13th ICCRTS: C2 for Complex Endeavors

"USING A COMMAND AND CONTROL LANGUAGE TO SIMULATE OPERATIONS IN A MULTI-AGENT ENVIRONMENT"

(Suggested Topics)

Topic 3: Modeling and Simulation

Topic 2: Networks and Networking

Topic 9: Collaborative Technologies for Network-Centric Operations

Erik Borgers
Mink Spaans
Jeroen Voogd

TNO Defence, Security and Safety
PO Box 96864
2509JG The Hague
The Netherlands
0031-070-374-0154
Erik.Borgers@tno.nl

Dr. Michael R. Hieb

Center of Excellence for C4I
George Mason University
4400 University Drive
Fairfax, VA 22030
USA
001-703-993-3990
mhieb@gmu.edu

Remco Bonse
Agent Technology,
Intelligent Systems group
Dept. of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands
remco.bonse@cs.uu.nl
STUDENT

Using a Command and Control Language to Simulate Operations in a Multi-Agent Environment

Abstract

One of the most effective tools in planning complex operations between different organizations is a simulation of what is to be done. Given a common intent, a simulation provides a basis for understanding the different elements of an operation, and thus enables flexibility as a plan is developed and implemented. The state of the art in simulation is with multi-agent environments. Our work is in developing the abilities of agents so that they reason and act correctly in the simulation. We describe an agent engine called 2APL and its communication protocols.

One of the most critical problems in simulating military operations is communicating the intent of what is to be achieved to an agent. This intent can be transmitted effectively between humans, but is problematic when working with agent implementations, due to the large amount of interpretation a human performs. We use a language called the Command and Control Lexical Grammar (C2LG), derived from a body of work called Battle Management Language for its precision and C2 semantics. In this paper, we present our experience in using the C2LG and assess the language for use with simulation agents for developing more effective simulations for Complex Endeavors.

Keywords: Agile Command & Control, Decision Support, Battle Management Language, Command and Control Lexical Grammar, Intelligent Agents

1. Introduction

As discussed in several publications [Alberts and Hayes, 2003; Alberts, 2007; Grisogono, 2006; Grisogono and Armenis, 2007] agility is an important requirement for C2 in Complex Endeavors. Based on the results of research in the area of C2, Complexity Theory and Complex Adaptive Systems (CAS), several groups and organizations are actively dealing with this subject.

Agility is the ability to maintain effectiveness in a dynamic, complex and rapidly changing environment. Agility requires new capabilities in C2 processes and systems. It requires insight into the network of causes and influences, which are formed by both known actions as well as external causes and influences on both internal and external elements. C2 systems need to be more flexible in how they can deal with, and support, complex and evolving situations. A broader range of decisions need to be supported, requiring more collaboration and analysis of how various organizations can work together.

A promising approach to obtain these capabilities is through the use of simulation. However, simulations capable of flexibly modeling a wider range of operations, both military and civilian, still need to be developed.

The next chapter discusses the role that simulation can play to support agility, the requirements this puts on the simulation and the proposed simulation architecture. Chapter 2 describes the required elements for simulation in support of agile C2. Chapter 3 describes a Proof of Concept study, and finally Chapter 4 presents lessons learned and a way forward.

1.1 Simulation in support of agile C2

Simulations support operations in many different areas. Typically, these are categorized as 1) Training for Operations; 2) Decision Aids for developing Plans; and 3) Mission Rehearsal. A typical use for simulations today is to program various force structures and scenarios to evaluate several alternative plans. In the future, we will see a much greater use of simulations to better understand more complex situations.

Simulation models can also help decision makers in understanding cause and influence networks. This functionality can be obtained by statistical analysis of the results of many simulations each with random variations based on the current situation in the theatre of operation and the plan (such as a possible Course of Action (CoA)) to be evaluated.

However, the current set of simulations is much too brittle and inflexible to accommodate where C2 is evolving to. Thus we need to set forth a revised set of requirements to develop suitable simulations for the future.

1.1.1 General Requirements for C2 Simulation based support

We believe that in order to create a workable offset of simulation support tools for C2, these specific technological hurdles must be addressed:

- The time for initialization of a simulation from C2 information (information from the C2 Support Systems (C2SS) on which a plan is based) must be short compared to the total available time for planning.

- Misinterpretation of a COA must be minimal (preferably, there should be just one possible interpretation for the simulation to run), so that extra adjustment during execution is avoided.

- The simulation should execute with good enough fidelity to avoid having to do lower level planning for simulated troops. Again this would take too many operators and too much time.

- Running the simulation should require minimal use of human operators. The commanders' staff should not need to include a large group of simulation operators.

- The operator intelligence should be “pluggable”: it should be possible to exchange actors, either software-based or human-based, to play the role of operator.

- Simulations should be able to run much faster than real-time, or at least within the available time for plan adoption.

These requirements help us "bridge the gap" between Command & Control and Simulations. As of 2007, this gap still exists and the result is that simulators are still controlled by human operators, primarily run in real time, and use input that is not well correlated semantically or syntactically to C2 information. As a result, simulators are still mainly used in the areas of off-line and non-critical analyses and training, where availability of operators and time is less of a problem. However, we believe that the work in this paper describes an innovative combination of new and existing technologies that can help to change this and "bring M&S to the battlefield".

1.1.2 Specific Requirements for Agility

A number of simulation specific requirements should be met for a simulation to be of valid use for support of agility in C2. These requirements include:

- In order to support agility, plans cannot be based on 'standard solutions' with many simplifying assumptions (symmetrical enemy, uninhabited areas etc), as is common in classic conflicts.

- External influences, such as events in the theatre of operation need to be taken into account.

- The simulation should provide insight into causes and influences caused by known actions and other factors both internally and externally.

- The simulation should allow various players, both military and non-military, to each have their own doctrine, behaviours and organization.

1.2 Analysis of the Requirements Identified

The communication between actors can be considered as one of the most crucial influences on behavior. In particular, the communication of intent, the interpretation of this intent and decisions to act on this interpretation (or understanding) will ultimately determine the synchronization of efforts and the success of particular tasks. This means that simulations must also be able to represent and use intention. Scripted behavior is too rigid to be used effectively in an agile Command & Control environment.

For years, TNO (Applied Scientific Research Institute, Netherlands) has been active in building simulators that support the training of command and control staff in the planning processes. The automated transfer of data between C2 and Simulators has been an important research goal (see for example [Borgers et al, 2007]). The training simulators in most cases simulate the lower level units that interact with, and perform activities, in a simulated environment. As such, commanders build insight of the consequences of their planned actions. TNO, and others, are in the process of building and adapting simulators with new architectures to provide commanders decision support during operational activities.

Currently there are a number of standards built to allow simulations to communicate physical effects in the simulated environment. One such standard is the High Level

Architecture – HLA (IEEE, 2000). HLA allows communication of messages, but does not explicitly support expressing agent intent nor agent specific architectures. In practice, this has resulted in many different implementations of command agents that do not interoperate nor understand the intent of other implementations.

While interfaces could be built to translate from one format or protocol to another, in practice this is not feasible, as it results in an increasing number of point-to-point interfaces. Thus there is a need for a standard to communicate intent to allow the simulation of a complex planning process.

Our experience is that current simulators only are satisfactory for some of the proposed requirements. The contribution of this paper is to describe an agent architecture that has the following benefits:

- Reducing the use of human operators and running simulations more quickly facilitates the use of these simulators during the operational planning of an employed force. This allows better insight into the consequences of actions and, especially, the ability to adapt plans accordingly.

- A pluggable agent architecture better supports the planning process of different combinations of commanders. This allows more flexibility.

2 Proposed solution

In order to build better simulations, we need better, more intelligent agents and better communication between these agents. This necessitates both new technologies to represent the behavior of these agents, and a communications protocol that, among other things, can be used to express intent that is able to be interpreted in a standard way.

2.1 Proposed architecture

The architecture proposed in this paper has a number of important characteristics that enable the simulation to represent and use command intent.

