to do many different actions. Almost all of them will do at least one, but there is no good reason to eliminate the possibility that a Taskable Object may not be able to accomplish any actions.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.2, Requirement 5.2, Requirement 5.9

8.3.2. Actions shall specify the functional objective of the task, the Commander's measures of a task, and the situational context of the task.
The functional objective identifies the different means for accomplishing a mission. The measures of a task (both measures of effectiveness and performance) are significant because they provide a mechanism for evaluating effectiveness of a COA related

to Commander's intent and guidance and ,second, when used properly they prevent C2 lockup in a decision making model. The situational context provides additional constraints for the execution of the task.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.2, Requirement 3.9

8.3.3. Action instances shall be First Class Objects.
LIME will have first class objects for activities.

Traces Back To: Use Case Ref: 2.3.3, Requirement 5.2

8.3.4. Actions shall be hierarchically decomposable.
Define Action representation abstractions, and provide for composite actions such as method libraries that provide alternative methods for decomposing complex actions or goals in HTN planners.

Traces Back To: Use Case Ref: 2.3.3, Requirement 5.2

8.3.5. LIME shall support temporally-extended actions.
Most real-world actions take some amount of time to complete. Many of the tasks in the AUTL or 2525b task list have temporal extent: they start, they continue for some time, and then they stop. They may have relevant effects on the world state when they start, when they are complete, or throughout their execution. Their preconditions may be required to hold only at the start, or throughout their execution. In PDDL, temporally-extended actions are referred to as "durative actions."

Traces Back To: Use Case Ref: 2.3.2, Use Case Ref: 2.3.5, Use Case Ref: 2.3.6, Use Case Ref: 2.4.2, Requirement 3.11

8.3.6. Actions shall have preconditions and effects.
LIME must support preconditions and effects, at a minimum to support contingent plans. The primary reason for this requirement is because DG needs to support contingent COAs, but there are other potential uses for preconditions and effects:

- Encode simple domain physics (action ordering and main desired effects).
- Encode simulator-level (detailed) domain physics.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.3, Requirement 5.2

8.3.7. LIME will support reasoning about resources.
Resources will be associated with activities that require, consume, or produce them. LIME may support reasoning about resource types including consumable or replenishable resources, with discrete or continuous values.

Traces Back To: Use Case Ref: 2.2.4, Use Case Ref: 2.3.2, Use Case Ref: 2.3.6, Use Case Ref: 2.4.2, Requirement 4.2

8.3.8. LIME shall distinguish between doctrinal and "physical" constraints.
LIME's action representation shall distinguish between constraints on applicability of

actions (primitive and complex) that are doctrinal guidelines (*e.g.* "should have a 3:1 force ratio for task X") and those constraints that are representations of the physics of the world (*e.g.* "the unit executing task X must have a fires capability."). Note that if LIME does not distinguish between the doctrinal and physical constraints, we run the risk of making it impossible for Commanders using the DEEP GREEN system to be able to do "out of the box" thinking. However, if we do not put the doctrinal constraints in, then we risk having automated planning systems and other decision support systems make suggestions or plan for developments which are grossly unrealistic.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.2, Use Case Ref: 2.3.6, Use Case Ref: 2.4.2, Requirement 3.2, Requirement 3.4

8.3.9. Actions shall specify nominal and off-nominal Outcomes.
Nominal outcome specification will support conventional single-trajectory planning. Off-nominal outcomes provide support for simulation, contingency planning, etc. Actions may have a range of potential outcomes: for example, a troop movement action may succeed or not, use more or less fuel than a formula suggests, etc. Note that different (in particular, uncertain) outcomes may be due either to an action with conditional effects being executed in a state where some information is unknown, or the outcome may be genuinely random (in other words, they may be conditioned on unmodeled state information).

Traces Back To: Use Case Ref: 2.3.3, Use Case Ref: 2.3.4, Use Case Ref: 2.4.2, Requirement 3.2, Requirement 3.7, Requirement 4.4

8.4. LIME shall represent Commander's Intent, COAs, and Plans.
Deep Green shall accept Commander-generated COA as the high-level plans for any and all Sides.

8.4.1. LIME shall support consideration of multiple Sides' COAs.
Deep Green shall model multiple Sides (friendly forces, enemies, neutrals, etc.). Note that weather is also considered a Side in this context with COA generated by the weather officer.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.2.4, Use Case Ref: 2.2.5, Use Case Ref: 2.3.3, Requirement 3.2, Requirement 3.3

8.4.2. LIME shall support consideration of multiple COAs for each Side.
Each Side can have several — and possibly very many — COAs created for it.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.2.5, Use Case Ref: 2.4.2, Requirement 3.2, Requirement 3.3, Requirement 3.13

8.4.3. COAs shall be first-class objects.
LIME must be able to distinguish between different COAs, so it must be able to refer to them as things with identities.

Traces Back To: Use Case Ref: 2.2.5, Use Case Ref: 2.3.2, Requirement 3.13

8.4.4. A COA will consist of a set of actions, specified orderings among them, and optionally contingencies on which different branches (sets of actions) may be executed.
The Commander defines COAs as a series of steps and some synchronization/coordination actions he wants to accomplish, although not necessarily with a fully specified ordering. The detail-adding planner will augment this and provide it to Crystal Ball, as LIME.

Traces Back To: Use Case Ref: 2.3.2, Use Case Ref: 2.3.6, Use Case Ref: 2.4.2, Requirement 3.14

8.4.5. LIME shall express Commander's Intent.
This information is necessary to compute how well a particular path through the Futures Graph satisfies the Commander's Intent.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.2.5, Use Case Ref: 2.3.2, Use Case Ref: 2.4.2, Requirement 3.1, Requirement 5.5, Requirement 7.3, Requirement 7.4

8.4.6. COAs shall specify the Taskable Object or Objects responsible for each Action.
LIME will support identification of the object(s) of each action. The COA representation must contain enough information that an Activity Matrix can be extracted. See FM 101-5.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.3.2, Use Case Ref: 2.3.5, Requirement 3.13, Requirement 7.3

8.5. LIME shall represent the Futures Graph.
DG shall use simulation to project the possible future states resulting from the specified COAs.

8.5.1. States shall be first-class objects.
LIME must be able to explicitly represent a state as a thing, so that it can talk about its property of likelihood, and so that it can enter into relationships with other states (notably outcome relations for the edges in the Futures Graph).

Traces Back To: Use Case Ref: 2.3.3, Use Case Ref: 2.3.6, Requirement 5.3, Requirement 5.8

8.5.2. LIME shall support direct representation of objects with properties.
The LIME representations will *not* use a purely propositional representation for state as in PDDL. This will also provide effects constructs like *update*, *increment* and *decrement* to better mesh with a property-based state representation.

Traces Back To: Use Case Ref: 2.3.2, Use Case Ref: 2.3.4, Use Case Ref: 2.4.2, Requirement 3.12, Requirement 7.2

8.5.3. LIME shall represent DG system states and state structures.
LIME must represent the states that comprise the Futures Graph, and capture relationships between those states.

Traces Back To: Use Case Ref: 2.3.3, Use Case Ref: 2.3.4, Requirement 3.12, Requirement 5.3

8.5.4. Every state shall have zero or more distinct subsequent states.
This requirement supports incorporation of, perhaps differing, results from multiple runs of Blitzkrieg. Differing results may be related to likelihood or the set of possible outcomes of a particular action, or may be a reachback/remote simulation and have different time cycles. Includes location and state of domain objects.

Traces Back To: Use Case Ref: 2.3.2, Use Case Ref: 2.3.3, Requirement 3.12, Requirement 4.4, Requirement 5.2, Requirement 5.8

8.5.5. States shall support merging at different levels of abstraction.
Provide a means to determine state, sub-state relationships. LIME must be able to explicitly represent states as being subsumed by more abstract states. Many low-level states need to map into a smaller set of (abstract) states. Support binning together different states that are qualitatively similar.

Traces Back To: Use Case Ref: 2.3.2, Use Case Ref: 2.3.4, Requirement 4.1, Requirement 5.3, Requirement 5.6, Requirement 7.3

8.5.6. States shall support comparison with other states to determine similarity, which may depend on the COA/Intent context.
Represent information so DG components can make inferences on similarity of the states.

Traces Back To: Use Case Ref: 2.2.4, Use Case Ref: 2.2.5, Use Case Ref: 2.3.2, Requirement 5.3, Requirement 5.4, Requirement 5.7

8.5.7. States shall share a common clock for tracking time.
DG needs a clock if only to synchronize with the COP. For example, DG needs to know how old data is. The system also needs metric time to be able to reason about durations: how long it will take for units to travel, etc. We leave open for now the issue of whether we may be able to get away without absolute time.

Traces Back To: Use Case Ref: 2.2.2, Use Case Ref: 2.2.5, Use Case Ref: 2.3.2, Requirement 3.11

8.5.8. LIME shall support aggregation based on events and time.
It is important that LIME allows abstraction over events as well as over time so that (abstract) states can be annotated with the currently active COAs and position in the process of interpreting them.

Traces Back To: Use Case Ref: 2.3.2, Use Case Ref: 2.3.6, Requirement 3.11, Requirement 7.2

8.5.9. Arcs between states in the Futures Graph shall be first class objects.
LIME must support examination and reasoning about the Arcs between states. For example, we may need to determine the underlying assumptions that lead from a given state to another.

Traces Back To: Use Case Ref: 2.3.3, Requirement 3.12, Requirement 5.9

8.6. LIME shall facilitate internal and external communications.

8.6.1. LIME shall encode communications from Crystal Ball to Blitzkrieg.

Crystal Ball will use LIME to tell Blitzkrieg the states and COAs in the Futures Graph which it should investigate.

Traces Back To: Use Case Ref: 2.3.3, Use Case Ref: 2.4.2, Requirement 3.13, Requirement 4.2

8.6.2. LIME shall encode Blitzkrieg state descriptions.

LIME will allow Crystal Ball to dump enough of a state description into Blitzkrieg to allow Blitzkrieg to begin a simulation. We do not know precisely what sort of state descriptions Blitzkrieg will require as input. However, we may conjecture that Blitzkrieg will require some abstraction of the full state description, so we should assume that this communication may contain up to as much information as required to actuate the OneSAF simulation, since the OneSAF simulation must be at least as concrete as the Blitzkrieg simulation.

Traces Back To: Use Case Ref: 2.3.3, Use Case Ref: 2.3.4, Use Case Ref: 2.3.6, Requirement 4.1

8.6.3. LIME shall encode communication of Futures Graph subgraphs from Blitzkrieg to Crystal Ball, including abstract states.

The result of Blitzkrieg computations are subgraphs of a Futures Graph. These subgraphs must be sent to Crystal Ball, which will then fuse them into the main Futures Graph. Blitzkrieg will generate graphs of states and state transitions where the states are qualitatively different abstract states, so the encoding must allow abstract states.

Traces Back To: Use Case Ref: 2.3.3, Use Case Ref: 2.3.6, Requirement 3.13, Requirement 4.1,Requirement 5.11

8.6.4. LIME shall encode COA attributes in states from Blitzkrieg.

In order to evaluate states and fuse them into the Futures Graph, Crystal Ball must know the COAs used by all sides, plus the state of execution of those COAs, in those states.

Only Blitzkrieg will know how far the friendly and other-sides' COAs have progressed in any particular state, but that information is critical to the correct function of the full Deep Green system. Accordingly, that information must be provided in the subgraphs represented by LIME.

Traces Back To: Use Case Ref: 2.3.3, Use Case Ref: 2.3.4, Requirement 5.5

8.6.5. LIME shall encode Commander's Associate to C3I systems communications

Based on the BAA, the external world will be simulated by the OneSAF SIS, indicating

that the communication to the COA execution component must be of sufficient power to permit the actuation of a OneSAF simulation.

Traces Back To: Use Case Ref: 2.3.6, Use Case Ref: 2.4.2, Requirement 3.6, Requirement 7.7

8.6.6. LIME shall encode C3I updates to Crystal Ball.
This requirement is somewhat controversial, and may be dropped. The COA execution updates must be accepted and used by Crystal Ball to recognize the state or states in the Futures Graph that correspond to the current state of execution.

These state updates might be expressed as abstract (partial) state specifications in LIME, or might simply be accepted in some form more directly out of the Execution (C3I) systems. In either case, we need some mechanism for lining up the state updates from the C3I systems with possibly-abstract state descriptions in LIME.[1]

Traces Back To: Use Case Ref: 2.2.4, Use Case Ref: 2.4.2, Requirement 3.6, Requirement 5.3, Requirement 5.5

8.6.7. LIME shall encode Crystal Ball to Commander's Associate communications.
Crystal Ball must be able to provide the Commander's Associate with a Futures Graph for presentation by Sketch to Decide, and for further development by Sketch-to-Plan.

