Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

APPLICATION OF BML TO INTER-AGENT COMMUNICATION IN THE ITSIMBW SIMULATION ENVIRONMENT

Philipp Hügelmeyer

Fraunhofer IAIS
Schloss Birlinghoven
53754 Sankt Augustin, GERMANY

Ulrich Schade

FGAN – FKIE
Neuenahrer Str. 20
53343 Wachtberg, GERMANY

Thomas Zöller

Fraunhofer IAIS
Schloss Birlinghoven
53754 Sankt Augustin, GERMANY

ABSTRACT

In this contribution we analyze communication requirements of multi-agent simulation systems using ITSimBw – developed at Fraunhofer IAIS – as an example. A focus is put on issues concerning inter-agent communication but complementary aspects of user interaction and coupling with C2 systems are also discussed. We propose an augmented version of the battle management language BML as a communication protocol that perfectly matches our communication requirements both syntactically as well as on the semantic level. We furthermore explain how such BML messages are processed by our system.

1 INTRODUCTION

The end of the so-called cold war era has confronted the member nations of NATO with new challenges concerning their security policy. Instead of a clear threat with well-defined geographical and political attribution, known organizational structure and established doctrinal principles we now face multifaceted risks. The accretion of fundamentalists religious beliefs accompanied by an increase in terrorist activities also contributes to the security political situation.

The German armed forces reacted to the challenges posed by these geopolitical factors by instantiating a transformation process which continuously identifies and remedies capability deficiencies. Due to the fact that the availability of experience values concerning the variegated risks is limited, the importance of modeling and simulation grows steadily. To this end, the IT office of the German armed forces funds research and development efforts for the generic, versatile simulation environment ITSimBw.

The spectrum of possible application areas encompasses analysis and planning, concept development and experimentation (CD&E), procurement management, education and training, as well as support for the military commander in action. Orthogonal to the application areas are a number of additional factors like spatial and temporal resolution or scenario size that determine the class of educible models.

A simulation environment that supports all application areas while accommodating the aforementioned influence factors has to satisfy a large number of requirements, whereof flexibility and ease of use are of prime importance. Thus, ITSimBw is based on an open architecture which provides a number of services, e.g. editors, viewers, and line of sight computation, for independent use. The simulation core itself consists of an execution environment for autonomous software agents that model active entities in simulation tasks. Environment instances can be dispersed across a computer cluster, thus enabling distributed simulation. Furthermore, in order to support applications in education and training, ITSimBw allows user interaction in the sense that simulated units can be commanded during simulation runs. Thus, human vs. human or human vs. computer-generated forces scenarios can be explored. Future work will also address the coupling of real-world units and equipment to simulated counterparts, thereby allowing a integration of real and simulated units in force maneuvers.

Taking into account both the multi-agent system architecture as well as the supported application domains outlined above, the need for reliable, unambiguous communication is clearly a major concern. It arises in a variety of specific areas:

- Inter-agent communication (possibly across different computers)
- Modeling of message exchange in command and control chains
- User interaction with simulated units
- Interaction with real units in mixed simulated / real-life war-games

Despite the fact that this contribution will focus on the aspect of inter-agent communication, the proposed solution should be ideally suited to cover the remaining areas equally well. In our opinion, a suitably adapted variant of the Battle Management Language (BML) provides the perfect match for our communication requirements.

BML is defined as an unambiguous formal language used to command and control forces and equipment conducting military operations and to provide for situational awareness and a shared, common operational picture (Carey et al. 2001). It is supposed to be used for military communication among C2 systems and their users as well as between C2 systems and simulation systems or robotic forces. The standardization of BML is tackled by SISO (Blais, Hieb, and Galvin 2005) in coordination with NATO RTO MSG-048 "Coalition BML."

The remainder of the paper is structured as follows: First we will give a more detailed description of ITSimBw's agent model. In particular, the specific requirements faced in inter-agent communication will be derived. In the following section, we will give an overview on BML and its sub-grammars for orders, reports and physical effects – the latter being a particular consequence of our intention to base all inter-agent communication in ITSimBw on BML. Then, the implementation of the BML message exchange in ITSimBw will be discussed. We finally conclude and outline further activities concerning the use of BML in ITSimBw.

