COMPONENT BASED MILITARY SIMULATION: LESSONS LEARNED WITH GROUND COMBAT SIMULATION SYSTEMS

Dr. Marko Hofmann
Institute for Technology of Intelligent Systems (ITIS) and
Institute for Applied System Analysis and Operations Research (IASFOR)
University of the Federal Armed Forces Munich
Heisenbergweg 39
Germany - 85577 Neubiberg
0049 89 6004 3242
marko@informatik.unibw-muenchen.de

KEYWORDS

Components, combat simulation, pragmatics, granularity, abstraction level, reuse, repositories.

ABSTRACT

Component based modeling is said to be one of the key technologies to improve design and development of software in general, and simulations in particular. The crucial question is which components are successful in special domains. During the last two decades scientists at our institute have designed and developed ground combat simulation systems – mainly for scientific purposes, but also to support the German army. Our experiences indicate that some assumptions of component based modeling are too optimistic with respect to directly reusable software components, especially if multifunctional. The main reason for this shortcoming lies in the problem of adjusting the pragmatics of different domain specific components. On the other hand, the reuse of concepts and algorithms has always been paramount in our model development.

INTRODUCTION

In order to master the complexity of reality we decompose it into parts (Alexander 1964; Miller 1956; Simon 1962). In computer science modularity is regarded as one of the most promising approaches to improve design and development of complex systems (Balwin 2000; Czarnecki 2000; Szyperski 1998). As simulation systems get more and more complex, too, component-based approaches spread through all simulation domains (Dahman et al. 1998; Kuijpers et al. 1998; Zeigler 1993; Zeigler et al. 2000). There are both scientifically and practically interesting aspects of such approaches such as the granularity and the abstraction level of the components, the preconditions for their successful coupling and the structure of the repository for the storage and retrieval of the components. As these aspects are not independent they are discussed here in unity.

A crucial task in component based development regardless of the special domain is to ensure that the components can be used without knowing details of their implementation. Ideally, it should be possible to use a component as a black box. However, during the development of our models the *technical* and *syntactical* aspects of coupling components didn't put the major challenge. All serious problems occurred on the level of *semantics* and especially *pragmatics*. It is definitely impossible to handle such problems with black boxes.

The remainder of this paper is organized as follows: Section 2 outlines the background of ground combat simulation systems. Section 3 discusses the problem of component coupling in complex simulation systems from a linguistic point of view which lays the foundation for the discussion of useful components in section 4, appropriate granularity in section 5, and how a repository should look like in section 6. The paper concludes by reiterating its main results and suggesting some future research directions.

GROUND COMBAT SIMULATION SYSTEMS

Over more than two decades scientists at our Institute (IASFOR) have analyzed ground combat simulation systems used in the German and other armies (for example: JANUS, HORUS, SIRA, PAPST, KORA, IRIS (Stricom; IABG; CAE; Schwierz 1995) and designed and implemented own simulation systems (see below). The level of complexity of these models reaches from simplified test simulation systems and relatively simple simulations based upon cellular automata (ZEGA and ZELGAT (Hofmann 2000)) up to full scope aggregated land battle models (KOSMOS (Hofmann et al. 1992)) and high resolution ground combat models (BASIS (Hofmann et al. 1984), COSIMAC-P, COSIMAC-WS (Hofmann 2000)), which are in terms of system theory (Flood 1993) extremely complex. During that time the reuse of concepts, algorithms and code has been practiced intensively. In the following I have tried to sum up these experiences.

Ground combat simulation systems (GCSS) are a very heterogeneous class of models (Hofmann 2000; Hartman 1985), nevertheless they all share some

Proceedings 15th European Simulation Symposium
Alexander Verbraeck, Vlatka Hlupic (Eds.)