Traces Back To: Use Case Ref: 2.3.4, Use Case Ref: 2.4.2, Requirement 5.1, Requirement 5.5, Requirement 5.10, Requirement 7.1, Requirement 7.2, Requirement 7.6, Requirement 7.8

8.6.8. LIME shall encode Commander's Associate to Crystal Ball communications.
Commander's Associate must be able to indicate that the Commander intends to pursue particular options at particular nodes in the Futures Graph. Those options/COAs must be transmitted to the Crystal Ball component for further investigation by wargaming with Blitzkrieg.

Traces Back To: Use Case Ref: 2.3.2, Use Case Ref: 2.4.2, Requirement 4.1, Requirement 7.3, Requirement 7.5

[1] To be fully precise, the state descriptions need not be encodable directly in LIME; the state descriptions must be compatible with the LIME data model, but may be encoded in arbitrary data structures. The issues we discuss here concern the meaning of LIME expressions, rather than their implementation as data structures, implicit or explicit.

Chapter 4

LIME Design Recommendations

4.1 LIME Design

LIME will serve as a domain modeling and inter-module communication language within DEEP GREEN. Given the representational requirements of this task, LIME needs to be able to encompass uncertain action outcomes, limited information and contingent planning and execution, reasoning about resources, state abstraction, trajectory constraints, temporally extended and overlapping actions, adversary planning and action, dynamically created and destroyed objects and resources, and task decomposition, among other things. The goal of the seedling is to provide a head start to the DG program by defining language requirements and a set of design recommendations.

The conclusions we have reached over the course of this program point to many features that LIME must have, independent of how it is implemented. The following describes those features and provides a context for approaching them based on existing standards and technologies, identifying what can be taken directly from existing work, what needs to be modified, and what is entirely new. The text below indicates the LIME requirements that apply to the design discussion.

To the extent possible, the recommendations are implementation-neutral, because there is more than one way to approach these problems and, more pragmatically, the DG program has not yet started, so exactly how the software gets implemented depends on who gets the contract. For example, we have used the term "class" throughout this document. However, the classes in this document need not necessarily correspond directly to programming language classes or types in the implementation. Further, we have felt free to employ multiple inheritance in this document. Multiple inheritance might be provided *in* the programming language chosen to implement LIME or it might not be directly available and have to be implemented *using* the programming language. Similarly, we assume that all LIME representations are extensible to grow with the program, but the exact mechanism (e.g., dynamically at runtime or requiring a reboot) for that extensibility is left open. Finally, the properties of the classes can be dynamic or static, but exactly which ones or how that is encoded is also left for implementation. **Requirement 8.2.2.**, **Requirement 8.2.3.**, **Requirement 8.2.5.**, **Requirement 8.2.6.**

In order to minimize the difficulties of integrating DG with existing and planned simulation and C3I systems, we have striven to make LIME compatible with data interchange standards wherever possible in part to facilitate populating relevant aspects from external sources. In particular, where possible, we have taken our data model elements from the Military Scenario Definition Language (MSDL) [13, 14]. We have drawn from the 2005 version [13] as well as from the newer 2007 version [14] because the older draft included Course of Action elements that were not approved for the 2007 standard. Where MSDL constructs have been available but not sufficient to meet the needs of DG, we have proposed extended representations based on the existing MSDL elements. **Requirement 8.1.9.**, **Requirement 8.6.6.**, **Requirement 8.6.5.**

Where MSDL elements have *not* been available, we have fallen back on the Joint Command Control Communications Information Exchange Data Model (JC3IEDM) (JC3IEDM). Note that the JC3IEDM is also the framing data model that underlies MSDL: MSDL uses particular JC3IEDM elements, assembling them into a data model sufficient to initialize simulation systems. Similarly, Coalition-Battle Management Language (CBML) selects a subset of JC3IEDM elements and relations to meet the needs of integrating C3I systems (the CBML and MSDL standardization efforts are being coordinated to ensure that the two data models will be compatible). By and large, we have tried to give our LIME classes and properties the same names as the corresponding MSDL and JC3IEDM classes.[1] **Requirement 8.1.9.**, **Requirement 8.6.6.**, **Requirement 8.6.5.**

Note that we intend LIME to provide the shared data model for the DG system and expect that there will be an internal, direct implementation of LIME as a network of data structures. Neither MSDL nor the JC3IEDM provides this implementation, nor would it necessarily be appropriate for them to do so. These are data models and interchange standards,[2] not internal representations. What is important is that there be a straightforward translation of important subsets of LIME (e.g., the simulation configuration subset, the COA subset, etc.) into these data interchange formats. **Requirement 8.1.9.**, **Requirement 8.6.5.**, **Requirement 8.6.6.**

As an example of the distinction between data interchange and internal data structure needs, in many cases MSDL objects have complex objects that group together related information (e.g., the Relations of a Unit group together many different properties of the unit, such as their attachment, support, and affiliation). This makes the representation easier to factor into a network of XML entities. However, for internal data use, it is more convenient to directly attach these properties to the data structure representing the unit (so that we can simply talk about the affiliation of the unit, without dereferencing an intermediate Relations object). **Requirement 8.2.1.**, **Requirement 8.2.3.**

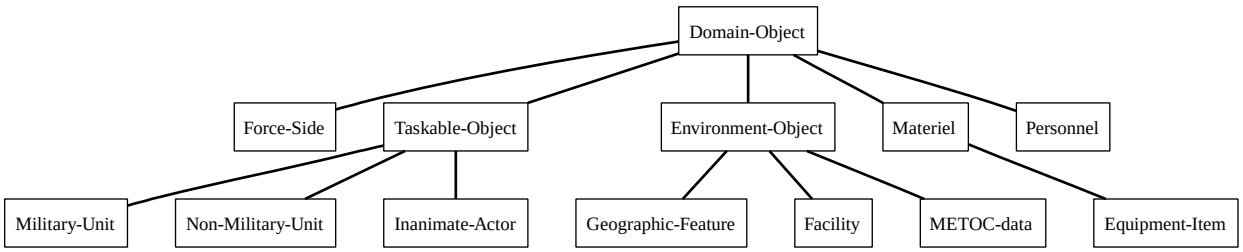

Figure 4.1: View of the top-level class hierarchy of Domain Objects.

4.2 Domain Objects

DOMAIN-OBJECTs are the things that make up the representation of world state that will populate the DG Futures Graph. To represent the military operations that will be controlled through DG, we must represent the *context* of the operation (the ENVIRONMENT-OBJECTs), the sides active in the Area of Operation (AO) (the FORCE-SIDEs), the assets they command (the TASKABLE-OBJECTs), and the resources at the disposal of these assets (MATERIEL and PERSONNEL). We show the top levels of the DOMAIN-OBJECT sub-class hierarchy in Figure 4.1. We discuss each of these classes in this section, and their key properties. Subsequent sections will discuss how they may be actuated by Commander's Intent and COAs, how their attributes are bundled into states to place in the Futures Graph, and so forth. **Requirement 8.1.4.**

4.2.1 Forces and Sides

The Forces are component forces that owe allegiance to a given Side. In our specification of the FORCE-SIDE class, we closely follow the draft 2007 Military Scenario Definition Language (MSDL) specification [14]. Significant attributes of a FORCE-SIDE include optional *MilitaryService*, *CountryCode* and *AllegianceHandle* properties. In addition to this information, these objects can have *Associations*. *Associations* are data structures which provide information about the affiliations between different FORCE-SIDEs. See the MSDL specification, section 6.4.1. **Requirement 8.1.2.**, **Requirement 8.1.3.**, **Requirement 8.4.1.**

4.2.2 Taskable-Object

The TASKABLE-OBJECT is defined in the requirements as "A Domain Object that can be tasked by an Agent to carry out some action (*e.g.*, a military squadron)." It is worth noting, however, that LIME has an expansive view of what may carry out an action. In particular, in addition to military and non-military units, LIME also provides INANIMATE-ACTOR. This pseudo-agent is provided to model the actions of nature. So, for example, we may have weather agents that have their own courses of action, and can carry out pseudo-actions (really environment processes) such as Fog, or Rain. **Requirement 8.1.1.**, **Requirement 8.4.6.**, **Requirement 8.3.1.**

[1]We have typically modified the MSDL CamelCaseNamingConvention to the JC3IEDM-style Hyphenated-naming-convention in the process, however.

[2]MSDL is implemented as an XML schema, and there is an XML schema and a database schema for the JC3IEDM.

One of the most important properties of each TASKABLE-OBJECT is its *current-activities*. In general, this will be a set-valued property. The set value arises for two reasons: first, because some actions can be performed concurrently by the same TASKABLE-OBJECT, and second, because states may extend temporally over a period of time in which a sequence of actions are carried out by the same TASKABLE-OBJECT. In order to accurately reason about states in the Futures Graph, and in order to be able to restart Blitzkrieg from arbitrary Futures Graph states, it is necessary to know what the various TASKABLE-OBJECTS are doing in each state. **Requirement 8.2.1.**, **Requirement 8.3.1.**, **Requirement 8.5.2.**, **Requirement 8.6.1.**, **Requirement 8.6.2.**

MILITARY-UNITS will be the focus of the Phase I DEEP GREEN effort. For them, we propose to adopt much of the MSDL Unit entity. In particular, we envisage adopting the existing attributes of *Name*, *SymbolIdentifier*, *UnitSymbolModifiers*, *CommunicationNetInstances* (this may not be much used in Phase I) and *Relations*. We also expect that the original MSDL Disposition and FormationPosition elements would be "dereferenced," so that in LIME one could directly speak about state attributes such as the unit's *location*.[3] Note that the *Relations* will provide information about the hierarchical structure of the military units, affiliation with Forces and Sides, attachment relations, and support relationships. **Requirement 8.1.2.**, **Requirement 8.1.6.**, **Requirement 8.2.3.**

MSDL will have to be extended because it provides only information needed to describe the *start* of a scenario. As part of the DEEP GREEN program, we will have to identify a set of further MILITARY-UNIT properties sufficient to characterize the operational status of units for the purposes of the Commander and for Blitzkrieg simulation. Keeping track of key materiel resources and personnel that are able to function will be of particular interest in LIME. See the discussion of MATERIEL and PERSONNEL, below. Additional attributes that we know must be added are information about the *morale*, *leadership*, and *training* of MILITARY-UNITS. **Requirement 8.1.5.**, **Requirement 8.1.8.**, **Requirement 8.3.7.**

NONMILITARY-UNITS will be treated similarly to military units, but additional subtleties arise; these will have to be worked out in connection with the development of COIN and OOTW simulations by the Blitzkrieg contractor and the OneSAF PM. In particular, if we were to treat NONMILITARY-UNITS, representing groups like the refugees in our Binni scenario, similarly to MILITARY-UNITS, only under the command of a side other than red or blue, then we would lose the ability to simulate influence on them. This would be unacceptable, for example, in COIN simulations where it is important to model the effects of civil affairs (*e.g.*, infrastructure improvements and diplomacy efforts to build relationships) on the population. These scenarios require DEEP GREEN to be able to sometimes treat NONMILITARY-UNITS like units, and sometimes like passive entities to be manipulated by Red and Blue Sides, especially using means other than force. **Requirement 8.1.3.**, **Requirement 8.4.1.**, **Requirement 8.4.2.**

[3] See discussion of the relation to MSDL in Section 4.1.