2 THE ITSIMBW ENVIRONMENT

In order to provide the necessary background for understanding the inter-agent communication requirements in ITSimBw, we will give a brief outline of the core ideas in the following three subsections.

2.1 Agent Focus

As has already been stated in the introduction, ITSimBw is designed from the ground up with an agent-oriented paradigm in mind. As a consequence of this approach, virtually every part of a simulation is modeled as an autonomous agent. This even includes the landscape, weather phenomena like clouds or rain, and inanimate objects such as bridges and other buildings. Although many of these agents are usually passive, they can be activated at any time if required by the investigated scenario. A simulation scenario is thus given by the society of agents forming its constituent parts (c.f. Weiss 1999 or Wooldridge 2005).

The aforementioned society of agents A is formed by a set of individual agents a_j, – also called actors in our terminology:

$$A = \{a_1, ..., a_j, ..., a_k\}.$$

Each actor a_j is composed of a set of typed attributes U_j, whose value assignment determines the agent's state, together with an action function, which entails all operations that can be performed by the actor.

$$a_j = (U_j = (u_{1,j}, ..., u_{m,j}), f_j).$$

Structural information concerning agents, i.e. names and types of attributes, is described via agent types. Formally, we have $type(a_j) = t$, if and only if agent a_j is of type t. The system supports agent templates that can be used to store prototypical value assignments. Thus, an individual agent can be created either by instantiation of its type, or by copying from a pre-defined template.

2.2 Behavior Description

In ITSimBw we use LAMPS (Language for Agent-based Modelling of Processes and Scenarios) for the specification of agent behaviour. This language has been developed – like ITSimBw – at Fraunhofer IAIS. It is based on high-level Petri nets (Jensen 1992). Thus, LAMPS inherently supports parallel and concurrent processes, and is not just an extension of sequential simulation languages like Maisie, Modsim II (Bryan 1989) or SIMSCRIPT III (Rice et al. 2005). Like other modern simulation languages (L'Ecuyer and Buist 2005), LAMPS can be displayed both graphically and as a rule-set.

According to its underlying formal model of high-level Petri nets, LAMPS consists of the following four basic concepts:

- *Places* hold states, i.e. they contain subsets of the agent attributes. The contents of a place is called token and can be of an arbitrarily complex structured type. Places can contain several tokens of different or identical types.
- *Actions* describe the effects that agents apply to themselves or to other agents.
- *Relations* denote the links between places, actions, and agents. Relations correspond to the arcs in the Petri net model.
- *Agents* correspond to the conditions of Petri nets. An agent in LAMPS observes the set of places that have relations to the agent's actions. Based on these places the agent decides which actions are executed and which parameters should be used.

A basic example of a LAMPS graph is shown in Figure 1. If a token containing unit name and location is put in the place *enemy spotted*, the agent *Inf A* will execute action *combat*. According to the outcome of this action, a token is written either to place *mission success* or *mission failure*. ITSimBw features a user interface, where behavior descriptions of this kind can be comfortably edited. This GUI also provides online syntax checks that aid the visual programming process. The resulting LAMPS-graph is then translated into executable code that can be run when the agent is executed. LAMPS fragments can be recursively combined allowing for modular assembly of complex behaviors from simpler parts. Clearly, not all actions can themselves be described as LAMPS-graphs. Those considered to be elementary are programmed as

Java classes with a common interface that essentially provides the method `call(params)` for action execution.

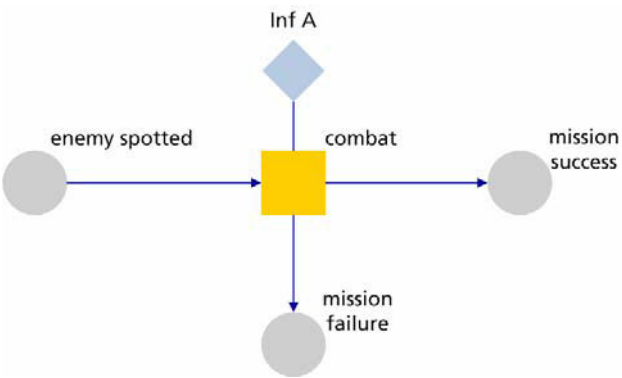

Figure 1: A simple LAMPS graph example.