ISBN 3-936150-28-1 (book) / 3-936150-29-X (CD)

fundamental parts. Every GCSS has to model the following **aspects of combat**:

1. terrain and environmental representation,
2. movement,
3. attrition,
4. transportation (at least of ammunition),
5. communication and
6. reconnaissance.

Generally, GCSS are discrete event simulations based upon an event queue. The GCSS mentioned above aren't real time simulations, anyway internal time management is essential. Thus, the core of any GCSS will look roughly like Figure 1. If the GCSS is used for analysis in a closed simulation, it is necessary to add

7. command and control modeling,

which shouldn't be a part of the central simulator - for reasons explained in (Hofmann 2000).

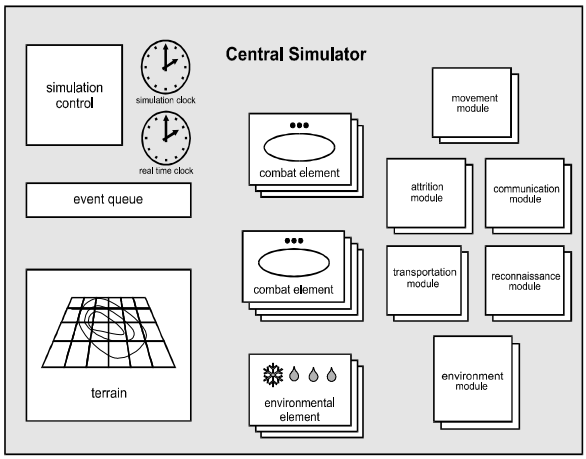

Figure 1: Essential parts of a GCSS

The major distinctions between the models, beside different **purposes** (acquisition, decision support, analyses, training)**, scopes** and **user modes** (closed simulation or interactive)**,** is their level of **resolution**: the level of detail at which the real world system and its behavior is modeled. Referring to (Davis and Bigelow 1998) and (Davis and Huber 1992) resolution in combat simulation systems has six "components":

1. temporal scale,
2. spatial scale,
3. processes,
4. entities,
5. attributes and
6. dependencies.

This classification is arguable, but useful to illustrate the degrees of freedom for the modeling. The range of two of these dimensions (spatial scale and entities) can be easily depicted. Figure 2 shows how a combat scenario would look in an aggregated model and Figure 3 shows how a combat would look in a high resolution model. The rectangles in Figure 2 represent brigades (pale gray) and divisions (dark gray) as a whole. Attrition occurs, when two enemy rectangles overlap, and is calculated with Lanchester's differential equations (Taylor 1983). In Figure 3 the symbols represent single weapon systems of an attacking (pale) and defending (dark) force like tanks, armored infantry vehicles, mortars, armed helicopters and some other elements of combat like minefields, artillery impact areas and reinforcements of the ground.
Detailed explanations of these figures can be found in (Hofmann 2000). For further reading about the principles of ground combat simulation I suggest (Hartman 1985) and (Davis and Zeigler 1997).