4.2.3 Environment Objects

ENVIRONMENT-OBJECTs cover the framing data about the area of operation (AO). Two subclasses, GEOGRAPHIC-FEATURE and FACILITY cover map features, and a third, METOC-DATA covers the meteorological and oceanographic data (for Phase I, we anticipate that only meteorological data will be included). **Requirement 8.1.1.**, **Requirement 8.1.4.**

The FACILITY LIME class corresponds directly to the JC3IEDM entity of the same name. We expect to diverge from the JC3IEDM slightly in not using the JC3IEDM FACILITY-TYPE, but rather directly have a set of subclasses corresponding to facility types.[4] Some FACILITYs [sic] will also be resources. One possibly unusual feature of FACILITY that DEEP GREEN will require is the ability to talk about FACILITYs *that do not yet exist*. For example, in the Binni scenario, we see that the companies that escort the survey teams must build LANDING-ZONEs and then must guard them. This means that COAs must be able to talk about these facilities *before* they actually exist, requiring that we must be able to mark them as being only *potential*. Note that this representational need is served in the MIL-STD 2525B, which provides for the use of dotted line for future entities. **Requirement 8.2.7.**, **Requirement 8.3.7.**

GEOGRAPHIC-FEATUREs represent features of the AO that are not man-made. These include entities like rivers, hills, and other features. It is critical that we be able to represent these as identifiable things (rather than things that are simply *implicit* in map data) so that the Commander can refer to them in natural ways, and to enable qualitative simulation and reasoning about military engagements. **Requirement 8.1.3.**, **Requirement 8.1.4.**

The METOC-DATA class is directly modeled on the MSDL entity ScenarioWeather. See section 6.3.3 of the MSDL specification. Note also that the MSDL entity ScenarioWeather is compatible with the JC3IEDM METEOROLOGIC-FEATURE. See JC3IEDM section 5.5.4. We expect that Phase I will need only a relatively small subset of the information that ScenarioWeather can provide. Initially, we expect Precipitation, Light, and Visibility to be the most important items to use here. The state of the METOC-DATA object will be manipulated by pseudo-actions carried out by the INANIMATE-ACTOR(s). **Requirement 8.1.1.**, **Requirement 8.1.3.**, **Requirement 8.4.1.**, **Requirement 8.4.2.**

4.2.4 Materiel

Instances of MATERIEL will typically be attached to units (and some other DOMAIN-OBJECTs) to aid in tracking the holding object's capabilities to perform tasks and provide resources and support. We will primarily be interested in instances of MATERIEL that are also subclasses of RESOURCE. See Section 4.3 below. A particularly important subclass of MATERIEL is EQUIPMENT-ITEM, which we adopt from MSDL. MSDL has an EquipmentItem object that corresponds to the pools of equipment attached to a unit. This object provides

[4] Note that DG can comply with the JC3IEDM simply by writing a pair of FACILITY and related FACILITY-TYPE to correspond to a LIME object of a particular FACILITY subclass.

properties such as Name and Quantity.[5] **Requirement 8.1.5.**, **Requirement 8.5.2.**, **Requirement 8.1.7.**

4.2.5 Personnel

To track the state of units in an operation, DEEP GREEN must be able to track the number of their operational personnel. The PERSONNEL class will carry information like that of EQUIPMENT-ITEM. It will have static attributes describing the kind of personnel in a particular pool. Depending on what sort of operations DEEP GREEN focuses on, and at what command echelon, we need to make different choices about how many different PERSONNEL pools will need to be tracked for a particular unit. For example, in COIN it will be important to separately track personnel that are capable of performing Civil Affairs activities; in MIC it may be possible to simplify and treat military units as homogeneous groups of personnel. We expect that PERSONNEL will use data structures like the JC3IEDM PERSON-TYPE to characterize personnel pools. In some cases, as in the Binni scenario, in connection with the Survey Teams, or in COIN to track influential local leaders, we may even need to track individual personnel. **Requirement 8.1.4.**, **Requirement 8.1.5.**, **Requirement 8.3.7.**

4.2.6 State of domain objects

So far, we have simply listed the properties of classes, in much the same way they are presented in data models like MSDL and the JC3IEDM. There is one important difference from conventional object models that we should bear firmly in mind, and that is that when using these properties, we will always be considering not just what is the property-value for an object, but what is the property-value for an object *in a given state* (see Section 4.5). Most APIs for state in DEEP GREEN's LIME models will be functions of states and objects, not just objects. This is somewhat akin to the way the JC3IEDM allows one to talk about the type of an object *according to a particular REPORTING-DATA*. The main difference is that the JC3IEDM allows only limited attributes to be relative to REPORTING-DATA, whereas the role of state in LIME is pervasive. We will address this further in Section 4.5. **Requirement 8.2.4.**, **Requirement 8.5.3. Requirement 8.5.2.**

4.3 Resources

This is a discussion of what kinds of things might be modeled as resources, and what kinds of resource models are required.

4.3.1 Domain Objects Modeled as Resources

There are several qualitatively different features of the domain that we may want to model as resources. A preliminary list includes **Requirement 8.3.7.**:

- Environment Objects representing Terrain features that have limited ability to support military operations, such as a movement corridor or landing zone.

[5] This MSDL property is actually indirectly mediated through an EquipmentSymbolModifiers entity. We expect to attach these directly to the MATERIAL entry.

- Environment Objects representing man-made features, such as bridges.
- Taskable Objects representing units at some echelon, such that each Taskable Object is assigned a single job. Note that the echelon at which this reasoning is done may vary: in some cases, an entire battalion may be assigned some task (and so is unavailable for any other task); in others, individual platoons or smaller units may each be assigned separate tasks.
- Assets associated with Taskable Objects. Examples may include fuel, ammunition, personnel, transport vehicles, artillery, communication or surveillance assets.
- Domain Objects representing equipment or material (*e.g.*, fuel at a depot) that may not be associated with any specific Taskable Object.

4.3.2 Examples from JC3IEDM

There are several places in the JC3IEDM containing information relevant to resource models. MATERIEL-TYPE includes EQUIPMENT-TYPE, which has subtypes enumerating classes of equipment that can be treated as *releasable* resources (see below for what that means). The classes of equipment listed include aircraft, electronic equipment, engineering equipment, land weapons, NBC equipment, railcars, vessels, vehicles, and miscellaneous equipment. Another subtype of MATERIEL-TYPE is CONSUMABLE-MATERIEL-TYPE, with subtypes corresponding to some kinds of *consumable* resources (again, see below). The listed classes of consumable resources include ammunition, and nuclear, biological, and chemical agents. Finally, geographic features are defined in the JC3IEDM in ways that are relevant to resource models. There are two relevant subtypes of OBJECT: FACILITY and FEATURE. FACILITY includes things like bridges and (man-made) military obstacles. FEATURE includes GEOGRAPHIC-FEATURE and CONTROL-FEATURE, which includes things like routes. See Section 4.4 for additional examples of resource-relevant geographic features. **Requirement 8.3.7.**, **Requirement 8.1.5.**

4.3.3 Resource Models

A simple set of generic resource types is well-known in the planning and scheduling community. While there are sometimes minor differences in how to draw the type hierarchy, there is broad agreement on the basic properties required **Requirement 8.3.7.**, **Requirement 8.1.5. Requirement 8.2.5.**, **Requirement 8.2.6.**:

Discrete or continuous? – Is the resource value a count of available items (tools, bullets, trucks), or a value representing a quantity of something (power, fuel, comm. bandwidth)?

Unary resources, representing a single instance of the thing being used, are a useful special case of discrete resources.

Releasable or consumable? – Is resource usage temporary, during the extent of the activity using the resource (using a tool, assigning a unit), or is the resource consumed by the activity, leading to a permanent reduction (fuel usage, force attrition)?

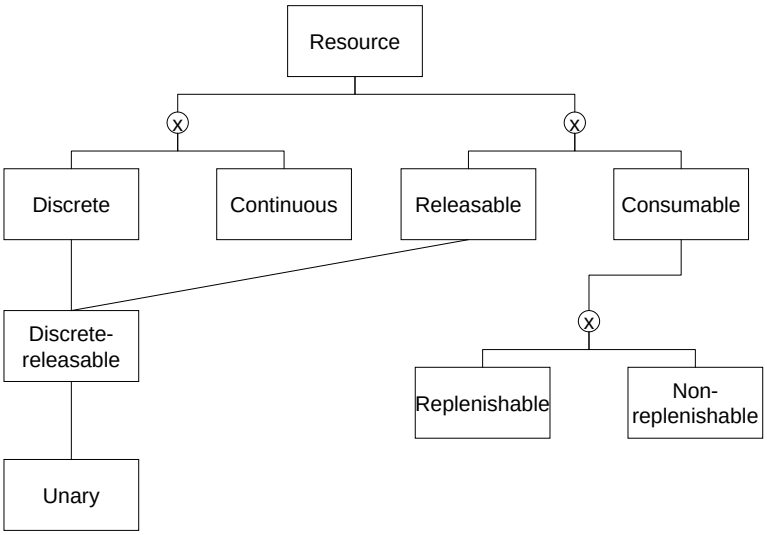

Figure 4.2: Resource type hierarchy.

Some consumable resources can be *replenished*, where the level of the resource available is increased by some activity (a fuel tanker arrives to top off storage tanks).

Similarly, some releasable resources may have changing availability over time (equipment comes out of maintenance, units achieve combat readiness, one force temporarily loans another personnel or materiel).

Instantaneous or continuous effects? – For a consumable resource, resource usage or replenishment for an activity may be modeled as happening instantaneously, for example at the start or end of the activity, or as a continuous process over the duration of the activity.

Continuous effects are significantly more difficult to handle correctly in implementation, and can be approximated (bounded above and below) by a model using instantaneous effects. Consequently, resource models including continuous effects should be employed only when they are required.

Other types of resources have been defined, especially where some special model was required for a particular application, but some combination of these characteristics should suffice for the resource modeling needs of DG. If not, additional resource types may be added.
Requirement 8.2.2.

We propose the taxonomy of resource types given in Figure 4.2, based on the above abstract taxonomy. Note that the circled x's indicate mutually-exclusive and exhaustive partitioning of the parent class, so that resources must be either discrete or continuous and

must be releasable or consumable. Note also that in the interests of space, we have not filled out the full type lattice (e.g., we have drawn no node for Discrete-Replenishable). **Requirement 8.3.7.**

4.3.4 Modeling Considerations

There are several modeling complications to keep in mind with regard to resources. One is their previously-mentioned dynamic nature: resources may be available only at certain times. Worse, the Domain Object which is being treated as a resource may itself be dynamically created and destroyed (a Landing Zone, for example). More generally, resource usage and availability can only be defined with respect to some state. So, we talk about the ammunition remaining for a given unit *in a given state* (this does include the initial state). **Requirement 8.2.6.**, **Requirement 8.5.3.**, **Requirement 8.5.2.**

Finally, abstraction in the DG state space will affect the resource representation as well. Depending on the level of abstraction present in a given state, an ammunition level may be represented differently. For most (non-unary) resources, we expect that the representation will be uniform (i.e., numeric) and what will change is the granularity or precision of the value. For instance, at different levels of abstraction a given value may be represented as $\pm 1\%$, $\pm 5\%$, or $\pm 10\%$. **Requirement 8.5.5.**, **Requirement 8.5.8.**

It is possible that the process of modeling a domain in detail will make it clear that using qualitatively different representations for resource values at different levels of abstraction is the right thing to do, but for now, variable precision in a uniformly numeric value seems the more promising approach. **Requirement 8.5.8.**, **Requirement 8.1.7.**

4.4 Terrain and Location

Terrain and location are concepts ingrained in military planning activities. Consequently, such concepts appear in nearly every military standard for planning ever created. Here we present a discussion of these concepts in the context (and scope) of LIME. Much of the discussion centers around MIL-STD 2525b and the JC3IEDM because the former is how the Commander (or DG user) will refer to terrain when interacting with the Commander's Associate (and hence in COAs) and the latter is the exchange model between the command and control systems (*e.g.*, OneSAF) and DG. Therefore, much of the related terrain and location information will be communicated to DG and exported from DG within these standards. MSDL has comparatively little to say about terrain, because the terrain is typically not communicated to simulations through MSDL. Instead, the terrain is incorporated in maps, and MSDL scenario definitions simply indicate which maps to use. **Requirement 8.1.1.**, **Requirement 8.1.9.**, **Requirement 8.6.6.**, **Requirement 8.6.5.**

4.4.1 Terrain

Terrain falls into three basic categories **Requirement 8.1.1.**:

- Object Specific Terrain: Terrain that need only be referenced in association with a specific DOMAIN-OBJECT, *e.g.*, location of a unit or locations encapsulated within tasks, like orientation of fires, ambush target location, etc.

- Operational Terrain: Includes battlefield's effects [6] and control measures. Battlefield effects are terrain abstractions that are specified without specific regard to any particular COA for any of the sides, *e.g.*, engagement areas, battle positions, infiltration lanes, avenues of approach, and key terrain. Control Measures are terrain that is important for coordination and control of one or more particular COAs, *e.g.*, unit boundaries, phase lines, coordinating points, axes of advance, and objectives
- Regular Terrain: The rest of the map including locations of rivers, roads, cities, forest, and so on.

These categories are described in the following subsections.

Object Specific Terrain

Object specific terrain are locations that are logically scoped to fall within a single step of the COA or within a single LIME object. For example, in indicating that UnitX should Ambush the enemy at a particular location, the Commander will draw the 2525b symbol for AMBUSH (2.X.2.6.1.1) and indicate the ambush position and the target position. Within Commander's Associate this will be turned into a ACTION (see Section 4.6) with the appropriate locations initialized within it. Other objects do not need to specifically reference these ambush points and if they did, they would either do it directly from the ACTION instance or, more generally, the Commander could indicate the location using another symbol (e.g., coordination point). Arguably, one could have a specific ambush-terrain object, but if this were the case there would be an ambush-action object *and* an ambush-terrain object, with a complex question to answer regarding the relationship between them. Instead, applying Occam's razor, the concepts are merged together and ACTIONs can have LOCATIONs. This is described more in Section 4.6 and Section 4.4.2. **Requirement 8.4.3.**, **Requirement 8.2.7.**

Operational Terrain

Control measures and battlefield effects are represented similarly in LIME (although they are different concepts in military planning) as OPERATIONAL-TERRAIN because in each case they indicate terrain objects that need to be referenced outside the scope of other DOMAIN-OBJECTs. For the purposes of this discussion, operational terrain marks a boundary at which a property holds or an action is taken, it may be a point, line or area. Examples are:

- Infiltration lane (marks the intended area for penetration of an enemy defense)
- Limit of Advance (marks the forward-most line to which friendly forces may proceed)
- Waypoint (a location along a route)

The difference between this and the Object Specific Terrain described above is that these are used for coordination and control across time or multiple actions in a COA or across multiple COAs, whereas the Object Specific Terrain is encapsulated within a single object or COA step. **Requirement 8.4.5.**

[6] "Battlefield's effects" is a doctrinal term with a specific technical meaning for military decision making but with connotations that may be quite misleading for computer scientists.