An important characteristic is the Agent-based approach that was taken. The agents communicate with the human operators, the users of C2 Support Systems (C2SS) and other agents in the simulation, using one and the same language, granting them the same set of actions a human operator has. A specific protocol for C2 called Battle Management Language (BML) and a set of engineered behaviors are used to achieve this. Another important aspect is that the Agents are not embedded in the C2 or in the Simulator. Agents are separate pieces of software that can be added to the infrastructure when needed, they act as operators but can also be replaced by operators when the situation changes. For example, a passive irregular enemy commander could be replaced by an aggressive one. Vice versa the same agents could be used to test the decisions made in any other simulator that is able to interpret BML. This makes the architecture potentially very flexible and also means that producers of these types of Command Agents do not need to take into account the specific C2 nor simulation system that will be used.

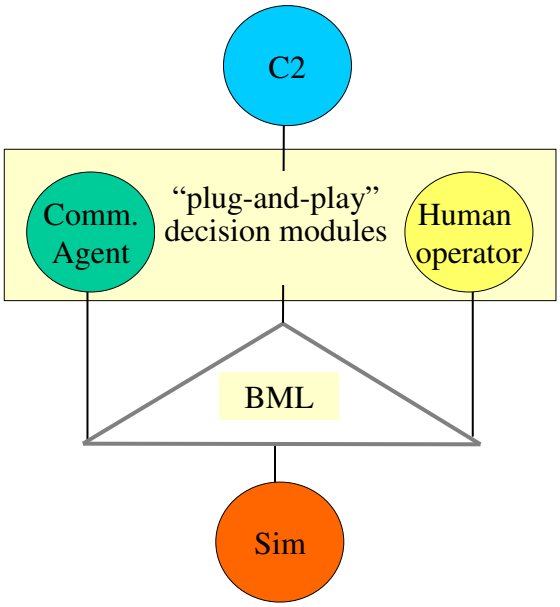

Figure 1. Sketch of the proposed architecture

The agents within the architecture described in this paper need to be able to interpret the BML messages from the C2 or Simulation System and be able to decide what to do. They also must be able to pass intent and other information to other actors, either operators, C2SS or other agents. As long as every agent, operator and C2SS sends and receives BML messages, a seamless connection between them is possible. As a matter of fact we have an architecture that enables an interchange between computer simulated agents and C2SS operators as needed. For instance, one can envision a battalion staff consisting of real humans with their C2SS where lower commanders and actors like Non-Governmental Organizations (NGOs) are simulated by agents.

2.2 Proposed communication protocol: BML

A fundamental requirement for support of C2 by simulators is the possibility to exchange data/information on both the physical and the semantic level. For the physical level, techniques are readily available to exchange data. We propose to use the new emerging BML standard for exchanging information at the semantic level.

The broad definition of a BML has been defined [Carey et al, 2001] as:

The unambiguous language used to command and control forces and equipment conducting military operations and to provide for situational awareness and a shared, common operational picture.

In other words, BML is a standardized language for military communication – which means it can be used to formulate orders, requests and reports.

BML as described precisely in the Command and Control Lexical Grammar (C2LG) [Schade & Hieb, 2006a], is designed as a formal language, i.e. the set of all expressions that can be generated by a formal grammar. This section will describe a brief overview of the C2LG. A formal grammar generally consists of a lexicon which provides the words of a language and a set of production rules indicating how the elements (words, phrases) of the language may be combined to construct longer expressions (sentences) using these words. The formal structure and unambiguous usage of the language elements is important, as, among other things, it increases clarity of communication between multinational forces and makes possible automatic processing of the expressions produced by the language.

The attributes and values used in the JC3IEDM comprise BML's lexicon. JC3IEDM is the data model developed as NATO standard by the Multilateral Interoperability Programme [MIP, 2007]. The data model defines terms for all the elements that may need communication during military operations, both conflict and non-conflict (humanitarian) operations. These terms range from tasks for orders or requests, descriptions of units, personnel, weather, time, location, etc. Using JC3IEDM terms as lexical items in BML facilitates the mapping of information expressed in BML into the data model. In addition, the JC3IEDM provides a sort of "dictionary" of the terms. This decreases misunderstandings about their meaning and thus eliminates ambiguity.

In this paper, we describe a formalism for BML, the Command and Control Lexical Grammar, that can be used to structure C2 information in a formal fashion. This formalism covers a grammar for tasking (called the C2LG) [Schade & Hieb, 2006a; Schade & Hieb, 2006b] and reporting [Schade & Hieb, 2007] following linguistic principles. The set of the expressions that can be generated by applying the lexicon and the rules of these grammars therefore build a Language for tasking and reporting. These grammars also provide the basis for representing more complex concepts used for Command Intent as shown in [Schade & Hieb, 2007].

2.2.1 Grammar Rules for Orders (and Requests)

The format of orders is defined by the NATO Standard STANAG 2014 "Format for Orders and Designation of Timings, Locations and Boundaries." An Operational Order is divided into five Sections: 1) Situation, 2) Mission, 3) Execution, 4) Administration and Logistics, 5) Command and Signal, and the respective annexes. Section 3 is used to "summarize the overall course of action," "assign specific tasks to each element of the task organization," and "give details of coordination." The tasking grammar [Schade & Hieb, 2006b] scope covers Section 3, "Execution", that consists of the Command Intent, the assignment of single specific tasks to specific units as well as the giving of details of coordination. Therefore, the basic rule of the tasking grammar is:

(1) $S \rightarrow CI\ \ OB^*\ C\_Sp^*\ C\_T^*$

This rule means that "execution" consists of the command intent (indicated by CI), basic order expressions to assign tasks to units (OB), spatial coordination (C_Sp), and temporal coordination (C_T). The asterisk indicates that arbitrarily many of the respective expressions can be concatenated together.

According to linguistic principles, basic order expressions are composed of a verb and its frame. The verb denotes a task. For the tasking grammar, tasking verbs are taken from JC3IEDM's table "action-task-activity-code" [MIP, 2007]. Thus, the rules to expand OB have the general form as given in (2a). (2b) and (2c) give examples for the tasks "advance" and "defend", respectively.

(2a) OB → Verb Tasker Taskee (Affected|Action) Where
Start-When (End-When) Why Label (Mod)*

(2b) OB → advance Tasker Taskee Route-Where
Start-When (End-When) Why Label (Mod)*

(2c) OB → defend Tasker Taskee Affected At-Where
Start-When (End-When) Why Label (Mod)*

Tasker is the name of the one who gives the order. Taskee is the name of the unit that is ordered to execute the task. Start-When and End-When are temporal phrases expressing when the execution of the task has to start and when it has to be finished. End-When is optional as indicated by the parentheses. Tasker, Taskee, Start-When, and End-When appear in each basic order rule.

Affected in (2a) has to be a term in the expression if someone, e.g., the enemy, will be directly affected by the task. Whether Affected is part of a rule depends on the tasking verb. For example, it is there in the case of ***attack*** or ***defend*** because the executing unit is tasked to attack the enemy or to defend against the enemy. It is not there in the case of ***advance***. The tasking verbs come with frames that express which kind of constituents are required, e.g., Affected. Action is similar to Affected. It only appears if the task affects an action, as a task of type ***assist*** does – the unit is tasked to assist the execution of another task by another unit. In addition, the type of the Where is also determined by the task. It is currently either an At-Where or a Route-Where. An At-Where denotes a location, and a Route-Where a path to a location. A Route-Where can be expanded to more complex concatenations of constituents as in "**from** LocationA **to** LocationD **via** LocationB **and** LocationC."

A basic rule ends with Why, Label and the optional Mod. Why represents a reason why the task specified by the rule is ordered – the mission's purpose. Label is a unique identifier for its task. By this identifier the task can referred to in other expressions, especially in temporal coordinations. The optional Mod (for modifier) is a wild-card that represents additional information that can be used to describe a particular task, e.g., formation – to specify a particular formation for an advance – or manner – to express for example whether the task in question has to be completed as fast as possible or more slowly, without taking any risks. Modifiers are particularly important for decision support.