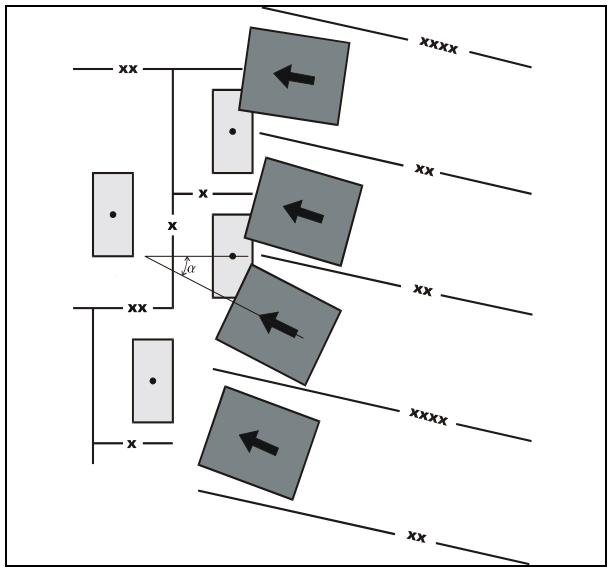

Figure 2: Depiction of a combat in an aggregated GCSS

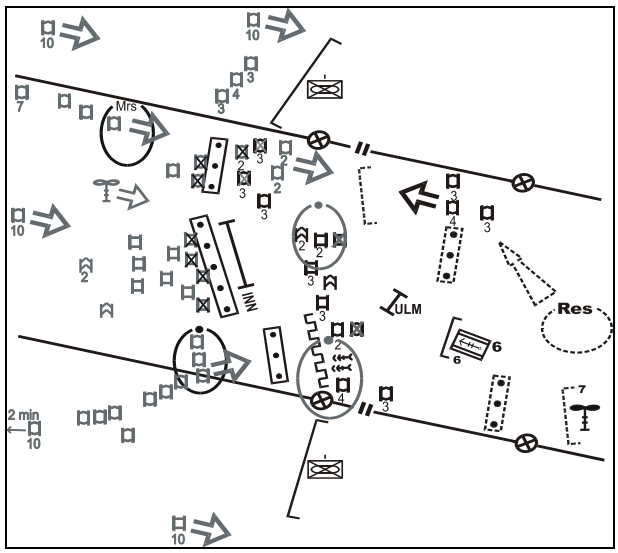

Figure 3: Depiction of a combat in a high resolution GCSS

COMPONENTS IN GCSS: SOME FUNDAMENTAL ASPECTS

Taking into consideration the different purposes, scales, user modes and resolutions of combat simulation systems, the degrees of freedom within each of these aspects and the necessity to tailor each model to fit the purpose, it is not very surprising that **reusing** software components directly in our GCSS was and still is a rare possibility, except from the use of some domain independent components such as random number generators and data bases. These components were always relatively easy to integrated because they do not contain any semantic and pragmatic context information.

In the following, the concept of pragmatics is introduced to explain the difficulties with the coupling of domain specific and "meaningful" components as general as possible.

As the complexity of the real world combats is too large to be fully captured in a model, it is necessary to simplify. Actually, the hard part of developing GCSS is not code generation but appropriate **modeling (abstraction and idealization)**. Since the measure of this appropriateness must be the purpose of the model, the value of a component cannot be judged by technical or formal syntactic correctness only, but must be evaluated on the semantic and pragmatic level.

The **basic assumption** for the following explanations is that the purpose of a component within a model is similar to the pragmatics of an utterance in linguistics.

Since most computer scientists are not familiar with the linguistic concept of pragmatics, a short description may be helpful. In the semiotic trichotomy developed by Charles Morris, Rudolph Carnap, and C. S. Peirce in the 1930s, syntax addresses the formal relations of signs to one another, semantics the relation of signs to what they denote, and pragmatics the relation of signs to their users and interpreters (Levinson 1983, Mey 1993, MITECS).

The central rationale for pragmatics is that sentence meaning (semantics) in natural languages vastly underdetermines speaker's meaning (intentions). The goal of pragmatics is to explain how the gap between sentence meaning and speaker's meaning is bridged (Sperper 2003).

In "linguistics words" (which sometimes seem to me a little bit convoluted), pragmatic information concerns facts relevant to making sense of a speaker's utterance of a sentence (or other expression). "The hearer thereby seeks to identify the speaker's intention in making the utterance. In effect the hearer seeks to explain the fact that the speaker said what he said, in the way he said it" (Bach 2003). Because the intention is communicative, the hearer's task of identifying it is driven partly by the assumption that the speaker intends him to do this. The speaker succeeds in communicating, if the hearer identifies his intention in this way, for communicative intentions are intentions whose "fulfillment consists in their recognition" (Bach 1979). In other and much simpler words, pragmatics is concerned with whatever information is relevant, over and above the linguistic properties of a sentence, to understanding its utterance (Sperper 2003).