2.3 Inter-Agent Communication

One of the constituting principles of agency is autonomy. This means that an agent is not directly acted upon by other agents, altering its internal state or executing its behaviors. Instead the agent itself decides when and how to update its internal state, or when and how to perform its behaviors. Consequentially, all interactions between agents are acts of communication. Messages are passed between agents requesting changes of state or execution of behavior.

Looking more closely at inter-agent communication, one can identify three main subject areas:

- orders,
- reports, and
- physical effects.

In military simulations, one commonly models agents at different echelons that interact according to the chain of command. Thus, orders are typically generated by the higher echelons and passed down to those agents at lower echelons, which in turn have to interpret and execute the given orders. Moreover, the latter agents are often confronted with the task of reporting back to the agents modeling higher echelon units or institutions. Additionally, even the impact of physical effects like weather influences or weapons fire do not affect agents directly. The receiving agent has to be informed about the presence of such effects, but then decides autonomously, how it is going to react. Thus also the mediation of physical effects is done via communication acts.

Therefore, a complete inter-agent communication protocol has to cover all three subject areas described above. Moreover, communication – especially between technical systems – is a multifaceted problem. One commonly distinguishes between the physical layer that is concerned with the hardware connection among different systems, the syntactic level that describes the structural format of the messages that are to be exchanged and finally the semantic level which is concerned with a common, unambiguous interpretation of the communicated content. A communication protocol typically solely defines the syntactic level, leaving aside the question of interpretation of the semantic content. This fact is particularly unfortunate as the achievement of a common understanding is really the core task of any communication act. The battle management language BML, which is described in the following section, has the desirable property that it both determines the syntax and – by virtue of tight coupling to doctrinal terms – also covers semantic issues.

Being developed for communication between C2 systems and robotic, simulated, or real forces, BML's original design contains means of describing orders and reports, but is lacking a comprehensive way of communicating physical effects. In this respect, this paper introduces the specific additions made to the core language, enabling it to serve as an ideal means for inter-agent communication in ITSimBw.

3 BML

BML is designed as a formal language. A formal language is the set of all expressions that can be generated by a formal grammar. In general, a grammar consists of a lexicon and a set of rules. The lexicon provides the words of the language, and the rules determine how to construct longer expressions, e.g., sentences, using these words.

In order to define BML, one has to provide the lexicon and the rules. BML has to convey information about military operations. It has been decided by the SISO project group and the NATO RTO MSG-048 that the attributes and values used in the JC3IEDM (Joint Command, Control, and Consultation Information Data Exchange Model; c.f. (MIP website)) should constitute BML's vocabulary. The JC3IEDM is the standard data model for C2 systems in NATO. The use of its terms in BML, therefore, supports the standardization and the integration of BML into C2 systems. It simplifies the mapping from BML expressions to data model entries.

Through a cooperation between the Center of Excellence for C4I, George Mason University, and FGAN-FKIE, Hieb and Schade, as members of the standardization groups, developed a formal BML grammar. They proposed rule sets for orders (Schade and Hieb 2006a, Schade and Hieb 2007), for reports (Schade and Hieb 2007, Schade and Hieb 2006b), and for the command intent (Hieb, and Schade 2007). These rules, together with the lexicon derived from the JC3IEDM, constitute a grammar for BML. In subsection 3.1, we will sketch the major aspects of this grammar and its rules, with a focus on the communication between a C2 system and a simulation system. Then, we will expand this grammar to cope with communication of physical effects in section 3.2.

3.1 The BML Grammar for the Communication between C2 and Simulation Systems

With respect to the communication between a C2 system and its users on the one side and a simulation system on the other side, conveying an order is of highest importance. For example, in a staff exercise, a simulation system can play the role of the forces if the commander is enabled to give the orders in such a way that the simulated forces react to them as intended. Also, the issuing of orders is a common phenomenon between agents modeling units at different echelons. Therefore, we start our short description of the BML grammar by discussing the rules for the formulation of orders.