2.2.2 Grammar Rules for Reports

With respect to reports, the C2LG represents reporting expressions (RB) as:

(5) S → RB*

The general form of a basic reporting expression depends whether the report is about military operations (task report), events (event report) or status (status report). The respective rule forms are given in (6a) to (6c).

(6a) RB → Task-Report Verb Executer (Affected|Action) Where When (Why) Certainty Label (Mod)*

(6b) RB → Event-Report EVerb (Affected|Action) Where When Certainty Label (Mod)*

(6c) RB → Status-Report Hostility Regarding (Identification Status-Value) Where When Certainty Label (Mod)*

Rule forms (6a) and (6b) are quite similar to the rule form for basic order expression as given in (2a). The differences are as follows: Neither (6a) nor (6b) has a Tasker. For (6a) this is because the reporter may not know the unit that has ordered the task he is reporting on, as with an action performed by the enemy. For (6b) this is because events like earthquakes happen and it does not make sense to say they are "commanded" by an organization. Rule form (6a) has a generalized Taskee named Executer in order to allow constituents like "four hostile snipers". (6b) has no Executer; it uses verbs that denote events.

Rule form (6c) uses Regarding instead of a verb to determine what kind of status is reported. Status reports can be given about the operational status of a unit (by using the key word status-general in Regarding), about the status of a unit's personal (status-person) and about the status of a unit's material (status-material). In addition, the position of a unit can be reported (position). All report forms include Certainty to specify the certainty of the report (see [Schade & Hieb, 2007] for more details).

2.3 Agent technology

Agent technology [Wooldridge, 2005] is an approach for meeting the requirements specified in chapter 1.1.1 and 1.1.2. An important part of these requirements is a solid communication protocol between agents (and between agents and humans) and a good representation of the behavior of agents. In the next paragraphs we will briefly discuss both topics.

2.3.1 Agent technology: Communication

Communication standards for agents are often based on a concept from language philosophy named Speech Act Theory (SAT) [Searle, 1969, 1979]. Within this philosophy *performatives* play an important role: they describe how the receiver of the

message should interpret the content. They have the option to accept, refuse, reply-to etc. They do not just have to blindly follow orders, but can negotiate. Compared to the situation of agents having to accept every message and try to interpret its content this gives a rich environment to communicate orders, reports and intent between actors. It also gives direction to what an agent should do with the content of a message (i.e. refuse an order because the sender is not a commanding superior).

The definitions of performatives have been captured in a standard by the Foundation of Intelligent Physical Agents (FIPA) [FIPA, 2002] and communication protocols are available as such in open standards. Performatives in FIPA ACL are defined in formal semantics inspired by Cohen and Levesque's work which defines Speech Acts as rational actions. Improvements on that work are given by Sadek (as described in [Chaib-draa and Dignum, 2002]). The resulting Agent Communication Language (ACL) is described in [FIPA ACL, 2002]. FIPA specifies (among other things) meta info like the "speech act", sender or language used, independent from the content, (which could be; for example, in BML).

We define agents as entities communicating using BML messages in combination with performatives. Relevant performatives are formally defined for use in combination with a C2LG-based BML.

2.4 Agent Behaviour

In this paper we will focus on *intelligent agents*, which is a subclass of agents. We call our agents intelligent because they have some sort of explicit model of the environment and can deliberate about their actions. In other words they have a model of the cognitive processes humans use to describe their mental behavior. The ability of commanders to send, receive and interpret orders was modeled in these agents. Defining behavior formally in a declarative fashion is another useful part of agent technology. Here we refer to the work of describing agent behavior in the terms of Beliefs, Desires and Intentions logic (BDI) [Cohen and Levesque, 1990]. What is striking in the BDI theories on agents, is that agents do not simply follow orders or perform tasks, but they try to fulfill goals based upon their own beliefs which could include received orders and reports as wells as a priori doctrine knowledge. BDI logic describes mental attitudes in formal logics. From these logics several agent development environments for describing and implementing the mental attitudes of software agents have been developed. Examples are Jason [Bordini and Hübner, 2006], Jadex [Braubach and Pokahr, 2007] and "A Practical Agent Programming Language" (2APL) [2APL, 2007]. All of these programming languages already have a way facilitating or interpreting speech act like communication.

Defining behaviors in goals that need to be achieved has a clear resemblance with mission command as used in modern C2 doctrines (see for example [RNLA, 2001]. Although complex endeavors complicate matters considerably, for a commander it is more or less clear how he should interpret messages, because he has knowledge of the real world, military doctrine, social skills etc. The commander has had years of training and experience to behave in the right way for bringing mission to success. In creating flexible agent behaviour (Mission Command) it is important to capture the reasoning a

commander follows during his decision-making. Therefore we have to make a model of how a commander achieves his goals. There are different ways to do this, highly depending on what you want to be able to do with the model. As described, we want to use it in decision support processes for planning complex endeavours. It is thus important to explicitly model the reasoning steps so we can trace why a decision is made.

We have chosen to implement agent behaviour modelling with 2APL. After creating rules for every intelligent agent, the 2APL engine can be run, which tries to resolve plans in a cycle and respond to its environment. Note that 2APL only defines generic terms like goals, plans, procedural rules, so all behaviors, doctrines as well as 'common sense' reasoning, have to be implemented in these terms. 2APL offers a kind of inheritance mechanism, so that base classes can contain simpler behaviors that higher classes can specialize.

3. Proof of Concept (PoC)

In order to evaluate the concepts of pluggable agents and determine if they can be implemented to improve agility of a C2 planning process we developed an implementation of the agent architecture described in Chapter 1.5.

3.1 Simulator and other components

As a simulator we used Kibowi MP [Kibowi, 2007]. This is a simulator developed by TNO based on the current Kibowi trainer used by the RNLA (Royal Netherlands Army) for training commanders and their staff on battalion, brigade and division level. Kibowi MP's instruction set matched the C2LG language very well and we had availability of the source code of Kibowi and access to the original Kibowi development staff.

We chose the 2APL Agent platform (see [Dastani et al, 2007] and [2APL, 2007]) to implement our PoC. 2APL constructs (Goals, Procedural rules and Plans) closely follow the formal definitions of BDI which match well with the mission command domain.

We used a schema based upon the C2LG Grammar for orders and extended the language for reports, since in the version used these were not fully specified. We also added an ACL based header structure to the messages to this. The C2LG was used to design the schema of the Joint Battle Management Language – JBML (see [JBML, 2007])

A library was written in Java to create, construct and query the BML messages used and exchange them in XML. For distribution of these XML messages a publish and subscribe message bus was used. We believe that a future version could include a BML database plus web services to also push and pull messages using web services, as done by NATO MSG048 [Mevassvik et al, 2008] and [Reus et al, 2008].

Operators could intervene through the Kibowi interfaces, however in the PoC we had the aim that the agents would be able to execute the commander's plan without operator support.

3.2 Scenario Utilized

To demonstrate our Multi Agent System (MAS) we made a simple scenario where both different choices and adaptive behavior would be illustrated, challenging both the BDI concept and the BML language.

In this scenario we depicted a contemporary situation in which we could demonstrate the concepts explained above. Key factors were coordination of communication and actions by using ordering, reporting and some simple decision-making.

We designed the PoC to evaluate if the agents could react to new events only by using the BML messages they received, just like a commander only receives messages about the status of the battlefield. We avoided much dependency on the terrain for decision-making, since this would mean that every Command Agent would need to have an interface to the simulation terrain database. This would have complicated the PoC, though we believe it is an inevitable extension for the future. We also kept the landscape simple, with just a few woods, roads and rivers.

The (Order of Battle) ORBAT of the blue forces is depicted in Figure 2. This is a fairly new organizational form, where the goal is to optimize the coordination of many diverse types of indirect fire. The team of Fire Support Officers (Team FSO) must coordinate the artillery in this scenario. We simplified the ORBAT, since the Joint Fires Cell is normally located at the Battle group level so it can also have access to higher units, for example a squadron of planes. Most important to us was that fire is delivered with the right means at the right place based on the reports of the reconnaissance unit.