As an **example,** consider a mountain walk of an experienced climber and his friend, who has always stayed in flat land. During the walk the climber shouts "Stone" and expects his friend to seek for shelter. Unfortunately, his friend doesn't even raise a hand. On which communication level occurred the error? We can assume that the flatlander heard what his friend said (transmission), understood the phoneme "stone" and mentally translated it into the correct word "stone" (syntactic level) and knew what a stone is (extensional meaning of the word, semantic level). Hence the fatal error must have occurred on the pragmatic level as an *failure of communicating the demand of action.*

It is obvious that the line between semantics and pragmatics cannot be absolutely definite and that some aspects of contextual information and other connotation could be placed into the semantic bucket, too. (In the example, one could argue that the semantic of the word "stone" in the context of mountain hiking has to be extended) But in general it is not recommended to extend the borders of semantics, because it quickly leads to person dependent ambiguity in semantic definitions (What if a geologist shouts stone during a mountain walk? Is he delighted or terrified?).
It should be mentioned that even Noam Chomsky, the world's most famous and influential linguist has stated that "a general linguistic theory must incorporate pragmatics as a central and crucial component" (Chomsky 1999).

However, taking the nature of pragmatics into consideration it is no surprise that it has been omitted in computers science. The general guideline in all natural and technical sciences is to reduce subjective factors down to zero. Hence scientists from this research areas seek to find or define a pragmatics-free (context and connotation free) experimental system. Unfortunately, that approach has seldom worked in human or social sciences or whenever human behavior and communication have to be regarded.

So far only the linguistic aspect of pragmatics has been discussed. The following sections change the focus to the relationship between models and pragmatics.

As an introduction to this relationship consider the definition of semiotic qualities of conceptual models (see Table 1) given by (Lindland et al. 1994).

Table 1: Definition of semiotic qualities of conceptual models (Lindland et al. 1994)

Syntactic quality	… is the degree of correspondence between a conceptual model and its representation.
Semantic quality	…is the degree of correspondence between the conceptual model and the real world.
Pragmatic quality	… is the degree of correspondence between the conceptual model and its (individual) interpretation.

The first connection between models and pragmatics is quite simple, but often underestimated. The standard situation of professional model development consists of a client who has a problem in a real world system which can't be investigated directly and a model development team who is charged with the task to solve this problem within a model. Since the clients view of the real world system generally differs from the view of the model developers, adjustments of both views are essential before starting to create a conceptual model of the real world system. We experienced this well known difficulty within our development teams, too. Therefore, from our experience, these adjustments together with proper model validation are keys to model quality (see Figure 4) (see Hofmann 2002). Generally, the adjusting of the different views of the client and the model developer, respectively, in our case, among the different model component developers is performed via natural language communication. Hence, the conceptual model can seldom be understood without taking into account the pragmatics of the communication.

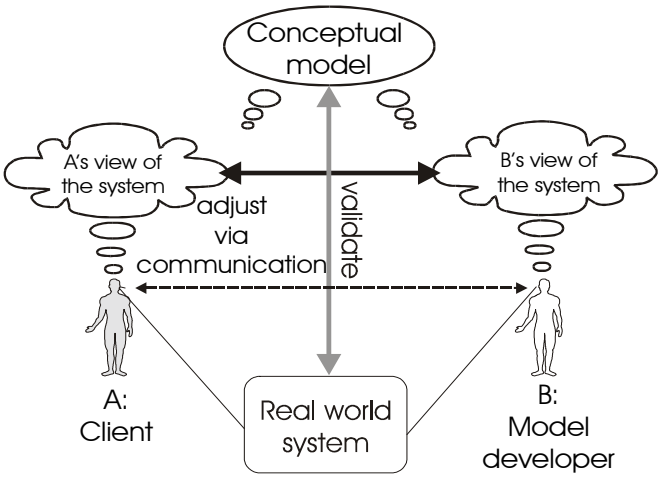

Figure 4: Adjusting personal views and validating a model

One of the central dogmas of modern computer science is the demand for unambiguous programs that **can be used without any additional context information**. Especially for component-based software architectures this requirement is said to be essential. Taking this dogma literally implies that documentation of programs mustn't be essential for model understanding and application, but only (extremely) helpful. Ideally the program/module itself (as a sequence of statements in a programming language) should contain the whole meaning/sense of the underlying (conceptual) model.