The doctrine of ordering and 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 paragraphs 1) Situation, 2) Mission, 3) Execution, 4) Administration and Logistics, 5) Command and Signal, and the respective annexes. The third paragraph is used to “summarize the overall course of action,” “assign specific tasks to each element of the task organization,” and “give details of coordination.” It is the information of this paragraph that a simulation system has to interpret correctly so that the simulated forces can react in accordance to a given order. In BML, paragraph 3 is generated by

(1) OrderParagraph3 → CI OB* C_Sp* C_T*

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

The rules to expand OB have the general form as given in (2a). (2b) and (2c) give examples for OB rules.

(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)*

Verb in (2a) indicates that a tasking verb will appear in this position in a OB rule. In (2b) this tasking verb is “advance” and in (2c) it is “defend.” The tasking verbs are taken from the JC3IEDM. Each verb spans a frame, and (2a) shows the general form of this frame. Tasker is to be expanded by the name of the one who gives the order. Taskee is the unit that is ordered to execute the task. Start-When and End-When are to be expanded by temporal phrases when the task should start and end, respectively. They denote 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 the frame of each tasking verb.

Affected in (2a) has to be part of the frame if someone, e.g., the enemy, will be directly affected by the task; Whether Affected is part of the frame 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 (cf. 2c). It is not there otherwise, e.g., in the case of advance (cf. 2b). In this way the linguistic principles are implemented. Action is similar to Affected. It only appears, if the task affects an action, e.g., if the tasking verb is assist. The type of the Where is also determined by the verb. It is either an At-Where or a Route-Where. An At-Where denotes a location, and a Route-Where a movement into a location. A Route-Where can be expanded to sequences of spatial expressions as in “**from** LocationA **to** LocationD **via** LocationB **and** LocationC.”

An OB rule ends with Why, Label and the optional Mod. Why represents a reason why the task specified by the rule is ordered. The Why links a basic order expression to the Command Intent. Label is a unique identifier by which the task can referred to in other expressions. The optional Mod (for modifier) is a wild-card that represents additional information necessary to describe a particular task, e.g., to specify a particular formation for an advance.

Let us assume as an example that the Multi-National Division W commands the 13th Dutch Mechanized Brigade to perform a fast tactical march to phase line TULIP using the predefined route DUCK. This would be expressed in BML as

advance MND-West M_BDE13(NL) **along** DUCK **start at** point-in-time-1 **in order to** enable (label of another task) label-ord-adv-11;

More details on the rules for the expression of orders as well as more examples are given in (Hieb and Schade 2007, Schade and Hieb 2006a).

With respect to formulating reports in BML, in principle the same kind of rules are used. A report consists of arbitrarily many basic report expressions (RB) as given in (3).

(3) Report → RB*

A basic report expression might express a report about a military task, about an event, about a position, or about a status. (4a) shows the general rule form of a basic task report, (4b) the general rule form of a basic event report, and (4c) the general rule form of a basic status report. The latter form is also used for basic position reports.

(4a) RB → ***task-report*** Verb Executer (Affected|Action) Where When (Why) Certainty Label (Mod)*

(4b) RB → ***event-report*** EVerb (Affected|Action) Where When Certainty Label (Mod)*

(4c) RB → ***status-report*** Hostility Regarding (Identification Status-Value) Where When Certainty Label (Mod)*

The rule form (4a) is similar to the form for basic orders (2a). However, there are three differences. First, there is no Tasker in the rule form. Second, instead of Taskee, there is the term Executer. Third, Certainty is inserted. Certainty expands to a modality operator that denotes the reporter's estimation whether the report is true, plausible or of uncertain truth. This is important if for example, a statement of a third person (civilian, prisoner of war, etc.) is reported.