Figure 4.3: An example Engagement Area from 2525b that shows composition of several different key terrain objects.

The OPERATIONAL-TERRAIN can be derived from 2525b, specifically the C2 and General Maneuver (2.X.2[7]), Mobility/Survivability (2.X.3), Fire Support (2.X.4), and Combat Service Support (2.X.5) categories of 2525b. In practice, there will not be a direct mapping from every one of these items to OPERATIONAL-TERRAIN classes because some of the 2525b symbols in these categories will be incorporated within other LIME objects, e.g., AMBUSH is better suited as an ACTION, and some are used in combination, e.g., an OBSTACLE (*e.g.*, SINGLE CONCERTINA (2.X.3.1.11.7.1), can also have an obstacle effect associated with it (*e.g.*, FIX (2.X.3.1.7.2) or TURN (2.X.3.1.7.3). In 2525b these are separate symbols, but in LIME, the class representing Obstacles would include an *effect*, which could be as simple as an enumeration [FIX, BLOCK, DISRUPT, TURN, or NONE] or possibly another object with more semantics. The choice depends to a large extent on the implementation decisions made by the Commander's Associate and Blitzkrieg performers. **Requirement 8.1.1.**

Some example items from the scenarios with 2525b symbology that the Commander might designate as operational terrain include:

- Mid Intensity Conflict: LINKUP POINT (2.X.2.1.1.8.5), ENGAGEMENT AREA (2.X.2.1.3.3), and OBSERVATION POST OCCUPIED BY DISMOUNTED SCOUTS OR RECONNAISSANCE (2.X.2.4.1.2.2)
- Humanitarian Aid: LANDING ZONE (LZ) (2.X.2.1.3.7) and SEARCH AREA/RECONNAISSANCE AREA (2.X.2.1.3.9)
- Counterinsurgency: 2.X.2.6.2.5 NAMED AREA OF INTEREST (NAI), TRAFFIC CONTROL POST (TCP) (2.X.5.1.11), and DETAINEE COLLECTION POINT (2.X.5.1.5)

Looking at a specific example, the Engagement Area in Figure 4.3 shows several of operational terrain locations related to each other. The Circle around the letters "EA" is

[7]These numbers refer to the reference number within the 2525b document.

the area in which the Commander expects to engage the enemy, *i.e.*, the area where the direct fire and indirect fire weapons are planned to attack. The oval with the line in it is a company-sized battle position (2.X.2.4.3.1) and the parallel lines extending from the sides of the oval show the sectors of fire (fields of fire) in between which the company is expected to fire (SUPPORT BY FIRE POSITION (2.X.2.5.3.4)). The oval with the dashed line in it is an alternate company position that currently is not occupied but which can be occupied by one of the three companies shown here or by a reinforcing company.

The 2525b standard, however, is just a specification for drawing the symbol, it does not describe the semantics. In this seedling, we examined a sample of about 10% of the over 300 symbols to get a feel for the kinds of information that should be associated with operational terrain. The DG program will need to prioritize these to translate into LIME and will most likely need to extend them (*e.g.*, when the Commander creates a new symbol for a control measure). **Requirement 8.2.2.**

Based on this analysis, we reached the following conclusions regarding the minimum information that needs to be incorporated with OPERATIONAL-TERRAIN its subclasses **Requirement 8.1.2.**, **Requirement 8.1.9.**, **Requirement 8.2.6.**, **Requirement 8.2.7.**:

- Location: It is important to note that LIME, following the JC3IEDM, models domain and environment objects as things which *have* geometric objects as their location values, rather than as things which *are* locations. For example, an AIM-POINT is an object whose *location* is constrained to be a THREE-D-POINT, not something that *is* a THREE-D-POINT (see below for more on locations).
- Name or Identifier: A unique identifier (possibly scoped to within a COA or AO) for the OPERATIONAL-TERRAIN object.
- Affiliation: The affiliation of the OPERATIONAL-TERRAIN that indicates the side that currently controls it, if any. 2525b lists affiliation as an enumeration of [Pending, Unknown, Assumed Friend, Friend, Neutral, Suspect, Hostile, Joker, Faker, and None]. To accommodate the larger number of sides and shifting affiliations in operations such as Counterinsurgency, however, we recommend that this be populated in LIME with a object representing the specific side, e.g., FORCE-SIDE (see Section 4.2.1).
- Status: The status of the OPERATIONAL-TERRAIN. 2525b lists this as Anticipated/Planned or Present (see discussion in Section 4.2.3.
- TASKABLE-OBJECTS: Each of OPERATIONAL-TERRAIN types will have one or more TASKABLE-OBJECTS associated with it. The *meaning* of these objects, however, is highly dependent upon the role that the specific OPERATIONAL-TERRAIN is supposed to fulfill. For example, an AIM POINT (2.X.2.1.1.3.1) requires the TASKABLE-OBJECT(s) that is/are aiming at it, while an ENGAGEMENT-AREA (2.X.2.1.3.3) contains the TASKABLE-OBJECT(s) that will (or are) operating there.
- Regular Terrain: OPERATIONAL-TERRAIN is indicated by the Commander relative to a map containing regular terrain (see next section); sometimes this underlying regular terrain is important. For example, a BRIDGE OR GAP (2.X.3.2.2.2) is drawn crossing a piece of regular terrain, *e.g.*, a river, that is not specifically drawn by the commander but is no less important.

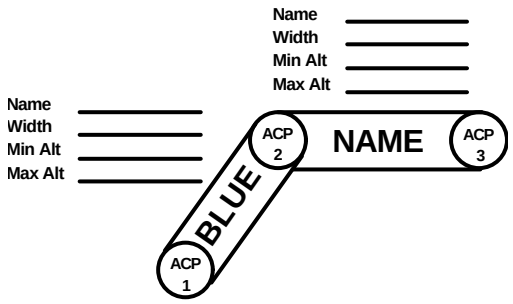

Figure 4.4: An example Minimum Risk Route from 2525b that reflect altitude and temporal information.

- Temporal Extent: All OPERATIONAL-TERRAIN objects have some temporal extent: they may persist indefinitely (insofar as DG is concerned anyway), persist for the duration of several COAs, just one COA, or even only for part of a COA. LIME does not address this within the OPERATIONAL-TERRAIN, but rather by the scope of the operation or COAs within which the OPERATIONAL-TERRAIN resides.
- Temporal Properties: In contrast to temporal extent above, some of the OPERATIONAL-TERRAIN types have temporal behavior that needs to be captured within the OPERATIONAL-TERRAIN. For example, MINIMUM-RISK-ROUTE (MRR) (2.X.2.2.2.2) is an air corridor that indicates a time frame for which it is active or open, as shown in Figure 4.4. This is a property of the MRR itself.

Regular Terrain

Regular Terrain is everything that is not in either of the other categories. On the Commander's map, regular terrain is present but it is usually not specifically referenced by the Commander. If there is something specific about it for the COA or operation, the Commander may indicate that with OPERATIONAL-TERRAIN, (e.g., control measures like NAMED AREA OF INTEREST (2.X.2.6.2.5)). For example, roads are Regular Terrain, and normally only the human map-reader associates the Key Terrain elements such as a RENDEZVOUS, drawn as a point symbol on a route, with a specific road. If components of DG must use this level of association to perform simulation or other reasoning, LIME would have to represent these links. We currently do not expect DG to need to register map features in this way. **Requirement 8.1.4.**

4.4.2 Locations

The topmost class in describing physical locations is LOCATION, which has a number of subclasses that allow us to capture the geographic location information about LIME entities. This whole section: **Requirement 8.1.1.**, **Requirement 8.1.4.**, **Requirement 8.1.9.**, **Requirement 8.2.1.**

The LIME location entities closely parallel the JC3IEDM LOCATION structure, so they

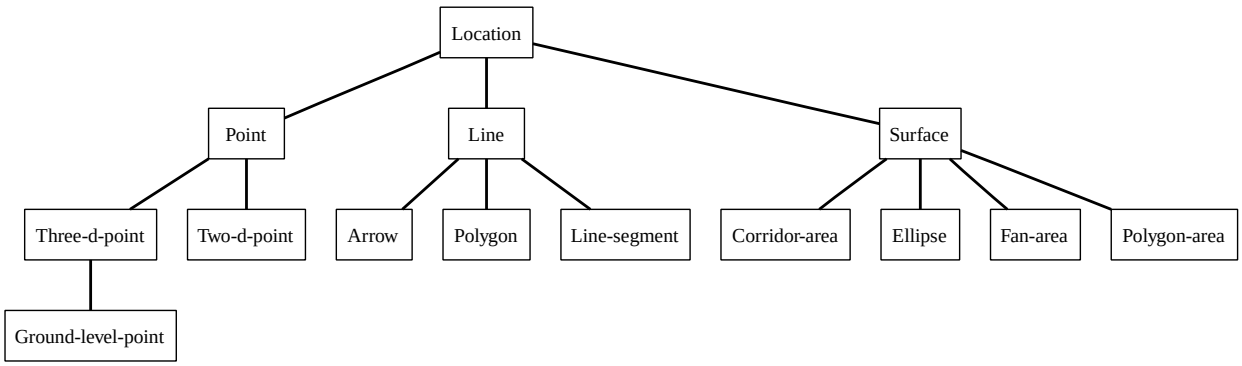

Figure 4.5: LOCATION subclasses.

will be readily translatable, forward and back. In LIME, the LOCATION will be an abstract concept — there may be objects with properties of type LOCATION, but all instances will be of concrete subclasses of LOCATION. Initially, LIME will offer the LOCATION subclasses shown in Figure 4.5. For the moment, we have omitted three-dimensional shapes, since we expect that initial DEEP GREEN efforts will primarily involve two-dimensional shapes "draped" on the surface of the Earth; later volumes will become more important and can be added. This and the following all apply to meeting: **Requirement 8.1.1.**, **Requirement 8.1.9.**, **Requirement 8.6.5.**, **Requirement 8.6.6.**

Point

As in the JC3IEDM, it's the POINT class that ties all of the locations to a place in space. The other LOCATION subclasses all reference component points in order to attach themselves to actual places. The POINT lines up with the JC3IEDM type GEOGRAPHIC-POINT: it names a point in two- or three-space, using latitude, longitude and (optionally) altitude. POINTs have the following attributes (others may be added later):

- *latitude*
- *longitude*
- *uid*

The subclasses of POINT include:

TWO-D-POINT: The TWO-D-POINT is the concrete class that corresponds to an unextended POINT. It has all and only the properties of its parents.

THREE-D-POINT: A POINT with an *altitude* property, as well. The *altitude* is defined to hold the altitude in meters MSL of the THREE-D-POINT. This can readily be translated into a JC3IEDM VERTICAL-DISTANCE object.

GROUND-LEVEL-POINT: The GROUND-LEVEL-POINT is a subclass of THREE-D-POINT that does not have an explicit *altitude* property, rather the value of this property is defined to be the ground altitude at the latitude and longitude of the point.

Line

LIME follows the JC3IEDM in defining as a LINE what in other contexts is often called a "polyline": a LINE is defined by its *line-points* property, an ordered set of points. The LINE object is defined as a linked set of line-segments between adjoining pairs of points in *line-points* (but see POLYGON, below). Line-segments on the globe are defined as being great-circle lines. The LINE class is compatible with another, abstract class, ARROW. If a LINE is also an ARROW, then the line is directional. There are two subclasses of LINE:

POLYGON: Represents a closed shape. We assume the existence of a line-segment between the final point in the *line-points* and the first.

LINE-SEGMENT: A simple case of LINE that has only two points.

These may all be readily translated to and from JC3IEDM LINEs, the only oddity being the need to "close the loop" when translating a POLYGON into a LINE.

Surface

SURFACE is an abstract class representing closed two-dimensional geometric shapes. At least initially, we will primarily be interested in two-dimensional shapes "draped" onto the globe. The SURFACE is a simplified subset of SURFACE in the JC3IEDM. Concrete subclasses of SURFACE are:

CORRIDOR-AREA: A CORRIDOR-AREA is defined by a *width* and a *line*, the former of which is a measurement in meters, and the latter of which is a LINE.