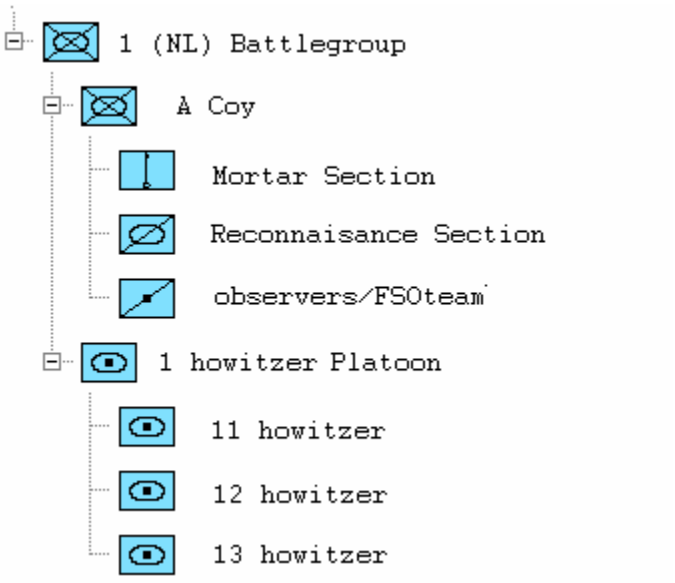

Figure 2. Order of Battle in the PoC

The red forces are not as well organized as the blue ones, typically irregular troops. In our scenario they do not use any coordination; they will simply fire at the blue forces when they detect them.

3.3 Chosen Course of Action

At the beginning of the scenario the Alpha company (abbreviated as "A Coy" in figure 2) command has a simple plan: he wants to perform a reconnaissance of the western part of the theatre. A Reconnaissance Section (RS1) is selected to complete this task. The RS1 is told to report any vehicles on the way. RS1 itself is not to engage in a firefight, in order to not compromise further reconnaissance .

When an enemy is spotted by RS1, the group will stop moving to prevent contact. A report is given by RS1 to Alpha command. The Alpha command can decide whether to authorize a call for fire or reroute RS1. When the call for fire is given to the Team FSO, the Team FSO then decides based on the reported location and observed unit type which fire is to be deployed (more factors could involved, but for our demo we limited the variety of used reports). In some cases it is decided the means can be the mortar group and in other cases the howitzer platoon. Then orders are given to the chosen group. Other means could be added (e.g. a fighter plane or even firing from a naval platform) but this was not done in the PoC. If the unit is destroyed or immobilized, RS1 continues its reconnaissance task. All of the information is exchanged using BML.

Figure 3 gives an overview of the initial position of the units at the beginning of the scenario.

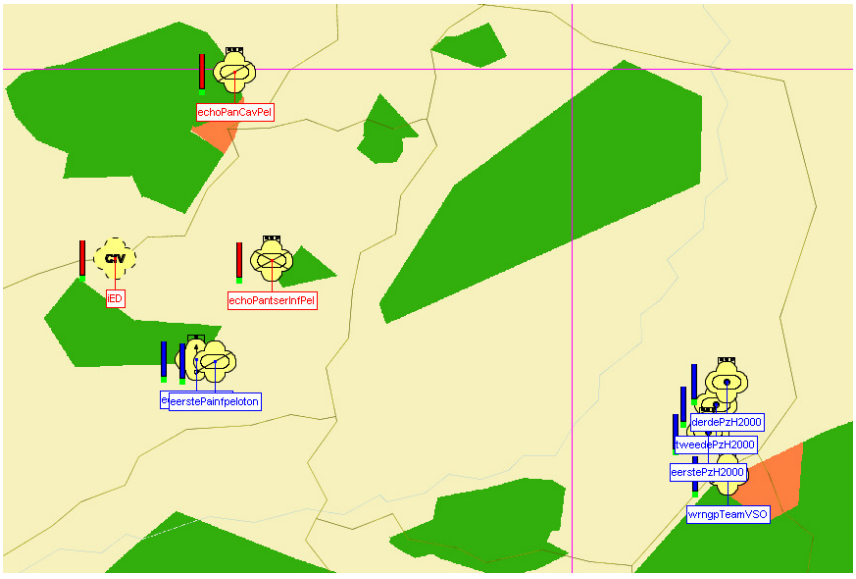

Figure 3. Initial setup from Administrator view

This view is the Administrator view from Kibowi, showing all units as neutrals.

3.4 Multi Agent System construction

We chose to model the behavior of the echelons in intelligent agents independent of the simulator used. This paragraph describes how our MAS implements a direct connection to the simulator.

Using the abstract view of agents above we specify our MAS as follows. Given an ORBAT O with n echelons that can be controlled, we have n+1 agents: A_x is the Agent representing the command of echelon x. In our experiment the ORBAT exists of 8 echelons (see figure 2) so we build 9 agents. The agent A_{SAF} is the Agent directly connected with the simulator. A_{SAF} is not an intelligent agent. Its purpose is to handle simple orders and create simple reports, thus linking A_x to the "real" unit in the simulator. Simple means that these reports and orders belong to the subset that is defined in the "ontology" described below. Simple also means that any Agent A_x can only receive reports from A_{SAF} with unit x as reporter. And that A_{SAF} will only handle orders with the same sender, tasker and taskee. For every unit in the ORBAT, including the commanding units, a Command Agent was build in 2APL and attached to the entity in the simulator. Note that we did not connect with a C2: The initial mental state of the highest command agent was as if it just received the C2 plan.

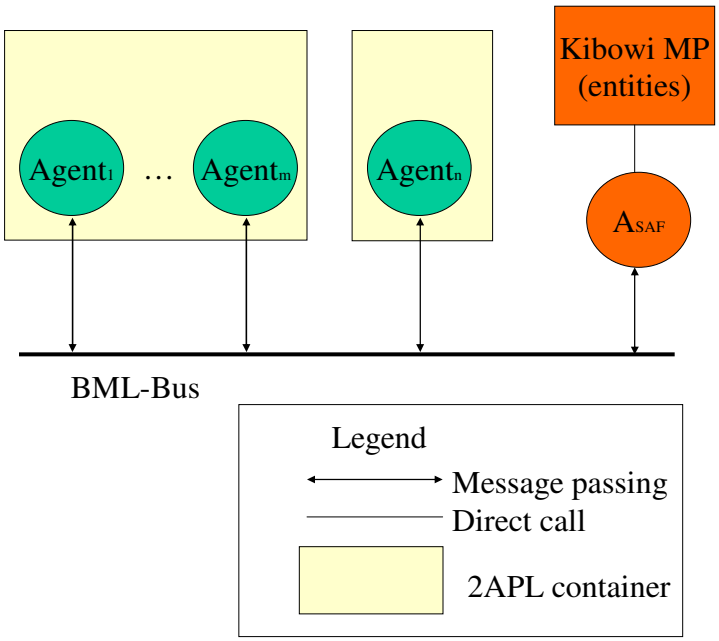

Figure 4. Abstract view of the MAS

In the simulator the units of the ORBAT are represented as simulation entities in a conventional way as used in a constructive training. We can view the A_x agents as the "brains" or command agent that control the "body" entities in the simulator, just as a human operator would control a single unit in a constructive training. The task of the command agent is to interpret and decompose complex orders from the C2SS or other agents to simple ones.

All lower level behavior is implemented in the simulator. In this way only a limited set of tasks like *move* and *attack* have to be implemented in the A_{SAF}. These orders will be implemented as partial BML orders, since the "bodies" inside a simulator will typically not use all the constituents of the C2LG Grammar. More details about how messages are interpreted can be found in the next paragraph. In this way, the simulator BML interface and indeed the simulator control logic itself can be kept relatively simple.