I do not doubt that from the perspective of software engineering this dogma is completely justified. There actually is a huge amount of software components that fulfill this black-box criteria. However, as far as I can see, these components are of a very fine granularity, and very often monofunctional. The simplicity of these components in terms of degrees of freedom is the reason why the black-box approach works. However, to base a general hierarchy of domain specific components - that finally would lead to complex multifunctional modules - on a black-box architecture is most probably an illusion of current software engineering. **In complex military, economic or logistic simulation systems the code vastly underdetermines the modeler's ideas and intentions**. Therefore, model documentation in natural language and additional verbal communication, despite all their disadvantages of ambiguity and connotations, are essential parts of the interaction among model developers and users. I am also convinced that the restricting of programming languages to syntax and semantics is an illusion, that has contributed to the software crises. Pragmatics as the part of semiotics that deals with the relation of signs to their interpreters must be included into the theory of programming languages, since reused models (programs) are means of communication between people, too.

USEFUL COMPONENTS

The by fare most useful things in more than twenty years of military simulation experience at the IASFOR have been **concepts to abstract and idealize reality** and algorithms extracted from this concepts - not necessarily implemented algorithms and not necessarily algorithms that could be reused without changes. Hence, what one really appreciates designing or improving a GCSS according to a model development process (Figure 5) are **well described and structured ideas of abstraction and idealization, which balance the model's need for simplification against the constraints of the system context and the imposts of the problem.** Sometimes it is even the documented system analysis preceding the design of a conceptual model which is the most useful thing of an older model. When developing complex simulation systems like GCSS, one starts with the model purpose (or the problem definition), minds the scope of the model and the user mode and chooses a global level of resolution for time and space. Afterwards, one has to find suitable

concepts for the modeling of the six (respectively seven) aspects of combat, fitting the unique combination of purpose, scope and general resolution. The relatively low degree of usefulness of *software* components is caused by the fact that they seldom fit into a new model without modifications. Additionally, if you try to find an appropriate model component for a new purpose, you will get lost with code. It will take you weeks before you grasp the idea of abstraction and idealization of the conceptual model from the executable model.

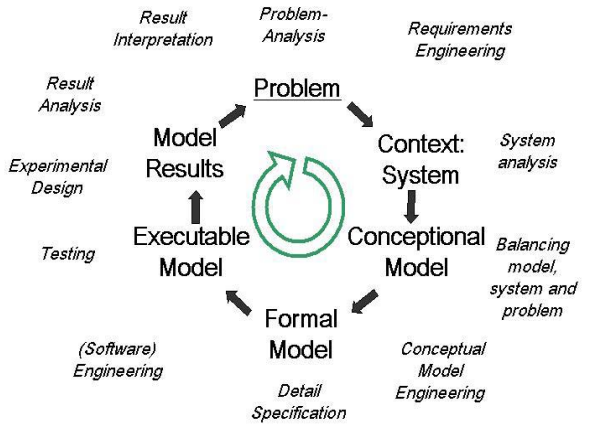

Figure 5: An idealized view of model development

GRANULARITY

As a result of their work with the French simulations system ESCADRE, (Igarza et. al) have stated that the reuse of military simulations as a whole (lowest granularity) for a new purpose is almost always impossible. These experiences and opinions conform with our own results: As a general rule we have found that model components which include more than one of the seven aspects of combat are too specific to be reused for new purposes. Hence the lowest granularity level successfully applied in our GCSS represents the real world system with components that depict one aspect of combat. From the higher granularities (with a further break down of the one-aspect-components into smaller components (hierarchy of components)), the most successfully one discriminates at the level of entities, and there attributes. Weapon system specifications, cell specifications in grid terrain or priority definitions for target selection, for example, could be reused in a high resolution combat simulation system after 15 years without any major modifications (see (Hofmann et al. 1984) and (Hofmann and Hofmann 2001).