ELLIPSE: A ELLIPSE is defined by two perpendicular, intersecting LINE-SEGMENTS, *ellipse-first-conjugate* and *ellipse-second-conjugate*.

FAN-AREA: A FAN-AREA is defined by a *minimum-range*, *maximum-range* (in meters), an origin (a THREE-D-POINT), an *orientation-angle* (in degrees from true north) and a *sector-size-angle* (in degrees). See the JC3IEDM for more details.

POLYGON-AREA: A POLYGON-AREA is defined by an associated *defining-polygon*, which is a POLYGON. Currently, polygons are defined to be "draped" on the surface of the globe. We may need more varied polygons later.

4.5 State Objects

In this section, we discuss the representation of States and the Futures Graph.

4.5.1 States and Transitions

The Futures Graph consists of *states*, connected together in a directed acyclic graph (DAG) by *transitions*. In the AI "classical" planning model, states are only implicitly represented, are static, have an associated truth assignment over some set of propositions, and are separated by single actions. In LIME, states are first-class objects, have temporal extent, and have associated sets of *object properties* and *active processes*. In this model, states are temporally extended over intervals over which processes occur, with concomitant changes

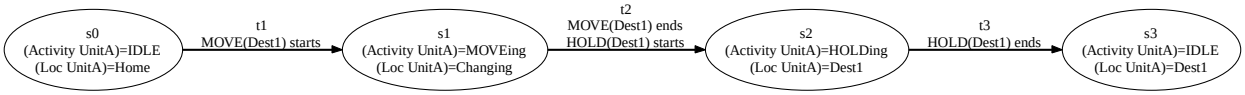

Figure 4.6: A very simple Futures Graph to illustrate state-encoded processes.

in the properties of objects involved in those processes. **Requirement 8.5.1.**, **Requirement 8.5.4.**

LIME states are connected by transitions, which have associated sets of instantaneous *events*. Events encode such things as the beginning or ending of *processes* (actions with temporal extent) which start or stop at that transition. In other words, the transition from one state to another happens when some process starts (it starts raining, or a new phase in a COA is initiated), or some process stops (the previous phase of the COA is completed), or both. While a transition may be associated with the beginning or ending of one or more processes, it is not the case that *all* processes will stop with the transition out of a state. Some processes may span several states. Transitions are labeled with numbers in the range $[0.0, 1.0]$, denoting the probability of that transition occurring. In the current DG vision, these probabilities will be generated as the result of simulation runs in Blitzkrieg. A state must have the property that the sum of the probabilities on all outgoing transitions sum to at most 1.0. If the sum is less than 1.0, that means there is a non-zero probability of no transition happening at all. **Requirement 8.5.9.**, **Requirement 8.3.5.**

In the simple example of Figure 4.6, a COA consisting of a MOVE followed by a HOLD might result in a Futures Graph in which there are four states s_0, s_1, s_2, and s_3, separated by transitions t_1, t_2, and t_3. In the initial state s_0, no actions are taking place, and the unit is at its starting location. In the transition t_1 leading to s_1, an activity corresponding to the MOVE action in the COA starts, and during s_1 that move is occurring. In transition t_2 leading from s_1 to s_2, the MOVE activity ends, and an activity corresponding to HOLD starts.[8] In s_2, the unit's location is the intended destination of the MOVE, and the unit is HOLDing. Finally, in the transition $t3$ leading to the final state s_3, the HOLD activity ends.[9] **Requirement 8.3.5.**, **Requirement 8.5.3.**, **Requirement 8.5.1.**

Suppose that under some circumstances the MOVE action can fail, leaving the unit somewhere in the movement corridor. In that case, the Futures Graph returned by Blitzkrieg will contain at least one additional transition t_{1a} leading from s_1 to a state s_{1b}, in which the MOVE activity has stopped, but the unit is in a location between the initial and the destination location. In that state the *hold* action cannot be executed because the unit is at the wrong location; Blitzkrieg is expected to return an indication that the planned COA could not be continued. The probabilities for t_1 and t_{1a} will be derived from statistics accumulated by Blitzkrieg over multiple simulations. **Requirement 8.6.3.**, **Requirement 8.6.4.**, **Requirement 8.3.9.**

[8] In some cases, it may make more sense to model this as two separate transitions with an intervening state s_{1a}, in which neither activity is occurring.

[9] This model of states is quite different from classical planning, and very similar those used in hybrid automata models such as Linear Hybrid Automata (LHAs) [2].

4.5.2 Attributes of State Objects

State objects will have at least the following attributes **Requirement 8.5.3.**, **Requirement 8.5.4. Requirement 8.3.6.**:

A Futures Graph– which the state is part of.
Incoming transitions – which may be a set.
Outgoing transitions – which may be a set.
Active processes – What action(s) are currently being executed by all sides? For states in the Futures Graphs returned by Blitzkrieg, each side may be executing one or more actions, all of which started no later than the beginning of the current state (i.e., on some transition into the current state) and end no earlier than the end of the current state (i.e., on some transition out of the current state).
Active COAs – What COAs are currently in process by all sides? Note that while a given side may not know what COA some other side is following, the simulator will have that information, and can (must) associate it with the state. In addition, where the various sides are in their COAs (a COA "program counter") must be maintained as well.

Finally, states will have a set of associated *metrics*. These metrics can be computed from the structure of the Futures Graph or features of the state itself, but may be cached for efficiency's sake. These metrics will at a minimum include the probability of reaching a given state s, given the current Futures Graph. **Requirement 8.5.6.**, **Requirement 8.5.9.**, **Requirement 8.3.9.**

Two other metrics that have been mentioned with respect to states and Futures Graphs in the DG program are *flexibility* and *quality*, but these have not yet been specified sufficiently to make a formal definition possible. Flexibility can be defined intuitively in terms of the *options* available to the commander at various points in the current COA. The question is, which points? Certainly not every state transition in the Futures Graph, since some of them will simply record steps in some phase of the COA, with no new information or enemy actions to respond to. **Requirement 8.5.6.**

Our current candidate for points at which to evaluate flexibility is *decision points*, which Army Publication FM 100-40 defines as follows:

> A decision point is a point in time or on the ground tied to an event where the commander must make a decision. His decision may be to do nothing. It is located to allow friendly units adequate reaction time after an enemy indicates that it is choosing a particular course of action. A decision point helps the commander decide where and when to adjust his maneuver to concentrate the effects of overwhelming combat power and defeat the enemy.

A more restrictive definition would be to evaluate flexibility only with respect to decision points associated with *critical events*, which are defined as events that directly influence mission accomplishment (FM 101-5, p. 5-18). In either case, flexibility may then be defined in any of several ways. One of the more promising is suggested in the DG BAA: impose

a lower bound on the number of alternative choices available to the Commander at any decision point. **Requirement 8.5.6.**, **Requirement 8.6.7.**

We can talk about the quality or utility of a state in similarly broad terms. There are several attributes associated with a state that might be of interest. One is some measure of *heuristic distance*: how far is the current state from a goal state, measured in terms of the number of transitions? Number of decision points? Number of critical events? How likely is execution from the current state to reach a goal state? **Requirement 8.5.6.**

Another quality measure associates a value with the state itself, by defining a value function that evaluates the state in terms of both its attributes and the object properties that hold in that state, mapping to a numeric value. Finally, we might be interested in what is commonly called *expected value*, which can be computed by evaluating the values of successor states to the current state, weighted by the probability of actually reaching those states. **Requirement 8.5.6.**

4.5.3 Attributes of Transition Objects

Transitions have attributes as well. These include **Requirement 8.5.9.**:

- A Futures Graph, which the transition is part of.
- A set of *events*, denoting the start or end of processes corresponding to actions taken, or to natural processes.
- An *originating* state, which is terminated when the transition occurs.
- A *resultant* state, which is initiated when the transition occurs.
- A *probability*, encoding the likelihood that this transition will occur, ending the state in which it originates.

4.5.4 Object Properties

Many of the Domain Objects in a scenario will include state-specific properties or property values. For example, the level of a fuel resource or the status of a Facility will be determined with respect to each Futures Graph state. Thus the access to a property value on a Domain Object must sometimes be with reference to some Futures Graph state. More formally, the state-varying properties of objects are a function of both the object instance and the state. We leave unspecified how the implementation should best support this functionality, as it will depend heavily on the selected infrastructure, however, it is clear that at some level of the API to the Futures Graph there must be a function from state $\times$ object $\times$ property-expression to a "propval." Further, since the "propval" is the value of the property over a *temporally-extended* state, it must be, in general, set-valued, not a singleton (although the set may be compactly represented by a simple label like "High"). **Requirement 8.5.2.**

4.5.5 Processes

States will include information about the set of processes being conducted by each active Taskable Object (*i.e.*, who is doing what, during that state). Recall that Taskable Objects include all of the military forces in the domain as well as non-military forces and inanimate actors (Nature). Any change in the properties of some domain object during the state should be the result of some process. **Requirement 8.3.1.**, **Requirement 8.4.6.**

4.5.6 States and Limited Information

A crucial property of battle planning and battle command is limited information: there are some things the Commander will not know (either because there is no way to observe them, they haven't been observed, or they have been observed/reported incorrectly).

Examples of limited information include:

- The affiliation of a particular force/unit.
- The size and properties (training, equipment, tactics) of a particular force/unit.
- What actions have been executed or are now being executed.

In a COA, the need to gather information about these kinds of properties or events is encapsulated in various kinds of information requirements: Commander's Critical Information Requirements (CCIR), Priority Intelligence Requirements (PIR), and the more generic Information Requirements (IR).

In the state model, each property and process shall carry with it annotations regarding which force Commanders are aware of the current state of that property or process. For example, the Commander issuing orders to a unit will know where the unit intends to go, but will not know exactly where they got to until the unit reports back. The Commander of an opposing force will not know the unit's location unless he takes specific actions to find it, or has other ways of tracking the unit's movement. The status of these annotations will be maintained by Blitzkrieg. Blitzkrieg contains the most detailed *observation model* available to the system, and is thus in the best position to track who knows what. **Requirement 8.1.7.**, **Requirement 8.2.5.**, **Requirement 8.2.6.**, **Requirement 8.1.4.**

Formally speaking, this kind of observation model is deliberately much weaker than even a limited modal formalism of knowledge or belief. Both the motivation and the justification for these simplifications is the fact that Blitzkrieg provides the dynamical model for the domain. The detailed state changes resulting from different action choices are determined by the simulation, not by the domain model. As discussed in Section 4.6, the domain model may contain a more abstract dynamics for use by the Option Generator and Detail Adding Planner, but those functions have not been defined in sufficient detail to determine detailed expressive requirements. **Requirement 8.3.9.**

A model in which the representation of "knowledge" is thus limited can easily be represented in a first-order model, without resorting to modal operators. As we can write (AMMUNITION-RESERVES COMPANY-A 50%), we can write (OBSERVED ENEMY-COMMANDER (AMMUNITION-RESERVES COMPANY-A 50%)). Expressing the fact that the enemy commander knows the ammunition reserves of company-A, without specifying what the level of

the reserves is, would force us to a second-order representation. This is undesirable. Fortunately, it is also unnecessary, since for any state in which we wish to examine the knowledge of some actor with regard to a property, the value of that property is specified as well. Note that this representational strategy can be implemented by associating object properties with annotations about which forces are aware of them. **Requirement 8.1.7.**

Since we can express what actors have observed about the current state in the same language as the object properties and active processes associated with that state, in the rest of this document we will not distinguish between statements about knowledge and statements about those properties and processes.

4.5.7 State abstraction and equivalence

We distinguish between *qualitative* states and *abstract* states. A *qualitative* state is one in which the properties of individual objects have values drawn from the power set of the domain of ground-level values for those property.[10] So, fuel levels may be represented by a range. Locations may be represented by a set of locations, or by a single location with larger extent. **Requirement 8.1.7.**

Qualitative states share with ground-level states the restriction that no processes begin or end within the state. A single qualitative state denotes a set of ground-level states, all with the same set of active processes and same COA "program counters" but with different values for the properties of some object(s). Blitzkrieg is described as a qualitative simulation, and may operate on qualitative states. Crystal Ball may further aggregate the states in the Futures Graph returned by Blitzkrieg as needed for computational efficiency, or for presentation to the Commander in Commander's Associate. **Requirement 8.1.7.**, **Requirement 8.6.7.**

An *abstract* state denotes a set of qualitative states, removing the restriction on processes and COAs. For example, suppose a Commander has been assigned to suppress some set of weapon emplacements (*e.g.*, artillery or surface-to-air missile sites). The order in which they are suppressed doesn't matter, but completion (or failure) of the task as a whole is important. In this case, Blitzkrieg could return a Futures Graph in which there is a single state before the task starts, then a great number of different states and paths through those states, depending on the order in which the sites are addressed, all ultimately leading to one of two states: a state for success (all emplacements suppressed) or a state for failure. What the Commander will want to know in this case, and certainly what *his* Commander will want to know, is probability of success or failure, and the resiliency of the best COA under various disruptions. **Requirement 8.5.8.**, **Requirement 8.6.7.**

Consequently, it makes sense for computational efficiency, and even more so for presentation to a human user, to remove the details of the order in which sites were attacked, unless that had some bearing on the outcome. So for all the paths leading to success, for example, we could have one initial state, one state that represents the process of suppressing all the sites in whatever order, and the final state in which success has been achieved. Being able to do this kind of aggregation will be crucial to DEEP GREEN's functioning.