The other two differences between (2a) and (4a) also result from differences between reports and orders: There is no Tasker in the task report because the Tasker is often not known to the reporter, in particular if the report is about a military task that is under execution by an enemy force. Besides, Taskee has been replaced by Executer for the same reason. Executer may be expanded by Taskee, Agent, or Theme (5a) – (5c), and in the latter cases a Label may be added for referring to the executing unit in other orders or reports.

(5a) Executer → Taskee

(5b) Executer → Agent (Label)

(5c) Executer → Theme (Label)

Executer is expanded to Taskee if the name of the executing unit is known to the reporter, and Taskee itself then is expanded to that name. Executer is expanded to Agent if not the name of the executing units but its type is known to the reporter. Agent then can be expanded by rule (6a) to a term that includes the size of the unit, its hostility, and its type, e.g., *battalion (of) hostile infantry*. Last, but not least, Executer can be expanded to Theme if only the type of main equipment used by the executing unit is known to the reporter. Theme expands by rule (6b) to a constituent that includes the number of this equipment, the hostility, and the type of the equipment – e.g., *four hostile battle tank(s)*.

(6a) Agent → Size Hostility Unit_type

(6b) Theme → Count Hostility Equipment_type

Besides reporting about a military task, a reporter can also report about an event. In the formal grammar this is done by rule (4b). In the case of events, the verbs are not taken from JC3IEDM's table "action-task-activity-code", but from the table "action-event-category-code." This is encoded by EVerb in (4b) in contrast to Verb in (2a) and (4a). Examples are "arson" (7a) and "refugee movement" (7b).

(7a) RB → ***arson*** Affected Where When Certainty Label (Mod)*

(7b) RB → ***refugee movement*** Where When Certainty Label (Mod)*

We will not go into detail about the format of status and position reports in the this paper, but c.f. (Schade and Hieb 2006b) for these details. Instead, in the following subsection, we will discuss how inter-agent communication, the communication among the simulation's agents, can be represented in BML.

3.2 To expand the BML Grammar for Inter-Agent Communication

In order to discuss inter-agent communication in simulations we have to differentiate between the communication of effects and "real" communication among agents. "Real" communication is the communication that would happen among agents if these agents would be real. For example, if we model a battalion, the real battalion would communicate with its companies as well as with the brigade it is part of. Therefore, we would have to represent this communication within the simulation.

Obviously, "real" communication among agents is military communication: the exchange of orders, requests, and reports. Thus, it can be – and should be – modeled by using those BML expressions that would also be used in the real world for real orders, real requests, and real reports. This approach not only solves the question of how to represent "real" inter-agent communication but also allows the seamless substitution of real forces (or robotic forces) by simulated forces and vice versa, an obvious advantage for staff exercises.

So, what is about the communication of effects? In order to constitute a consistent way of communication, we expanded our BML rules such that the communication of effects is also covered. The general form of a basic rule for communicating effects (WB) is shaped after the general form for event rules (4b). However, a specific term (Intensity) is introduced to allow the representation of the effect's quantity. The general form for effect rules therefore is

(8) WB → EffectVerb ***of*** Intensity (Affected) Where When Label (Mod)*

Intensity is to be expanded by a number. Currently, we have effect verbs of the following types: effects of fire, supply effects, consumption effects, demolition effects, and repair effects. Examples for rules are given in (9a) – (9e). There is one example rule for each effect type.

(9a) WB → ***fire(machine gun) of*** Intensity (Recipient) Towards-Where When Label (Mod)*
(9b) WB → ***supply(fuel) of*** Intensity Affected At-Where When Label (Mod)*
(9c) WB → ***consume(fuel) of*** Intensity At-Where When Label (Mod)*
(9d) WB → ***collapse of*** Intensity At-Where When Label (Mod)*
(9d) WB → ***rebuild of*** Intensity Facility At-Where When Label (Mod)*

The structural analogy between all kind of basic communication rules in BML facilitate the development of processing communication in the simulation.

4 PROCESSING OF BML MESSAGES

Having specified the grammar for all relevant aspects in inter-agent communication, we will briefly outline the way that BML formatted messages are processed by agents in the ITSimBw simulation environment.