On the other hand there are some components nearly useless in new models. Most of them belong to an intermediate class of granularity that lies between entity level and the "aspect of combat"-level. As an example take support modules for the "command and control components" such as terrain evaluation modules, assessment of the own and enemy situation modules and other estimation modules. Even slight modification of a model can devalue these modules completely.

REPOSITORIES

As a consequence of our reasoning, repositories for combat simulations systems, which would be in fact very useful, should not be restricted to software component libraries like the C++ or Java libraries in the net (Repositories 2003). It would be more promising to assemble concepts and algorithms applied in combat simulations together with documentations of system analysis, model experiments and successful model applications. Such a repository takes the whole model development process (figure 4) into account. In order to organize a part of this repository we currently work with the classification scheme showed in table 12. As an illustration some examples are inserted as a catchword. Further explanations can be found in (Hofmann 2000; Hartman 1985, Olsen (ed.) 1994, and Farell 1989).

Table 2: Classification scheme for GCSS-concepts

	Aggregated theater level modeling	**Aggregated corps/division and lower echelon modeling**	**High resolution bataillon/company and platoon modeling**	**High resolution single weapon system and single person modeling**
terrain and environmental representation	vector graphics	large grid terrain representation	narrow grid terrain, Line-of-sight algorithm	3-D virtual reality algorithms
movement	vector optimization	Branch & Bound	Branch & Bound, A*, dynamic programming	hitherto interactive
attrition	Lanchester	Lanchester-differential-equations,	markov-chain based approaches	single shot models based upon hit probabilities
transportation	classic OR-optimization	classic OR-optimization	transport capacities	explicit transport amounts models
communication	connection matrix	extended connection matrix	terrain considering algorithms	line of sight-communication
reconnaissance	simple probability approach	sophisticated probability approach	glimpse, scan, continuous - models	Line of sight algorithm
command and control	Case-based reasoning	rule-based reasoning	OR-optimization, rule-based reasoning	rule-based reasoning
In addition the classification considers for what purposes, scales and user modes the concept of modelling/algorithm has been successfully applied, what resolution in detail (temporal scale, spatial scale, processes, entities, attributes and dependencies) has been used and what kind of implementations are available.				

CONCLUSIONS

The main results of our experience with components in ground combat simulation systems are:

- it is seldom possible to reuse domain specific multifunctional components as black-boxes,
- the successful reuse of domain specific components seems similar to a successful communication on all semiotic levels,
- concepts and algorithms are the key for successful reuse of components,
- the value of components from different granularity levels is very different, too,

- model components which include more than one of the seven aspects of combat are, generally, too specific to be reused for new purposes, hence the maximal amount of useful functionality within one component seems to be limited,
- repositories for GCSS should include products from all phases of the model development process and not only executable code.

Whether these results are transferable to other application domains or not is difficult for me to answer, but I am convinced that similar experiences must have been made in some other domains, too.

REFERENCES

Alexander, C.J.W.: "*Notes on the Synthesis of Form*", Cambridge, MA, Harvard University Press, 1964

Bach, K. *The semantics-pragmatics-distinction*, http://online.sfsu.edu/~kbach/semprag.html (Feb. 2003)

Bach K. and Harnish, R. H.: *Linguistic Communication and Speech Acts.* MIT Press, Cambridge, MA, 1979.