[10] This is not to say that the *entire* power set will necessarily be representable in LIME.

Requirement 8.5.5.

The difference between qualitative and abstract states, then, is that abstract states may aggregate over states in which different processes are active (i.e., different actions are occurring), and more significantly, over the states reached through different sets or sequences of transitions. This may include states where one occurs after the other, in the sense that it follows the other along some path through the Futures Graph. So where in a Futures Graph consisting only of qualitative states there might be multiple transitions in a sequence, or even multiple paths through a larger graph, in the corresponding abstract state there may be just a single state, for which the action representation may include processes that begin and/or end "during" that state. **Requirement 8.1.7.**, **Requirement 8.5.8.**

4.6 Actions

In this section we discuss how actions are to be modeled in LIME. Note that most of the material in this section has to do with *meta*-modeling, with respect to much of LIME. That is, we will be specifying language constructs that describe *types* of actions, and relations between those action types. The actual actions and tasks that appear in most DG data structures will be *instances* of the action types described in this section. So, for example, the objects in this section will contain information about action parameters; those parameters will appear as properties of the action objects that appear in DG states. **Requirement 8.3.3.**

Action definitions will be split into two parts: action *declarations* and action *models*. An action declaration will specify the externally-visible attributes of an action, primarily its parameters and their types. This information will be necessary in order to pass actions to Blitzkrieg, or to digest action information in the Futures Graph. We also provide an account of *action models*. These action models are somewhat conjectural, and are primarily intended for the benefit of planning components of DG (e.g., the Detail-Adding Planner and the Option Generator), which are likely to use projection and to focus on the nominal outcome of actions. These are lower-fidelity action models than those needed by Blitzkrieg.[11] **Requirement 8.6.1.**, **Requirement 8.6.4.**, **Requirement 8.6.3.**

In this section, we use a lisp-like syntax for the action declarations and action models. This is helpful because we envisage these components being symbolically manipulated, which is very easily done with lisp-style s-expressions. XML would provide similar value for manipulation, but is less human-readable, so we have not used it here. **Requirement 8.3.1.**

4.6.1 Action declarations

An action declaration specifies the parameters of the action, its superclasses, and some miscellaneous added information. Information about the dynamics of an action, how it changes the state, the way it uses resources, and so forth, appears in the action model (see Section 4.6.2). **Requirement 8.3.4.**, **Requirement 8.3.1.**

Here is the form for declaring actions:

```
(action action-name
   [:parameters parameter-declaration-list]
```

[11] Blitzkrieg models are private to the simulator, and so out of scope of the LIME project.

```
[:inherits-from parent-class]
[:autl-index autl-index-string]
)
```

The effect of a declaration like this one should be to declare that there must exist an action subclass whose type name is *action-name*. If there is a parent-class specified with `:inherits-from`, then the *action-name* class should stand in an **is-a** relation to the class *parent-class*. The inherits-from declaration also influences the interpretation of the `:parameters` specification; see below. The optional `:autl-index` is a string that specifies the AUTL item number that corresponds to *action-name*. This is only used for documentation purposes at the moment, but could be useful for things like displays. **Requirement 8.3.2.**, **Requirement 8.3.4.**

Relation to AUTL The Army Universal Task List (AUTL) is the suggested starting point for defining the set of actions (*action-name* values). Although they are known to military planners, the attributes of the AUTL tasks are not written down. For example, it is not explicitly stated that a tactical reconnaissance task (ART 1.3.3) [12] requires a Unit to perform the recon; an area, route, or zone to recon in; or that the expected outcome is information or some other artifact. These parameters need to be documented in a way that the all DG components can utilize them. **Requirement 8.3.2.**

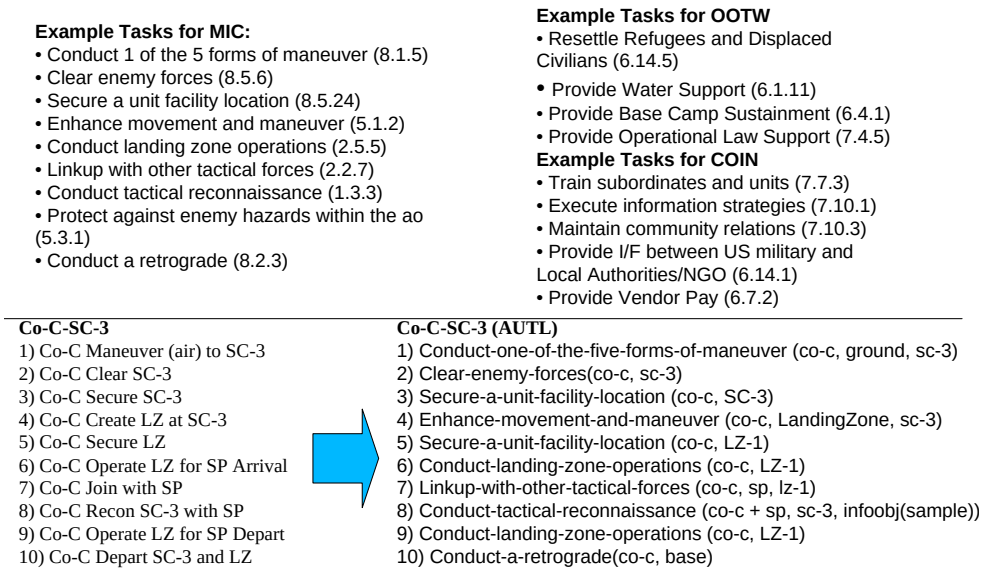

Figure 4.7: Mapping from the tasks required to execute a sample collection mission from the scenarios to the candidate AUTL action names.

We believe that the AUTL tasks at the ART x.y.z level (e.g., ART 1.3.3, "Perform Intelligence Preparation of the Battlefield") provide a set of action types sufficient to bootstrap the DG program. There are additional tasks, which provide specializations of the tasks at this level. For example, "Conduct Tactical Reconnaissance" has different subtypes for recon

[12] These are the references from the AUTL

of areas, zones, routes, and whether the recon is done by force or as a patrol. These more specialized actions can be added to the DG data model on an as-needed basis. **Requirement 8.3.4.**

There are also some cases where additional subtypes, over and above those provided by the AUTL, may be needed. For example, the AUTL provides a single task, "enhance mobility and maneuver" (ART 5.1.2) that covers a very wide range of possible activities (including, in our case, constructing a helicopter Landing Zone). In order for full shared understanding among the DG components, it may be necessary to introduce select subtypes below the leaf AUTL task types. For example, ART 5.1.2 covers road repair, road construction, and constructing a Landing Zone, activities that are radically different in their effects and durations. Using the action declaration facility, LIME can be extended to differentiate between these activities. **Requirement 8.2.2.**

LIME will need to support the definition of additional task subtypes for at least two more reasons **Requirement 8.1.4.**, **Requirement 8.4.1.**, **Requirement 8.1.3.**:

1. In out years of the DG program, we will be tackling COIN and OOTW. DG will need to be able to represent actions taken by non-military actors, and actors engaging in asymmetric warfare. Tasks like terrorist bombing attacks will need to be added. Since these are not tasks carried out by the US Armed Forces, they are not present in the AUTL or JUTL.

2. For orthogonality, we are representing environment processes as actions that can be performed by INANIMATE-ACTORs. For example, in the MIC scenario, we have a Fog process. Assigning these processes to INANIMATE-ACTOR(s)[13] allows us to have a place in the state/object model on which to position information about ongoing environment processes.

For Phase I, the x.y.z level of AUTL tasks provides a tractable number of tasks (about 170), so that the DG team should be able to add the parameter details needed. By prioritizing, this number can be reduced further (or at least spread out over the phases of the program). For example, the tasks defined in ART 6.0 need not be addressed until year 2 as these are more directly relevant to counterinsurgency scenarios. Figure 4.7 contains example AUTL tasks for each of the scenarios and also a mapping from a portion of the Humanitarian Aid sample collection mission to corresponding AUTL tasks. **Requirement 8.1.9.**, **Requirement 8.3.2.**

Parameters The `:parameters` give typed information about the slots of objects of the new type. The grammar of the *parameter-declaration-list* is as follows:

⟨*parameter-declaration-list*⟩ ::= '`(`' ⟨*parameter-declaration*⟩* '`)`'

⟨*parameter-declaration*⟩ ::= '`(`' ⟨*class-name*⟩ ⟨*param-name*⟩+ '`)`'

[13]For the initial phase of the program it may be sufficient to have a single INANIMATE-ACTOR, "Nature."

⟨*parameter-declaration*⟩ ::= '`(`' '`alias`' ⟨*old-param-name*⟩ ⟨*new-param-name*⟩ '`)`'

⟨*old-param-name*⟩ ::= ⟨*param-name*⟩

⟨*new-param-name*⟩ ::= ⟨*param-name*⟩

The parameters specification is influenced by the `:inherits-from` specification, if it is present. First, the ⟨*class-name*⟩ class will inherit all slots/parameters from its parent class. Second, when parsing the parameter declarations for ⟨*class-name*⟩, if the ⟨*param-name*⟩ already exists, then the corresponding ⟨*class-name*⟩ should specify a type that is compatible with the type previously declared for ⟨*param-name*⟩. That is, accumulated type declarations should only *narrow* the type of the slot. The '`alias`' declaration is provided so that programmers can specify class-specific aliases for a particular slot. For example, if an action-type were to inherit a very abstract `location` slot, it might wish to alias it so that it could also be referred to as the `target` (such aliasing is often done in the JC3IEDM). To do so, we would say:

```
(alias location target)
```

For this aliasing to be correct, the parameter ⟨*old-param-name*⟩ must have been defined in this action declaration form, or inherited from a parent action class. **Requirement 8.5.2.**, **Requirement 8.3.3.**

Example For example, in the Binni scenario, the COA uses the "Clear enemy forces" AUTL task (8.5.3). Here's how we might declare this type of action:

```
(action clear-enemy-forces
   :parameters ((alias location area-to-clear))
   :inherits-from conduct-tactical-mission-tasks
   :autl-index "8.5.3"
   )
```

Action declarations and action models For most programmers of the DG system, action declarations will be sufficient, and they may safely ignore the action models. Action models are intended for some more speculative components of the system, those that reason about the actions. For example, we expect that the action models will be of the greatest use for automatic planning systems that might be used in Option Generation (or possibly in the Detail-Adding Planner). **Requirement 8.3.1.**

4.6.2 Action models

As stated above, action *models* are intended to provide the information needed to plan projectively with the actions in the DG model. Note that LIME itself does not need action models at all, they are really only needed by planners (for DG these are the Commander and possibly portions of Commander's Associate, like Automated Option Generator), or by Blitzkrieg, and so not shared. Nonetheless, providing an action semantics for projec-

tive planning is very likely to be a technical requirement for the DEEP GREEN program.[14] **Requirement 8.3.5.**

Our initial design of these action models was heavily influenced by PDDL (the Planning Domain Description Language), and the extensions to that developed as LTML in the DARPA Integrated Learning program. LIME actions are all temporally-extended, or "durative" in the language of PDDL, so our actions parallel PDDL durative actions, rather than simple Strips or ADL operators. **Requirement 8.3.5.**

Our current focus is on planning only for nominal outcomes of actions. Actions can have off-nominal results, but these are represented in the state transitions, where Blitzkrieg has told us what happened. We provide some remarks about how to extend to conditional planning, below. The following apply to **Requirement 8.3.9.**, **Requirement 8.6.3.**, **Requirement 8.3.1.**

LIME diverges from PDDL in two important ways:

1. We assume an object-oriented state representation rather than a propositional state representation. Representing and reasoning about the properties of objects is cumbersome in PDDL, requiring explicit retrievals, deletions and assertions; PDDL provides no property update facilities.

2. We attempt to simplify domain modeling by building the concept of *resource* into the language. The intent is that the domain modeler be able to simply specify the resources used by an action, rather than have to encode each resource usage as a series of primitive database updates (which is effectively what PDDL effects are).