Irrespective of the kind of message – order, report, or physical effect – the receiving agent has to react to it. Here, reaction means calling the appropriate behavior for the message in question. Thus, a LAMPS behavior specification has to be designed for each verb that is to be understood by the receiving agent. The agent carries those behavior descriptions as inactive behaviors that do not use up CPU time when not specifically called upon by the BML interpretation process.

This process – which in turn is also an agent behavior – is called whenever a BML formatted message is received. At first, it identifies the message content according the to leading verb and chooses the corresponding behavior description. It then uses the known grammatical structure for the phrase at hand in order to extract the relevant information that needs to be filled in as parameters of the agent behavior. The thus prepared LAMPS behavior graph is then marked as active and is executed. A symbolic depiction of this translation process is show in Figure 2:

5 CONCLUSION

In this contribution, we have identified communication requirements for multi-agent simulation systems in general, and ITSimBw in particular. Focusing on inter-agent communication, we derived the three main subject areas – orders, reports and physical effects – that have to be addressed. We showed how a suitably modified variant of the well-known battle management language BML provides the means to address all communication needs in a uniform way while providing for syntactically well formed and semantically meaningful information exchange. The communication of physical effects by BML will profit in addition by the ongoing development of geoBML (Hieb et al. 2006), a BML extension that focuses on communication about terrain and weather.

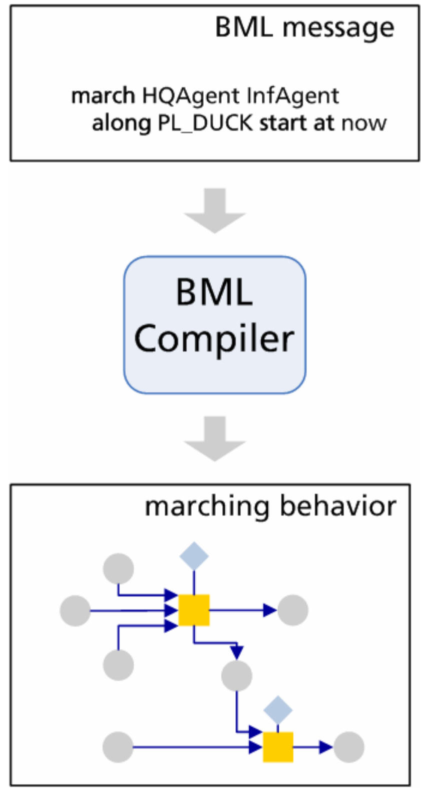

Figure 2: The BML compilation process.

The consequential use of BML as the communication protocol for ITSimBw opens up possibilities beyond the ones described in this paper. The practical use of ITSimBw as a tool for forces education and training relies upon an ergonomic way of issuing orders to simulated units. Here, a GUI that graphically supports the user while producing orders in BML format would be a well-received add-on to current ITSimBw functionalities. Furthermore, using BML for exchange of orders and reports facilitates the desired integration of real units and equipment in simulation runs, thus enabling a tighter integration of real and simulated forces in war-gaming exercises. Finally, the stronger coupling of simulation and C2 systems where the former provides decision support services for the latter is a promising area of future studies which also benefits from using BML as a joint communication tool.

ACKNOWLEDGEMENTS

The work on ITSimBw is done under contract for the department A5 of the IT-office of the German armed forces (IT-AmtBw). At this institution, the project is overseen by Major Thomas Doll. FGAN-FKIE's work on BML also is done under contract for the IT-Amt Bw, A5. It is overseen by Major Thomas Orichel. The valuable contributions from Major Doll and Major Orichel as well as those from Mrs. Kellyn Kruger (FGAN-FKIE) are gratefully acknowledged.

REFERENCES

Blais, C., M. R. Hieb, and K. Galvin. 2005. Coalition Battle Management Language (C-BML) Study Group Report, 05F-SIW-041. In *Proceedings of the 2005 Fall Simulation Interoperability Workshop* 2005, Orlando, FL.

Bryan, O. F. Jr. 1989. Modsim II - an object-oriented simulation language for sequential and parallel processors. In *Proceedings of the 1989 Winter Simulation Conference*, eds. E. A. MacNair, K. J. Musselman, and P. Heidelberger, pages 122-127.