The simulator will also generate events, which A_{SAF} will translate to BML messages. The description of the interpretation and creation of messages by A_{SAF} is a sort of ontology, since it links "real" events to the keywords in the message. The command agent subscribes to the events and can respond to the reports, possibly aggregating reports and sending them to higher echelons.

As an example, say we have Agent A_{RS1} and A_{Alpha} respectively representing the Alpha company commander and the Reconnaisance Section (RS1) commander. A_{Alpha} cannot directly move RS1's entity, since it is not its "command agent". Only Agent A_{RS1} could send an order with the BML task "MOVE" to A_{SAF} to actually move RS1's entity in the simulator. If the Alpha Company command wanted the RS1 to perform a reconnaissance, A_{Alpha} would send an order with a "RECONNAISANCE" task to A_{RS1}. A_{RS1} will (under the right conditions) adopt this task as a goal and thus consequently send the above mentioned "MOVE" order to A_{SAF} and carry out other actions and plans required for a reconnaissance. When the entity spots enemies, reports are created by A_{SAF} and sent to A_{RS1}. A_{RS1} might decide to pass these reports to A_{Alpha}.

3.5 Towards Formalizing BML

Using a language in a sensible way can only be done if we attribute some meaning to the utterances. Preferably everyone should attribute the same meaning. An ontology links language to objects and actions in a given domain, thus giving meaning (for more detailed descriptions see [Blais et al, 2006]. The JC3IEDM (Joint C2 Information Exchange Model) can be viewed as the semantics of BML. These semantics map the use of words to military doctrine described in English. Unfortunately our agents understand neither English nor doctrine. The C2 agents representing units only "understand" BML and the SAF agent only "understands" BML and direct simulator actions/calls. Therefore we will briefly describe the way BML utterances are mapped to simulator actions. This will be an abstract specification for any SAF agent that is used with our command agents.

In our example a BML message is passed between two C2 agents where the action to be performed is a "RECONNAISSANCE" by the Alpha unit. But there are many ways in which a reconnaissance can be done. The JC3IEDM does not "pin down" how to do this.

There are two solutions: 1) micro management – all actions for the reconnaissance are written down in a plan, which is then given to Alpha command. This is not according to military practice and will also lay a burden on the higher command and eventually the C2 users. 2) mission orders – give a standard order for reconnaissance in combination with an ORBAT and leave the decision making to Alpha Command. Alpha Command can then choose its own course of action within the boundaries of the order given and execute

that. It can even change the course of action during the reconnaissance, e.g. when enemies are discovered. A description of this would be a functional report of our agents behavior. In the end a subject matter expert should be able to judge such an agent description just as he would judge a chosen CoA from for example a real mission.

As example of an informal ontology we made the functional description of our A_{SAF}. It consisted of a number of statements like the ones below. Each statement links a BML sentence to an effect or action in the simulation. Take for example a variation on a C2LG statement 2b, 2b*, taken from our scenario.

(2b*) OB → ***move*** **RS1 RS1** Route-Where Start-When (End-When) Why Label (Mod)*

This would have the following effect in Kibowi:

Effect: The RS1 unit executes a "move over ground" to the coordinates provided by the Destination that is a mandatory part of in the Route-Where. The SAF sends a task report at the beginning of the execution and at the end it will send a status report with regarding instantiated to "POSITION" that tells the current position.

Where orders link to an action, reports are linked to events in the simulator. As described in the C2LG grammar for reports, reports can be given by the agents in a protocol as described above or after an order-to-report [Schade & Hieb, 2007]. Consider the ontology statement for 6c as an example.

(6c*) RB → Status-Report ***HOSTILE POSITION*** **bogey_1** Where When **FACT** Label (Mod)*

This report is given on the following events (when ordered or in protocol).

Event: -First Time a hostile unit comes in observation range of the unit.

-Every time a hostile unit in observation range changes its identification status.

-Every time a hostile unit in observation range changes status (e.g. when it is destroyed)

Doctrine knowledge how to respond to messages can be expressed in 2APL rules. In 2APL, "Plan Goal" Rules are used to activate plans when the conditional belief, called *Guard*, becomes true. Apart from this, 2APL offers Procedural Rules that state what to do at this moment. Some of these only consist of sending an order to the A_{SAF} like in 2b*. Some also adopt new goals and update the beliefs of the agent. 2APL also offers Plan Repair Rules, which can be used to adapt failed plans. For example; a plan to execute an order fails if the agent does not have the capability for the specified task. The plan will than be replaced with the plan to send a "REFUSE" message to the tasker. The inheritance functionality also helped us reuse behavior. For example; most rules that concern checking if an order can be executed by this agent, are put in a default

commander agent program. Since all agents should be able to react to orders, they all inherit these rules and goals. To give an idea of the scope of the agents behavior we present the number of rules used in the PoC. The default commander program consisted of 4 Plan-Goal (PG) rules, 8 Procedural (PC) rules and 2 Plan-Repair(PR) rules. On top of these rules all agents get specific rules for their unit. The most complex agent was $A_{A\ Coy}$ representing the Alpha company command. This agent has 11 PG rules (most of which we expect to be reused in other company command agents in future experiments) as opposed to A_{RS1} with 5 PG and 4 PC-Rules. This corresponds with the intuition that this agent (at a lower echelon) should execute more orders in comparison to $A_{a\ Coy}$ that has a more reasoning/goalkeeping purpose.

Below we give an example of the 2APL code used.

```
Goals:
defend(building001, where(coordinates(52.2367, 5.3729 ,25)), when(at, 1300))

PG-rules:
defend(Target, where(Coordinates),When)  <- executing([]) and not near(Coordinates)
                                  |{
                                  adoptGoal(advanceTo(where(Coordinates)))
                                  }

defend(Target, where(Coordinates),When)  <- executing([])  and near(Coordinates)
                                  |{
                                  adoptGoal(holdOffensive(where(Coordinates)))
                                  }

advanceTo(where(DestCoord)) <- executing([])  and curtime(Now) and myPosition(MyCoord)
                                  |{
                                        @apaplAgent(calculateRoute(MyCoord, DestCoord,RouteWhere );
                                        execute( task(    who(me),
                                                          what(advance,advanceDestination),
                                                          RouteWhere ,
                                                          when(at, Now) ,
                                                          taskID(advance) ))
                                   }
```

Figure 5. PlanGoal rules in 2APL

Planning Goal Rules in 2APL take the form

GoalTerm *G* <- BeliefTerms *Guard* | PlanTerm *P*

Guard is a technical term for a condition that must become true to enable the PGrule, because it is possible that there are different rules for achieving the same goal. If an agent with this rule has goal *G* and the *Guard* becomes true he will execute plan P.

Figure 5 shows a fragment of 2APL code for a company agent. In the notation used here variables start with a capital, these will be instantiated with facts when running the program ((see [Dastani et al, 2007] and [2APL, 2007] for more details on this). The instantiated goal of this agent is to defend "building001" at 1300 hours. When the agent

can conclude from its belief base that his unit is near the building in other words "**near**(Coordinates)" is true it will select the first PGrule and hold that position offensively. Otherwise the guard of the second PGrule becomes true and a new Goal "**holdOffensive**(**where**(Coordinates))" is adopted. In the PGrule for that goal another program "apaplAgent" will be called to calculate a route called "RouteWhere" leading to the destination. When the advance task is executed the agent's unit will be at the location of the bridge and can execute the plan for the offensive hold in a similar matter.

This architecture with the agents described could complete the simple scenario only using BML. Meaning the Alpha Company reacted to enemies in its line of sight with the appropriate fire. In some runs the enemy was too quick with its reaction and the RS1 had to engage in a fight. This usually ended badly since the opposing vehicles were stronger. But this only illustrates that the scenario was not such that the blue forces would always win.

Our agents were flexible since they did not depend on specific data – such as the exact locations of the enemy or own units. The approach was, however, very dependent on the scope of the BML, which was extended from the initial C2LG for the PoC.