Baldwin C. and Clark, K.: "*Design Rules: Volume 1. The Power of Modularity*" MIT Press, Cambridge, 1999.

CAE : Information about SIRA only available with permission: http://www.cae.com/ or http://offizierschule.de/hptzh/sira/ (March 2003)

Chomsky, N.: *On the nature of pragmatics and related issues. (1999)* http://cogprints.soton.ac.uk/documents/disk0/ 00/00/01/26/ cog00000126-00/chomsweb_399.html.

Czarnecki, K. and Eisenecker, U. W.: "*Generative Programming*". Addison Wesley, 2000.

Dahmann, J.S., Kuhl, F. and Weatherly, R.: „*Standards for Simulation: As Simple As possible But Not Simpler. The High Level Architecture For Simulation*", SIMULATION 71:6,7 p. 378-387,1998.

Davis, Paul K., and Reiner K. Huber. 1992. *Variable Resolution Modeling: Issues, Principles and Challenges*, N-3400-DARPA, RAND, Santa Monica, Calif.

Davis, P. K., Zeigler, B.: "Multi-Resolution Modeling and Integrated Families of Models" in: *Technology for the United States Navy and Marine Corps,* 2000-2035: Becoming a 21st Centrury Forcs; Volume 9, National Academy of Sciences, 1997

Davis, P. K., Bigelow, J. (1998). *Multi Resolution Modeling.* RAND Corporation, Santa Monica, USA.

Duchan, J. F.: *The Pragmatics Revolution* 1975-2000, University at Buffalo. http://www.acsu.buffalo.-edu/~duchan/1975-2000.html (Feb 2003)

Farrell, Robert L. 1989. *Remarks on Ground Combat Attrition Modeling*, paper presented at Army Research Office Workshop on Attrition Modeling in Large-Scale Simulations, February 2-4, Washington, D.C

Flood, R. L. and Carson, E. R.: *Dealing with complexity: an introduction to the theory and application of systems science,* Plenum Press, New York. 1993.

Green, G. M.: *Pragmatics and Natural Language Understanding*, Hillsdale, NJ: Lawrence Erlbaum. 1989; 2nd edition 1996.

Hartmann, J.K.: *Lecture Notes in High Resolution Combat Modelling and: Lecture Notes in Aggregated Combat Modelling.* Naval Postgraduate School, Monterey (CA). 1985

Hofmann, M.: *Zur Abbildung von Führungsprozessen in hochauflösenden Gefechtssimulationssystemen. Dissertation,* Universität der Bundeswehr München. NG Dissertationsverlag, München, 2000.

Hofmann, H. W. and Hofmann, M.: On the Development of Command & Control Modules for Combat Simulation Models on Batallion down to Single Item Level" in: "*New Information Processing Techniques for Military Systems*", RTO Meeting Proceeding MP-049, Neuilly-sur-Seine, Cedex; Frankreich. 2001.

Hofmann, M.: "Introducing Pragmatics into VV&A" in: Proceedings of the *2002 European Simulation Interoperability Workshop (*Euro-SIW), London, 2002,

Hofmann, H.W., Litzbarski, S., Rochel, T., Steiger, K.: „Basis - Ein Gefechtsmodell auf Btl/Rgt-Ebene, Band 1: Beschreibung des Gefechtsmodells". *IASFOR-Bericht S-8401*, Universität der Bundeswehr München, Neubiberg. 1984

Hofmann, H.W., Rochel, T., Schnurer, R., Tolk, A.: KOSMOS - Ein Gefechtssimulationsmodell auf Korps-/Armee-Ebene, Band 1: Beschreibung des Gefechtsmodells. *IASFOR-Bericht S-9208*, Universität der Bundeswehr München, Neubiberg. 1992

IABG: Information about HORUS, PAPST and KORA only available with permission: http://www.iabg.de/ (2003)

Igarza J.-L. et al.: „*Development of a HLA compliant version of the French ESCADRE simulation support environment (SSE): lessons learned and perspectives*", Centre d'Analyse de Défense (DSP/DGA), Frankreich, 1998

Kuijpers, N. van Gool, P. and Jense, H.: „*A Component Architecture for Simulator Development*", TNO Physics and Electronics Laboratory, The Hague, 1998.