Here is a proposed action model skeleton:

```
(action-model action-name
   :parameters parameter-declaration-list
   :resources resource-list
   :duration duration-spec
   :condition condition-spec
   :complex boolean[15]
   :effects effects-specification
   :autl-index autl-index-string
   )
```

The LIME action models are similar to PDDL durative actions. We have extended that representation to provide more expressive power, so that it is less difficult to engineer these action models. The discussion here assumes that the reader is familiar with PDDL durative actions. A more complete discussion will be supplied later.

[14] As action models are a side issue for LIME itself, this section is less complete than other parts of this document. There are some unresolved issues in terms of an appropriate action semantics of which we are aware, others are likely to arise as the planners and Blitzkrieg are further designed.

[15] Optional: defaults to false.

Parameters The parameters of the action model should be congruent with the action declaration for the same action name. However, we permit multiple action models for the same named action. This permits different ways to perform the same action, depending on particular parameter values. For example, a maneuver action might be performed differently by a mechanized unit than by an irregular unit that must move at marching speed. **Requirement 8.3.9.**

Resource declarations The `:resources` specification of action models allows the modeler to specify a subset of parameters (or objects indirectly reachable from those parameters) that should be treated as resources. The syntax is as follows:

⟨*resource-list*⟩ ::= '`(`' (⟨*unary-spec*⟩|⟨*metric-spec*⟩)* '`)`'

⟨*unary-spec*⟩ ::= '`(`' ⟨*resource-spec*⟩ '`)`'

⟨*metric-spec*⟩ ::= '`(`' ⟨*resource-spec*⟩ ⟨*amount*⟩ '`)`'

⟨*resource-spec*⟩ ::= ⟨*name*⟩ | '`(`' ⟨*slot-name*⟩ ⟨*value*⟩ '`)`'

For example, the modeler may indicate that the agent of the action is to be treated as a unary resource, which will be seized at the start of the action, and not released until the end of the action, so that any other action that requires the same resource cannot occur concurrently. The ⟨*resource-spec*⟩ allows relative resource specifications, which are very common in this domain. For example, if a unit (referred to by the `$agent` parameter) is carrying out a maneuver, the relevant resource might be `(fuel $agent)`. We are currently working on an extension of this notation that would allow the modeler to specify resource usage as a function of the duration of the action, using a notation similar to that of PDDL+. At least initially, we expect that this would be compiled out into a durative action using a conservative over-approximation on resource use, rather modeling a continuous change in resource availability. **Requirement 8.3.7.**, **Requirement 8.3.5.**

Duration and Conditions The `:duration` and `:conditions` attributes of the action model are treated exactly as in PDDL with durative actions, with the exception that additional conditions and effects are implied by the `:resources` declaration. One difference in the conditions is that LIME conditions will typically refer to the values of object *properties*, rather than testing for membership (or absence) in a set of propositions. In particular, `equal` and `member` (for set-valued properties) will be important. Quantifiers, boolean operators, and temporal modalities will all be imported from PDDL. **Requirement 8.3.6.**

Effects specifications LIME uses a state representation that is consistent with conventional object-oriented or frame-based schemes: the overall system state is modeled in terms of the values of properties of a set of objects. In this way it is distinguished from the propositional representation of PDDL. Accordingly, in place of PDDL's database update operations for effects specifications, LIME uses the following state change specifications **Requirement 8.3.6.**:

set `(set` ***property-expression value***`)` Create a new assertion that *property-expression* has the value *value*. This does not erase any previous values for *property-expression*, so it can be used for multiple-valued properties.

unset `(unset` *property-expression value*`)` Undo the assertion that *property-expression* has the value *value*. This does not cause any reversion to default value, etc. Primarily intended for multiple-valued properties.

update `(update` *property-expression value*`)` Update the old value of *property-expression* to *value*. Should only be used for single-valued properties.

increase `(increase` *property-expression increment*`)` Update the value of *property-expression* by adding *increment* to it. Should only be used for single-valued properties.

decrease `(increase` *property-expression decrement*`)` Update the value of *property-expression* by subtracting *increment* from it. Should only be used for single-valued properties.

These expressions may be nested within `when`, `forall`, and `exists`, as in the ADL dialects of PDDL. *property-expression*s will be of the following form

⟨*property-expression*⟩ ::= '`(`' ⟨*property-name*⟩ ⟨*value*⟩ '`)`'

⟨*value*⟩ ::= ⟨*property-expression*⟩ | ⟨*name*⟩

Complex declarations We anticipate the need for hierarchical action specifications, and are currently working out the details for this part of the LIME action model. The `:complex` duration will signal that a particular action is hierarchical and must be executed indirectly by finding and instantiating a *method* for it, in HTN fashion. **Requirement 8.3.4.**

An Example Here is a simple example of the kind of action model we have in mind:

```
(action move
   :parameters (source - location
                destination - location)
   :resources ((agent) (route) ((fuel agent) (* duration (fuel-rate agent))))
   :duration (/ (distance source destination)
                (movement-speed agent))
   :condition (and (at start (equal (location agent) source))
                   (over all (not (exists (x - unit)
                                          (and (not (= x agent))
                                               (value (blocked-by route) x))))))

   ;; this effect specification gives a conservative
   ;; over-approximation of the unit's location
   :effect (and (at end (set (location agent) destination))
                (at start (unset (location agent) source))))
```

Remaining Issues for Action Semantics

The main issues left unaddressed in this discussion concern the degree to which the DG planning models need to represent the *purposes* (i.e., indirect effects) of actions, as well as

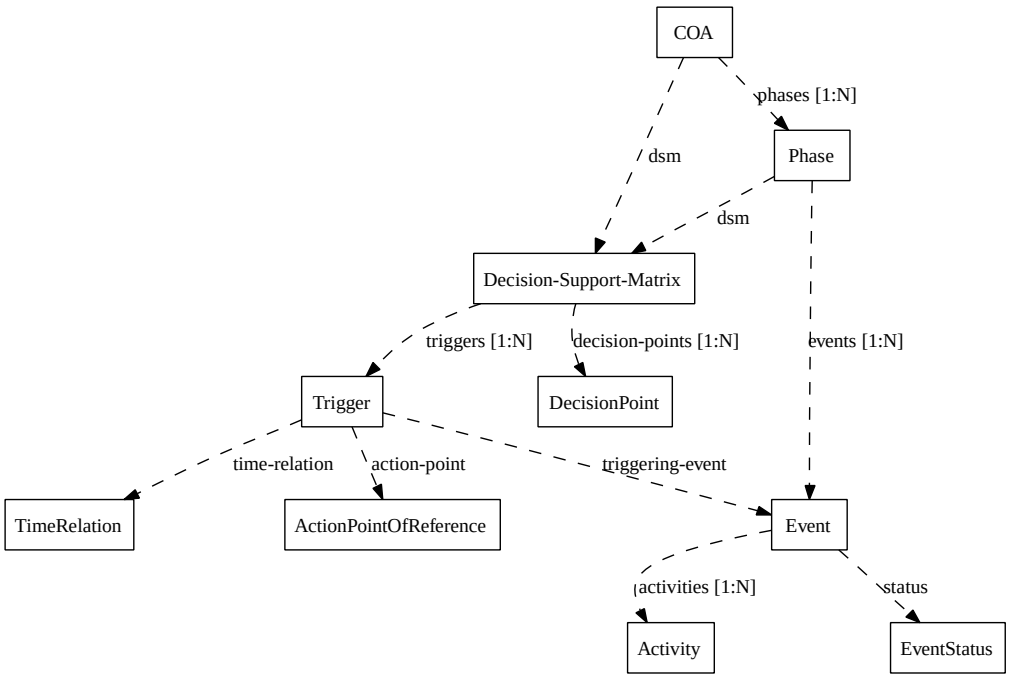

Figure 4.8: Course of Action elements.

the essentially adversarial nature of the domain. The complexity added by these properties is in fact one of the drivers of the current DG concept: use a simulator to extract projective information about plan execution, rather than trying to figure out how to model all of the various influences and effects.

4.7 Plans and COAs

A COA is the plan of action for each of the sides that the Commander produces through interactions with the Commander's Associate. In general, a COA will specify a set of desired actions (Section 4.6), along with the relationships between actions (*e.g.*, ordering) and events that the Commander desires. Portions of a COA may be conditional, meaning their activation is dependent on some aspect of the world state or the status of prior actions. **Requirement 8.4.1.**

Composing actions into COAs effectively requires a programming language that can express all the desired flow-of-control structures. Unlike many standard programming languages, and like procedural systems such as PRS [7], some of the COA control flow is based on event-driven and time-driven concepts, rather than linear and branching data-flow. Furthermore, because each side has multiple COAs and even alternative COAs defined for it, COAs can have extensive parallelism (concurrent or overlapping activities). **Requirement 8.4.2.**, **Requirement 8.6.8.**

The draft 2005 MSDL spec [13] provides a design for COAs that are divided into Phases.[16] Phases are in turn decomposed into Events (both controllable and uncontrollable tasks or

[16]The COA component was removed from the MSDL standard in order to keep within the standardization timeline; we anticipate that this component will be returned to the standard in a later version. [19]

occurrences).[17] Events wrap Activities (planned or possible actions/tasks/occurrences) for all of the forces and sources of change in a scenario, adding information about Status. The Phases of a COA are linked together into a Decision Support Matrix (DSM) showing how different Phases should relate to each other via Triggers. The Events within a Phase are also linked together into a DSM showing how different Events should relate to each other via Triggers. Triggers test when Events start or end (the ActionPointOfReference), indicating how new Phases or Events (*e.g.*, blue tasks, red tasks) should occur (or are expected to occur) at times relative to the Event changes. COAs and Phases provide scoping for the Trigger constructs. **Requirement 8.4.3.**, **Requirement 8.4.4.**

The following notional example shows the construction of a simple sequence where `activity-B` should start two hours after `activity-A` ends.

```
(let (($activity-A (move ....))
      ($activity-B (defend ....)))
 (trigger $activity-B :starts-after-end-of $activity-A :relative-time (hours 2)))
```

LIME may provide alternate ways to bind the identifiers such as `$activity-A` that are used as references (or "handles" in JC3IEDM terms).

MSDL provides a relatively rigid COA hierarchy with only two basic named levels of task hierarchy, Phases and Events/Activities. However, arbitrary levels of hierarchy are supported by allowing unit Activities to be entire subordinate COAs themselves. The intent is that at any echelon there are Phases and Events/Activities, and lower-echelon COAs may be involved in filling in the details of how an Activity is performed. **Requirement 8.4.2.**

While examples and detailed semantics are notably missing from the 2005 MSDL COA design, the general form meets our needs and the newer JC3IEDM includes a more complete set of action status values and temporal relations. LIME should be able to adopt and extend the MSDL COA constructs to handle additional needs, such as more explicit access to the LIME state representation and more forms of control flow. **Requirement 8.4.4.**

Features that will need to be added include **Requirement 8.4.2.** :

- The ability to refer to the outcome status of a triggering event; *e.g.*, do `task-B` if `task-A` succeeds and `task-C` if it fails. MSDL already has an Event Status slot and enumeration, but there are two remaining enhancements required. First, the Trigger must be extended to be optionally contingent on the Event Status, not just on its start/stop times, and second, the Event Status enumeration should be extended to capture the notion that a task can be concluded but unsuccessful.
- The ability to logically combine triggers to form composite triggers; *e.g.*, do `task-F` after both `task-A` *and* `task-B` succeed, where those tasks are potentially concurrent.
- The ability to tie triggers to the *end* of activities (in LIME terms: to the events ending those activities). In some cases, the Commander may need to specify triggers for the cessation of an activity. For example, "defend the perimeter of the LZ until all aircraft have taken off."

[17] The use of the term "Event" in MSDL is not consistent with the use of that term elsewhere in this document. Specifically, MSDL Events may have temporal extent, while LIME events are instantaneous.

- The ability to refer to the value of arbitrary state features in Triggers. For example, in the Binni scenario, the landing zone security task must persist indefinitely after the sample-collection team leaves, but only if the site is found to be contaminated. One way to capture this requirement would be to make the ending of the security task conditional on the perceived state of the world:

```
(let (($secure (secure $site....))
      ($collect-sample (collect-sample $site....)))
 (trigger $collect-sample :starts-after-start-of $secure)
 (trigger $secure :ends-after-end-of $collect-sample
     :if (not (contaminated $site)))
```

  Alternatively, the outcome status of the `collect-sample` task might be used to indicate whether the site was contaminated or not, and the Trigger could be conditional on that value. Note that the test of the `contaminated` slot of the `$site` is implicitly referring to the value in some state - in this case, the state when the trigger fires, at the end of `$collect-sample`.