An important point is that much of the behavior could be reused given our agent architecture, but in the POC we made every rule, as this was the first implementation . To test the reusability we plan to do at least another experiment and invite others to use (parts of) our architecture. The scenario takes 10 to 15 minutes in real time to execute. This could be speeded up at least 300%, but the limiting factor here was our simulator.

4. Discussion and future research

The PoC tested whether the combination of BML and Intelligent Agent technology leads to a more flexible kind of decision support tool, potentially useful for improving C2 agility. The system built for the PoC and the scenarios that were run on it yielded some interesting points for discussion compared to the requirements.

4.1 Analysis of the requirements

The PoC addressed a number of the technical and agile requirements discussed in Chapter 1.1.1.

The issue of misinterpretation of a COA is addressed by the use of a formal language, the C2LG in combination with an ontology. The use of an open standard and clearly defined meaning for a future BML will allow agents and simulators to perform military commands that more exactly match intent.

The PoC includes command agents at different echelons and at a different fidelity. This prevented micro management. Instead high level orders could be given that were translated to low level orders by the agents themselves.

No human operators were used at runtime in the PoC. The agents were powerful enough to run the scenario by themselves.

The architecture of the PoC is pluggable. All information exchange in the scenario is based on C2LG. Agents do not run inside the simulation, but can be added as loosely coupled components.

The PoC can be run at any speed, as long as the agents and the simulator have enough computing power at their disposal.

The other technical requirements mentioned in chapter 1 not addressed by our PoC, are the rapid initialization of a simulator from a C2SS and the interactions with human operators. Both issues could be tested by integrating with a C2LG editor like the one presented in [Reus et al, 2008].

The agile requirements in Chapter 1.1.2 are also illustrated by the PoC.

The PoC builds on simulated commanders (and other agents) behavior instead of using simplified patterns. This can lead to unexpected interactions instead of "pre-programmed" patterns. Of course, the commander's decision making is a simplification of the real decision making process, but even with a few simple decision rules, complex interactions can take place.

For the same reason, the command agents can respond and adapt to external influences and events in the form of reports and make decisions based on their doctrine.

The PoC provides insight into the results of actions, but does not actually analyze the results. Another layer might be added that could run a simulation for multiple conditions and draw more certain conclusions.

The PoC modeled various military players as agents, which differ in doctrine and organization. The C2LG behaviors used were based on the JC3IEDM. When non-military players would be added, the vocabulary might need to be extended for additional actions and reports typical for these non-military players.

4.2 Other issues

BML is designed as a language for commanding units. We found that it was quite compatible with Kibowi. The information needed for the entity actions was available in the C2LG orders and the events could produce the C2LG reports. We expect that other tactical simulations can be adopted to work with BML with not too much effort.

Work on formalizing BML is still under development. We described how our SAF agent links simulation truth to BML words and sentences. Formalizing this would speed-up building SAF agents for other simulators and simplify discussions with Subject Matter Experts on the meaning of words in the language.

In the PoC we used a limited set of reports and orders as described in the C2LG. It is expected that the C2LG will evolve in areas such as commander's intent, requests and order interdependencies [Schade & Hieb 2008]. More powerful CoAs can then be exchanged than the one presented. Another important extrapolation of the reasoning capacities is by providing agents with all the information available in a (future) Common Operational Picture, including terrain products such as Geospatial BML produces [Hieb et al, 2006]. The availability of this information should of course reflect the availability of this information to the military player.

Declarative, rule-based, programming such as with 2APL provides a scenario and doctrine independent method to produce command agents. Using 2APL, however, requires some good craftsmanship and quite some effort to make sure sensible behavior is obtained. This involves both programmers and military experts.

5. Conclusion and recommendations

We have presented an architecture that is highly modular, resulting from the combined use of BML and Agent technology. With this architecture we can combine existing simulators with newly build agents tailored for decision support. Our PoC has shown that the proposed architecture addresses many of the requirements that would make simulation a more useful tool for C2 (especially in the key capability of agility).

Adapting an existing constructive simulation to the C2LG BML proved relatively easy. A very powerful concept is that the agents implemented are both simulation system and C2SS independent. These agents communicate through the emerging C2LG BML standard only. Thus agents can be supplied by any third party and could be easily interchanged.

The agent technology that we used to build the simulation is very suitable because of its flexibility in defining behavior and its advanced communication technology. Much more work is needed to implement specific behavior.

We believe that the combination of agent technology and C2LG is a powerful one. Of immediate interest are the FIPA ACL and Intelligent Agent concepts. Standards and open source software are already available, which can accelerate the use of agents and BML in an agile setting for decision support.

References

Alberts, David S. and Richard E. Hayes, Power to the Edge: Command, Control, in the Information Age. CCRP, 2003.

Alberts, David S. 2007. The Future of C2:Agility, Focus and Convergence. 12th ICCRTS June 2007

2APL website: <http://www.cs.uu.nl/2apl>, 2007.

Blais, Curtis, Chuck Turnitsa, and Per Gustavsson, "A strategy for ontology research for the coalition battle management language product development group," in Proceedings of the Fall Simulation Interoperability Workshop 2006, 2006.

Bordini, Rafael H. and, Jomi F. Hübner. BDI agent programming in AgentSpeak using Jason. In Proceedings of the Sixth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA VI). CLIMA, June 2006.

Borgers, Erik, Wim Huiskamp, Nico de Reus, and, Jeroen Voogd. Research and development towards application of MSDL and C-BML in the Netherlands. In Proceedings of the 2007 Spring Simulation Interoperability Workshop, 2007.

Braubach, Lars and, Alexander Pokahr. Goal-oriented interaction protocols. In Fifth German conference on Multi-Agent System TEchnologieS (MATES-2007), 2007.

Carey, Scott, Kleiner, Martin, Hieb, Michael R. and Brown, Richard, "Standardizing Battle Management Language – A Vital Move Towards the Army Transformation," Paper 01F-SIW-067, Fall Simulation Interoperability Workshop, 2001.

Chaib-draa, Brahim and, Frank Dignum. Trends in agent communication language. In Computational Intelligence 18 (2), 89–101., volume 18, page 89–101. Blackwell Synergy, 2002.

Cohen, Philip R. and, Hector J. Levesque. Intentions is choice with commitment. In Artificial Intelligence. Elsevier Science Publishers Ltd, 1990.

Dastani, Mehdi and Dirk Hobo, and, John-Jules Meyer. Practical extensions in agent programming languages. In Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems. AAMAS'07, ACM Press, 2007.

Doctrine committee of the Royal Netherlands Army. AFM 1 Command and Control, chapter 3: Command and Control and Doctrine, pages 44–48. Doctrine Committee of the Royal Netherlands Army, 2001.

Federation for Intelligent Physical Agents. FIPA ACL Message Structure Specification, sc00061G edition, 2002.

Federation for Intelligent Physical Agents. FIPA Communicative Act Library Specification, sc00037j edition, 2002.

Foundation for Intelligent Physical Agents, FIPA Agent Management Specification, 2002.

Grisogono, Anne-Marie and Damien Armenis , Mission Command needs the Adaptive Stance, Complex'07, Defence and Security Track, July 2007, Gold Coast

Grisogono, Anne-Marie. The Implications of Complex Adaptive Systems for Command and Control CCRTS, San Diego, 2006

Hieb, Michael, Michael Powers, J. Mark Pullen, and Martin Kleiner. "A Geospatial Battle Management Language for Terrain Reasoning," In Proceedings of the 11th ICCRTS, Cambridge, UK, 2006.

Hieb, Michael and Ulrich Schade. "Formalizing Command Intent Through Development of a Command and Control Grammar." In Proceeding of the 12th ICCRTS, Newport, RI, 2007.

IEEE 1516-2000 - Standard for Modeling and Simulation High Level Architecture - Framework and Rule 01-May-2000

JBML website: <http://netlab.gmu.edu/JBML/BML>, 2007.

Kibowi website: <http://www.kibowi.com>, 2007.