Levinson, S. C.: Pragmatics, Cambridge University Press, Cambridge. 1983.

Lindland, O. I., Sindre, G., Solvberg, A.: Understanding quality in conceptual modeling. *IEEE Software*, Vol 11, No. 2, March 1994, 42-49

Mey, Jacob L.: *Pragmatics: An introduction*.: Blackwell, Oxford .1993

Miller, G. A.: "The magical number seven plus minus two: Some limits of our capacity for processing Information", *Psychological Review 63*, pp. 81-97, 1956.

MITECS: Abstracts on Pragmatics thttp://cognet.mit.edu/MITECS/Entry/horn2 (2003)

Olson, W. K. (ed): *Military operations research analyst's handbook, Volume 1: Terrain, unit movement, and Environment*, MORS, Alexandria, VA. 1994.

Pötzsch, V. *Entwicklung und Implementierung von Modulen für die Prozesse Bewegung, Abnutzung und* Aufklärung *und Entwurf eines Rahmens für Führungsmodule für das Gefechtssimulationmodell COSIMAC. Diplomarbeit*. Universität der Bundeswehr München, Neubiberg. 1997.

Repositories: (March 2003)
http://www.sei.cmu.edu/publications/documents/98.reports/98tr011/98tr011chap04.htm
http://www.npsnet.com/danf/software/library.html

Schwierz K. (1995). Laborbetrieb IRIS. DASA-Dornier, Friedrichshafen.

Simon, H. A.: "The Architecture of Complexity", in: Proceedings of the American Philosophical Society 106, pp. 467-482, 1962, reprinted in: idem, *The Science of the Artificial*, 2nd ed. Cambridge, MIT Press, 1981.

Sperper: *Pragmatics-modularity-and-mindreading*.: http://www.dan.sperber.com/pragmatics-modularity-and-mindreading.htm

STRICOM: (March 2003)
http://www.stricom.army.mil/PRODUCTS/JANUS/

Szyperski, C.: *Component Software – Beyond Object-Oriented Programming*. Addison-Wesley, 1998.

Taylor, James G. 1983a. "An Introduction to Lanchester-Type Models of Warfare," in Proceedings of the Workshop on Modeling and Simulation of Land Combat, L.G. Callahan (ed.), Department of Continuing Education, Georgia Institute of Technology, Atlanta, Ga.; Taylor, James G. 1983b. *Lanchester Models of Warfare*, two volumes, Operations Research Society of America, Arlington, Va.

Turner, Ken (ed.) *The Semantics/Pragmatics Interface from Different Points of View*. (Current Research in the Semantics/Pragmatics Interface, vol. 1). Oxford: Elsevier, 1999.

Zeigler, B. P., Praehofer, H. and Kim, T. G.: *„Theory of Modelling and Simulation"*, ACADEMIC PRESS, San Diego, USA, 2000

Zeigler, B. P. : „*Object-Oriented Simulation with Hierarchical, Modular Models*", ACADEMIC PRESS, Boston, USA, 1993.

Author Biography

MARKO HOFMANN is Project Manager at the Institute for Technology of Intelligent Systems (ITIS), Neubiberg, Germany. After his studies of computer science at the University of the Federal Armed Forces in Munich he served two years in an army battalion staff. From 1995 to 2000 he was research assistant at the Institute for Applied System Analysis and Operations Research (IASFOR) at the University of the Federal Armed Forces. Since April 2000 he is responsible for basic research in applied computer science (component based modeling, combat simulation systems, VV&A). He gives lectures at the University of the Federal Armed Forces (operations research).