4.7.1 Temporal Relations

Following the JC3IEDM [9] notion of ACTION-TEMPORAL-ASSOCIATION, LIME supports the expression of temporal relations among actions, events, and states in two general ways:

- Fixed (absolute) with respect to the standard calendar and wall-clock time (*e.g.*, 120700Z Sep07).
- Relative with respect to an origin time point or event (*e.g.*, 3 hours after mission start, or any time after the preceding action has ended).

The JC3IEDM includes a set of possible relations between intervals. The "closed interval" relationships listed in [9] Table 216 specify possible relations between both ends of two time intervals (*e.g.*, A starts and ends before B). The "open interval" relationships in Table 218 of the same document specify the possible relations of a single timepoint of each interval (*e.g.*, A starts before B (no statement about endings)). Collectively, these relations include the full set of relations in Allen's Interval Calculus [1] up to, but not including, disjunction (so, no non-overlap relation). **Requirement 8.3.5.**, **Requirement 8.5.7.**

4.7.2 Information Requirements

COAs have quite a lot of other information associated with them. Most of this information (the mission, risks, obstacles) we can represent as attributes that do not need to be parsed or interpreted, but simply carried along with the COA itself.[18] One possible exception is the associated information requirements, for example the CCIRs (Commander's Critical Information Requirements). In COA development and refinement, CCIRs arise from the identification of decision points and their triggers, and in turn may give rise to new tasks

[18]For a more complete description of the elements associated with a COA, see Chapter 5 of the Army Field Manual FM 101-5, http://www.dtic.mil/doctrine/jel/service_pubs/101_5.pdf.

in the COA at this echelon or a subordinate echelon, where those tasks are for the purpose of satisfying one or more information requirements. So, the IRs are reflected explicitly in the COA as tasks. As we have already discussed, the observation model by which those tasks lead to new knowledge is implicit in the simulation model internal to Blitzkrieg. Consequently, the information requirements, while they must be kept associated with the COA, impose no additional syntactic or semantic requirements on LIME. **Requirement 8.6.2.**

4.8 Goals and Commander's Intent

Commander's intent is key to assessing the value of different Futures Graph states and consequently the effectiveness of a particular COA. Current doctrine offers a number of different definitions for intent, all of which are more suited for human interpretation than for evaluation within a computer program. Four key elements of intent are:[19]

- The purpose of the mission
- The desired end state
- Key tasks
- Qualifiers

In order to evaluate Futures Graph states automatically, DG performers will need to provide some mapping from the elements above to formal definitions that can be evaluated against a given Futures Graph or a given state. Using the LIME language elements defined thus far, we can map these elements as follows:

• **Goal States** — Goal states can be used to express both purpose ("Prevent the enemy from gaining control of the maneuver corridor.") and desired end state ("Preserve at least 50% of current fuel stocks."). **Requirement 8.4.5.**, **Requirement 8.4.6.**,
• **Abstract Tasks** — Key tasks can be expressed as abstract tasks in LIME. For example, a task such as `defeat-in-detail` will be present in LIME as an abstract task. **Requirement 8.3.2.**
• **Trajectory Constraints** — Trajectory constraints can be used to represent qualifiers.[20] A simple example of a trajectory constraint is found in our Binni scenario, where the Commander requires that every sample-collection site is secured before the sample-collection team arrives. Constraints of this sort should be written as rules applicable to *any* instantiation of the sample-collection activity. One way to accomplish this is to modify the representation of the sample-collection activity to decompose into a sequence of subtasks, the first being to secure the area, the second to secure the sample. However, this approach does not really align with the notion of capturing intent as a constraint on the selection of action alternatives guided by their effects. Furthermore, the approach does not permit alternative sample-collection activities that are *not* subject to the constraint. **Requirement 8.4.5.**, **Requirement 8.4.2.**

[19] Thanks to BAE Systems for the specific wording of this list.
[20] *Measures of Effectiveness* are a closely-related concept, also encodable using trajectory constraints.

A cleaner alternative is to provide a way to express the constraint as a restriction on acceptable COAs. This requires the ability to refer to variables that bind to COA elements. For example, we can imagine:

```
(constraint
  :parameters ((COA $plan))
  :body (forall (sample-action $sample) in (actions $plan)
          (and (exists (secure-action $secure))
               (member $secure (actions $plan))
               (eventually (starts-before-and-ends-after $secure $sample)))))
```

Trajectory constraints may also refer to state features. The modal temporal operators from PDDL3.0 [5]

```
(at end <GD>) | (always <GD>) | (sometime <GD>) | (within <num> <GD>) |
(at-most-once <GD>) | (sometime-after <GD> <GD>) | (sometime-before <GD> <GD>) |
(always-within <num> <GD> <GD>) | (hold-during <num> <num> <GD> |
(hold-after <num> <GD>)
```

provide an extensive set of *state trajectory* constraints. These, combined with the temporal relations on activities specified in the JC3IEDM, provide the expressive power needed to specify trajectory constraints relating state properties to other state properties, or activities to other activities. **Requirement 8.3.8.**

4.9 Conclusions

In this document, we have provided a set of design guidelines for LIME. Providing a fully-specified design at this stage is neither feasible nor desirable, given the considerable uncertainties about the detailed technical approach that will be pursued on the DG program. Instead, we have provided a description of how to meet the LIME requirements. We have shown how existing data models and standards may be used or extended to satisfy these guidelines, and documented the extensions required for these base representations to be of use in the DG program. Some of the main points include:

- MSDL will probably be the base for the state descriptions. However, in its current form, it is not sufficient to support the "Order Generation" function from CA.
- JC3IEDM will help address the domain requirements, but will need to be extended or wrapped somehow to express LIME planning models, and to express COAs. Also, since JC3IEDM has named types like "action" and slots like "subaction" a specified semantics for those relationships, an agreement on semantics will be needed between DG and the systems it is integrated with.
- The AUTL is a good source for the action types required for DG. Currently it appears that the AUTL tasks at the third level (x.y.z) are well-suited to DG's scope. However, AUTL tasks are essentially just symbols, without either an execution semantics, or a set of parameters specifying well-formed tasks. Both are needed.
- PDDL is a plausible candidate as a basis for representing change between states in LIME. However, AI planning languages including PDDL tend to represent states as sets

of propositions, which has the advantage of being domain-independent and maximally expressive, but can actually obscure important aspects of models, particularly such structured relationships as spatial reasoning, temporal reasoning, and object modeling (part/whole relations, e.g.). As discussed in Section 4.6, while we can adopt much of PDDL's structure, the representation of state must be extended.

At this point, the requirements imposed on LIME by the domain (and the design decisions dependent on those requirements) are well-understood. The implications of the detailed technical approach taken within each of the DG components (Commander's Associate, Crystal Ball, and Blitzkrieg) and of the overall system architecture are less well-understood, but some conclusions can be drawn. We have tried to be explicit about the limits of that understanding, where we have made assumptions, and where we have punted. Consequently the generation of a detailed language definition for LIME is a task left to the early days of the DG program.

Chapter 5

LIME Seedling Evaluations and Conclusions

5.1 Evaluation

In addition to review of early drafts of these chapters by a few people with a close knowledge of DG, a fairly complete draft of each of the main sections of this report was sent to seventeen peer reviewers, a mix of military and intelligent systems experts, to conduct a preliminary review of this document. They were requested to provide the following feedback:

1. Scenarios and use cases: Suggestions about adding further use cases, or refining our existing use cases, to better cover the intended capabilities of Deep Green.

2. Requirements: Assistance in identifying redundant, missing, or underspecified requirements for LIME, as well as requirements needing further explanation or motivation.

3. LIME design document: Comments on refining and extending the design recommendations and the accompanying rationale.

Despite a relatively short turnaround time prior to this report needing to be finalized, feedback was received from nine of the evaluators.

- Mr. Jeffrey Abbott, Acusoft
- Mr. Pete Corpac, STA Associates
- Dr. Ken Forbus, Northwestern University
- Mr. Eric Jones, BAE
- Dr. Dana Nau, University of Maryland
- Dr. Mark Peot, Teledyne
- Dr. Robert Schrag, Global Infotek
- Prof. Austin Tate, AIAI, University of Edinburgh

- Dr. Ken Whitebread, Lockheed Martin ATL

Their thoughtful and constructive feedback was very useful and has been folded into the preceding documents where possible. Remaining items (e.g., suggestions on where to begin implementation) have been retained in the LIME documentation repository. In the event that evaluator comments arrive after submittal of this document, they will be stored in the LIME documentation repository, and also forwarded to DARPA so they are not lost when the DEEP GREEN program starts.

5.2 Conclusions

This report provides recommendations for developing a domain modeling and inter-module communication language for use during the DEEP GREEN program. Driven by the program description from [20] and relevant scenarios, use cases were developed to drive the LIME requirements that led to the design recommendations. As stated earlier, our intent was not to prescribe a specific implementation approach, but rather to describe the aspects necessary to support DG and to minimize the difficulties of integrating the DG components and integrating DG with existing and planned external simulation and C3I systems. This work is intended to form a starting point for the DEEP GREEN program performers and will evolve and adapt as the DG development begins in earnest.

5.3 References

[1] J. Allen, "Towards a General Theory of Action and Time," *Artificial Intelligence*, vol. 23, pp. 123–154, 1984.

[2] R. Alur, C. Courcoubetis, T. Henzinger, and P.-H. Ho, "Hybrid Automata: an Algorithmic Approach to the Specification and Verification of Hybrid Systems," in *Hybrid Systems*, R. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors, Lecture Notes in Computer Science 736, pp. 209–229, Springer-Verlag, 1993.

[3] *Counterinsurgency: Army FM 3-24 and Marine MCWP 3-33.5*, Headquarters of the Department of the Army and HQ Marine Corps Combat Development Command, Department of the Navy, December 2006.

[4] Free dictionary definition of "First class object.". Available online as `http://encyclopedia.thefreedictionary.com/First-class+(object)`.

[5] A. Gerevini and D. Long, "Preferences and Soft Constraints in PDDL3," in *Proc. ICAPS workshop on Planning with Preferences and Soft Constraints*, A. Gerevini and D. Long, editors, pp. 46–53, 2006. Available online as `http://www.plg.inf.uc3m.es/icaps06/preprints/i06-ws1-allpapers.pdf`.

[6] R. Holliway. *CountryWatch - Interesting Facts of the World*, 2007. Available online as `http://www.countrywatch.com/facts/facts_default.aspx?type=text\&topic=SEPKK`.

[7] F. F. Ingrand, M. P. Georgeff, and A. S. Rao, "An Architecture for Real-Time Reasoning and System Control," *IEEE Expert*, pp. 34–44, December 1992. Available online as `http://www.laas.fr/~felix/publis/publis.html`.

[8] Jane's Information Group. *Jane's World Insurgency And Terrorism Gerakan Aceh Merdeka (GAM)*, 2006. Available online as `http://www.caul.edu.au/datasets/jwit2006sample.doc`.

[9] *THE JOINT C3 INFORMATION EXCHANGE DATA MODEL*, Multilateral Interoperability Programme, February 2007.

[10] D. Killcullen, "Twenty-Eight Articles: Fundamentals of Company-Level Counterinsurgency," *Military Review*, May–June 2006. Available online as `http://usacac.army.mil/CAC/milreview/English/MayJun06/indexmayjun06.asp`.

[11] C. C. Krulak, "The Strategic Corporal: Leadership in the Three Block War," *Marines Magazine*, Marines Magazine, January 1999.

[12] G. Luck, "Insights on Joint Operations: The Art and Science," Technical report, U.S. Joint Forces Command, September 2006.

[13] *Specifications for Military Scenario Definition Language (MSDL)*, Simulation Interoperability Standards Organization, Military Scenario Definition Language Study Group, 2005, Unpublished initial draft.

[14] *Specifications for Military Scenario Definition Language (MSDL)*, Simulation Interoperability Standards Organization, Military Scenario Definition Language Study Group, July 2007, Unpublished initial draft.

[15] G. Packer, "Letter from Iraq: The Lesson of Tal Afar," *The New Yorker*, The New Yorker, April 2006. Available online as `http://www.newyorker.com/archive/2006/04/10/060410fa_fact2?printable=true`.

[16] R. A. Rathmell, "A Coalition Force Scenario 'Binni - Gateway to the Golden Bowl of Africa'," in *Proceedings of the International Workshop on Knowledge-Based Planning for Coalition Forces*, pp. 115–125, Edinburgh, Scotland, May 1999.

[17] C. Robinson, "In the Spotlight: Indonesia's Free Aceh Movement," Technical report, Center for Defense Information, May 2002. Available online as `http://www.cdi.org/terrorism/aceh-pr.cfm`.

[18] J. F. Schmitt, *Mastering Tactics: A Tactical Decision Games Workbook*, Marine Corps Association, Quantico, VA, 1994.

[19] J. Surdu, 2007, personal communication.

[20] J. Surdu. *DARPA BAA 07-56 Deep Green*, 2007.