Mevassvik, Ole-Martin, Nico de Reus, J.Mark Pullen, Scott Carey, Nicolas Cordonnier, Lionel Khimeche, Ulrich Schade, Nanne LeGrand, Sergio Galan, Sabas Gonzales Godoy, Michael Powers, Kevin Galvin. 2008. NATO MSG-048 Coalition Battle Management Initial Demonstration – Lessons Learned and Way Forward. Paper 08S-SIW-082 presented at the Spring Simulation Interoperability Workshop, April, Providence, RI.

MIP website: <http://www.mip-site.org>, 2007.

Schade, Ulrich and Michael Hieb. Formalizing Battle Management Language: A Grammar for Specifying Orders, Paper 06S-SIW-068, Spring Simulation Interoperability Workshop, Huntsville, Alabama, 2006.

Schade, Ulrich and Michael Hieb. Development of Formal Grammars to Support Coalition Command and Control: A Battle Management Language for Orders, Requests, and Reports. In Proceedings of the *11th ICCRTS,* Cambridge, UK, September 2006.

Schade, Ulrich and Michael Hieb. Battle Management Language: A Grammar for Specifying Reports. In Proceedings of the 2007 Spring Simulation Interoperability Workshop, 2007.

Schade, Ulrich and Michael Hieb. A Linguistic Basis For Multi-Agency Coordination. To be published in Proceedings of the *13th ICCRTS*, Bellevue, WA, June 2008.

Searle, John Roger. Speech Acts: an essay in the philosophy of language. Cambridge University Press, 1969.

Searle, John Roger. Expression and Meaning: Studies in the Theory of Speech Acts. Cambridge University Press, 1979.

Reus, Nico, de, Paul de Krom, Ole-Martin Mevassvik, Anders Alstad, Ulrich Schade, and Miloslaw Frey. BML Enabling of National C2 Systems. In Proceedings of the 2008 Spring Simulation Interoperability Workshop, 2008.

Wooldrigde, Michael. An Introduction to MultiAgent Systems. John Wiley & Sons, 2005.

Appendix A: XML Structures Used in the PoC

We based the BML message passing on the C2LG and Agent ACL principles. We chose to pass XML based messages. The main structure consist of an ACL based header plus content derived from the C2LG and more specific, JBML.

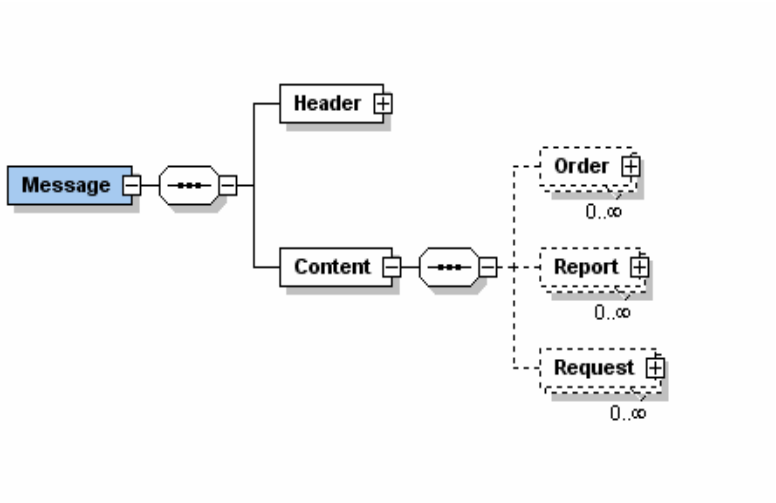

Figure A1: Top level view the BML Structure

An ACL header was designed based reusing the C2LG message header already available. We added tags for Performatives, Language and Ontology, coming from ACL.

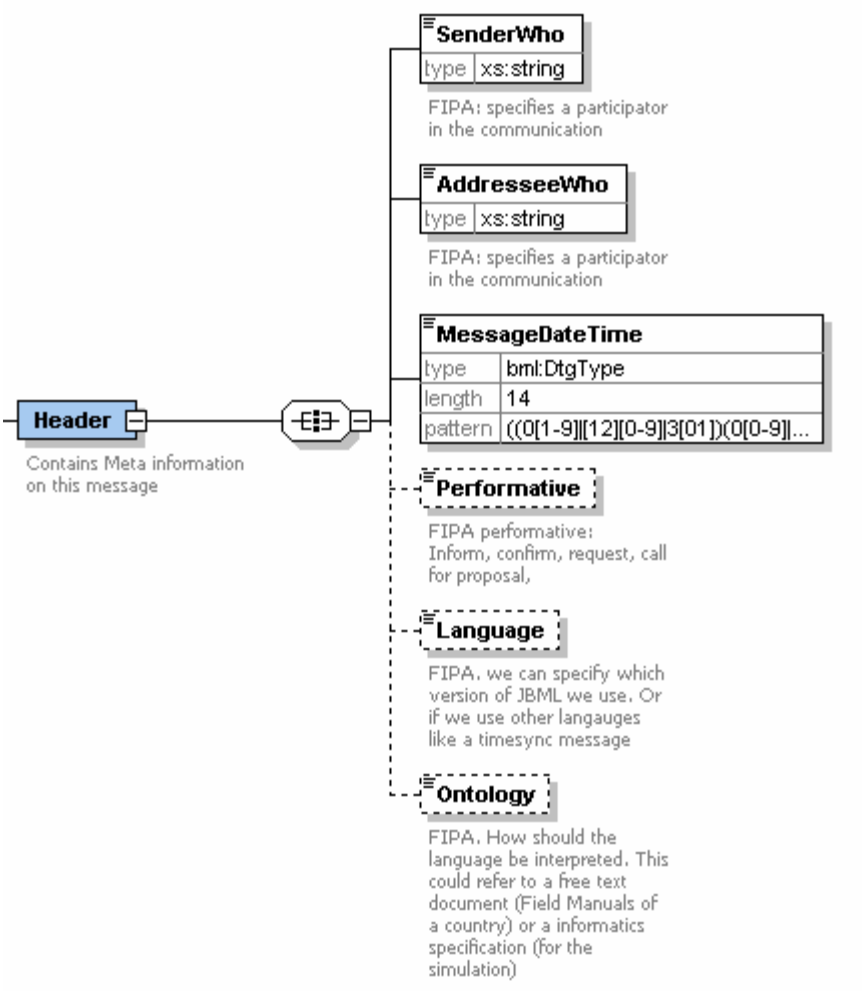

Figure A2: Header Structure.

These parameters are described in more detail in the FIPA ACL Message Structure Specification [FIPA ACL, 2002]. The SenderWho and AddresseeWho in this part of the header are the agent names to send messages to. This is opposed to the Tasker and Taskee, which in our configuration can only be names of units specified in the ORBAT. The language parameter indentifies the BML language used in the message. The ontology should specify how the language should be interpreted. In our experiment we have not yet used this parameter. Although we did specify what our SAFagent does, this ontology will be in free text, and thus not really useful for runtime agent reasoning. A free text ontology is however instructive when this kind of C2 agents are to be used in combination with other simulators.

The JBML community has already constructed XML messages for Orders, based on the C2LG. We decided to reuse these definitions for JBML version 15, with some small additions for our experiment.