Carey, S., M. Kleiner, M. R. Hieb and R. Brown. 2001. Standardizing Battle Management Language - A Vital Move Towards the Army Transformation, Paper 01F-SIW-067. In *Proceedings of the 2001 Fall Simulation Interoperability Workshop*, Orlando, FL.

Hieb, M.R., M. W. Powers, J. M. Pullen, and M. Kleiner. 2006. A Geospatial Battle Management Language (geoBML) for Terrain Reasoning. In *Proceedings of of the 11th International Command and Control Research and Technology Symposium,* Cambridge, UK.

Hieb, M.R. and U. Schade. 2007. Formalizing Command Intent Through Development of a Command and Control Grammar. In *Proceedings of the 12th International Command and Control Research and Technology Symposium,* Newport, NJ.

Jensen, K. 1992. *Coloured Petri Nets: Basic concepts, analysis methods and practical use. Volume 1: Basic Concepts*. Monographs in Theoretical Computer Science. Berlin: Springer.

L'Ecuyer, P. and E. Buist. 2005. Simulation in Java with SSJ. In *Proceedings of the 2005 Winter Simulation Conference*, eds. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 611-620.

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

Rice, S. V., A. Marjanski, H. M. Markowitz, and S. M. Bailey. 2005. The Simscript III Programming Language for Modular Object-Oriented Simulation. In *Proceedings of the 2005 Winter Simulation Conference*, eds. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, .621-630.

Schade, U., and M. Hieb. 2006a. Formalizing Battle Management Language: A Grammar for Specifying Orders, Paper 06S-SIW-068, In *Proceedings of the 2006 Spring Simulation Interoperability Workshop*, Huntsville, AL.

Schade, U., and M. Hieb. 2006b. Development of Formal Grammars to Support Coalition Command and Control: A Battle Management Language for Orders, Requests, and Reports. In *Proceedings of the 11th International Command and Control Research and Technology Symposium,* Cambridge, UK.

Schade, U. and M. R. Hieb. 2007. Battle Management Language: A Grammar for Specifying Reports. Paper 07S-SIW-036, In *Proceedings of the 2007 Spring Simulation Interoperability Workshop*, Norfolk, Virginia.

Weiss, G., ed. 1999. *Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge MA.

Wooldridge, M. 2005. *An Introduction to MultiAgent Systems*, 3rd edition, John Wiley & Sons, West Sussex, UK.

AUTHOR BIOGRAPHIES

PHILIPP HÜGELMEYER graduated in Artificial Intelligence and Computational Linguistics at the University of Osnabrueck, Germany, in 2002. He worked as a researcher at the same university until 2005. Now he is a researcher at the Fraunhofer Institute for Intelligent Analysis and Information-Systems in Sankt Augustin. His research interests include multi-agent systems, agent based simulation and social choice theory. His email-address is <philipp.huegelmeyer@iais.fhg.de>

ULRICH SCHADE is a Senior Scientist at the Research Institute for Communication, Information Processing and Ergonomics (FKIE) that is part of FGAN financed by the German MoD and is a Lecturer at the Institute for Communication Research and Phonetics, Bonn University. Dr. Schade received his MA in Mathematics in 1986 and his PhD in Linguistics in 1990 at Bielefeld University (Germany), developing a connectionist model for language production processes. His research interest include Natural Language Processing, Cognitive Models, and the theory of Command and Control. E-mail address: <schade@fgan.de>

THOMAS ZÖLLER is currently working as a project manager overseeing the development of ITSimBw at the department ART of the Fraunhofer Institute Intelligent Analysis and Information Systems (IAIS) in Sankt Augustin, Germany. He has studied Computer Science together with mathematics as subsidiary subject at Bonn University (Germany), from which he received Diploma and doctoral grades in Computer Science (MS / PhD equivalents) in 1999 and 2005 respectively. His research interests include Pattern Recognition, Data Analysis, and Multi-Agent Simulation. He can be contacted via <thomas.zoeller@iais.fhg.de>