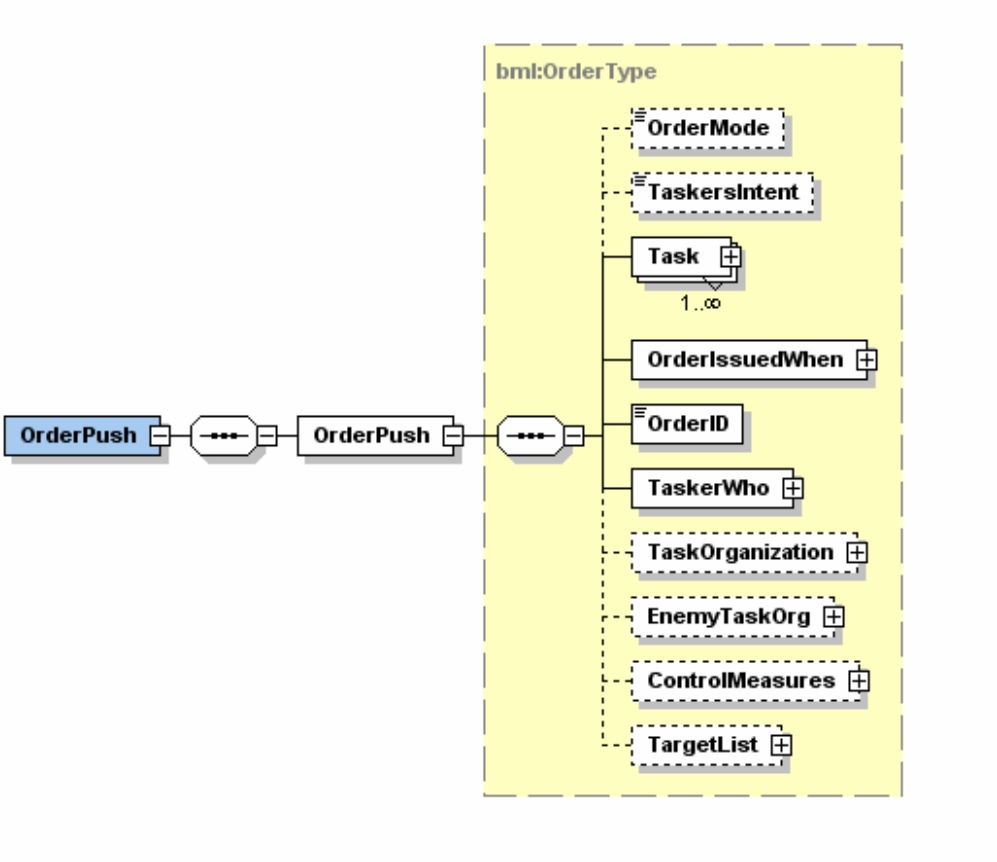

Figure A3: JBML order structure for pushing an order in the database

A more 'conceptual' change is that JBML was designed to be used with web services for pulling and pushing orders into an adapted JC3IEDM database. In our experiment we worked with a message server, which fitted the architecture chosen. The OrderType as used by the OrderPush was reused with a new <Order> tag instead of the <OrderPush> tag. No further changes were needed, which indicates that the structures of JBML can also be used for other purposes then database access.

We designed XML versions of Status and Task reports based the C2LG according to Schade and Hieb's language design in [Schade & Hieb, 2006a; Schade & Hieb, 2007]. Event reports where not used in the scenario. For the actual XML, we reused the JBML structures. The reports have the following main structure:

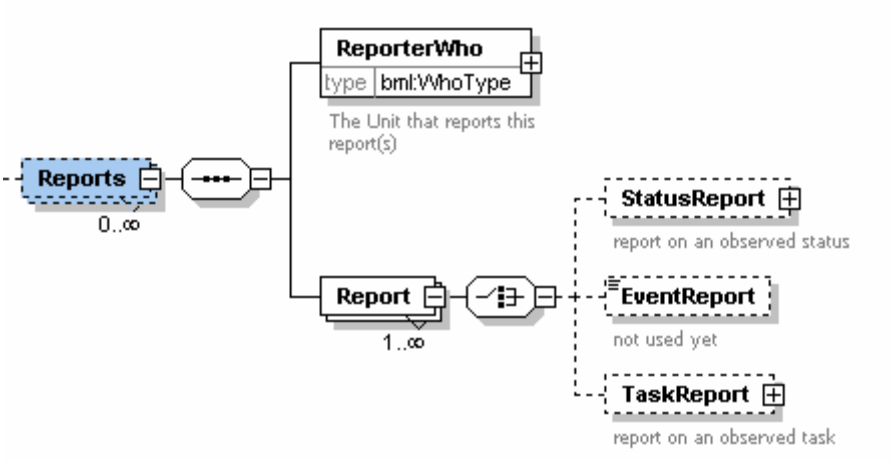

Figure A4: Report Structure

Task reports have the capability to report on ongoing tasks, for example an attack, either of the own party or the observed enemy. In the latter case, the enemy can be classified in different ways: e.g. by its organic unit number, but also by size or equipment (see [Schade & Hieb, 2007]), in line with military reporting practice. The Status reports can be used to report on the status such as its position, the status of its material etc.

These structures used are as follows:

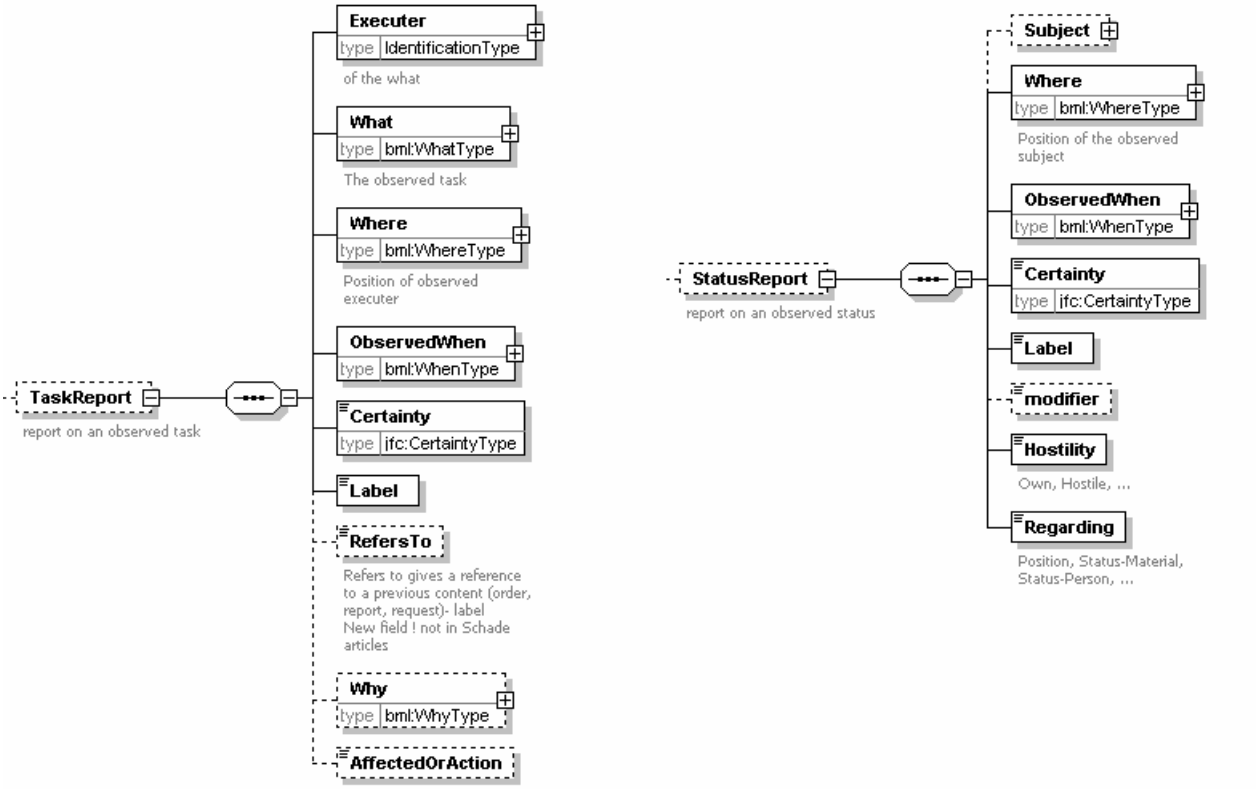

Figure A5: C2LG Task and Status Report Structures

Almost all the XML types from the BML name space have been taken from the JBML 1.5 XSD from George Mason University (GMU), see [JBML, 2007]. In the figure above the only new type we added is the jfc:CertaintyType which contains values like “FACT”, “PLAUSIBLE”, compatible with those suggested in [Schade and Hieb, 2007].