AiMT

Advances in Military Technology
Vol. 11, No. 2 (**2016**), pp. 159-170
ISSN 1802-2308, eISSN 2533-4123
DOI 10.3849/aimt.01105

Language Representations Based on C-BML and Their Processing

Ľ. Dedera[1*] and M. Benčík[2]

[1]*Department of Informatics, Armed Forces Academy of Gen. M. R. Štefánik, Liptovský Mikuláš, Slovak Republic*
[2]*Department of Communication Information System Services, Base of Mobile Communications and Information Systems, Ružomberok, Slovak Republic*

The manuscript was received on 4 February 2016 and was accepted after revision for publication on 12 October 2016.

Abstract:

The paper tries to put together one computer languages view on Battle Management Languages (BMLs) with the current C-BML Phase 1 Standard: We present a case study on creating language representations by means of formal parsable context-free grammars and corresponding language processors in the area of military C2 systems utilizing the standardized C-BML data structures. We point out how techniques and tools previously used in the area of compilers (namely Flex, Bison) are exploitable in the military domain and thus might be helpful in integration of national command and control systems and deployment in multinational environment. We start with an introduction to C-BML principles and data model; next we describe the basis for our (Slovak) language representation followed by its specification in the form of a parsable context-free grammar; next follows the section devoted to the lexical, syntactic, and semantic processing of the language representation with the utilisation of Flex and Bison tools.

Keywords:

C-BML, MSDL, Domain-Specific Language, grammar, syntax, semantics, language processor

1. Introduction

Nowadays computer languages can be met with in various application domains. They are usually being used as a formalized means for representation of information and knowledge in all areas where their computer processing is expected. The main difference between them and natural languages, besides their limited expressive power and lexical vocabulary concentrated on a particular application domain, is the fact that

* *Corresponding author: Department of Informatics, Armed Forces Academy of Gen. M. R. Štefánik, Demänová 393, 031 06 Liptovský Mikuláš, Slovak Republic. Phone: +421 960 42 30 29, E-mail: Lubomir.Dedera@aos.sk*

their syntax as well as semantics is based on strict formal rules ensuring unambiguous meaning or interpretation of constructs created in the language.

Military application domain has much in common with the attitude mentioned before. For example, long before the foundation of computer science and formal languages and grammars as their integral part, in armies there existed de facto standard-standardized and formalized languages for giving orders, commands, and reports. Besides national levels, this fact has been reflected on the Coalition level [1]. Gradual deployment of military information systems called out for the need of their integration, interoperability and mutual exchange of data. The last mentioned problem has been solved, on the Coalition level, with a broad standard JC3IEDM [2] based on XML schemas. Further initiative in this field has been a series of projects and activities around the development of the Coalition Battle Management Language (C-BML) [3-6], which resulted in establishing the standard [7, 8]. According to this document, C-BML is defined as a standard language for expressing and exchanging plans, orders, requests, and reports across C2 systems, live, virtual and constructive modeling and simulation systems, and robotic systems participating in Coalition operations. Its Product Development Group has identified three phases of the development of the standard [7]:

- Phase 1, Data Model: This phase has finished with the standard [7] itself and provided basic set of terms and concepts (i.e. basic vocabulary in the form of XML schemas based on JC3IEDM as the starting point) for constructing more elaborate C-BML expressions. The standard has also established practices for identifying and describing proposed changes to the JC3IEDM for C-BML applications.
- Phase 2, Formal Structure (Grammar): This phase should establish a grammar (syntax, semantics, vocabulary) for more complex concepts and structures such as plans, orders, request, and reports to express a particular doctrine.
- Phase 3, Formal Semantics (Ontology): This phase should provide formal semantics of concepts from the previous phases to ensure automated processing of the content of the C-BML expressions and conceptual interoperability across systems [7, 9].

C-BML is closely related to another computer language within the military domain – Military Scenario Definition Language (MSDL); currently the development of both standards is covered by the common C2SIM Product Development Group [10]. Both C-BML and MSDL have also been studied, e.g. within NATO STO MSG–145 in order to promote the interoperability between C2 and simulation systems. Both MSDL and C-BML standards are formatted into XML (Extensible Markup Language) schemas. MSDL specifies force structures, environment, and other information for initialization of simulation systems; C-BML, on the other hand, can be utilized for describing the execution of military scenarios. MSDL defines a military scenario as a specific description of the situation and course of action for each element in the scenario. The representation of a scenario reflects aspects such as common mission, enemy, terrain, weather, troops and support available, time available, and civil elements of the military situation that have to be taken into consideration [11]. There are several strategies for MSDL and C-BML cooperation [12]. One of these strategies is based on the translating MSDL/C-BML elements into the required form. Translating one language into the required form will also be shown in the next chapters.

From a software engineering point of view, C-BML can be considered as an example of a Domain-Specific Language (DSL) designed specifically for the military

domain. In [13], the use of C-BML for communicating with multi-robot systems is described. In [14], a modeling case study using C-BML for the exchange of orders and reports related to patrol mission is presented. Papers [15, 16] describe how C-BML and/or MSDL are used in the Technical Cooperation Panel of the Coalition Attack Guidance Experiment (CAGE) nations or in the French-German COMELEC army training initiatives, respectively.

From the development of the C-BML grammar point of view, experiments and demonstrations with Command and Control Lexical Grammar (C2LG) with a GUI editor [4, 5] have been performed with the aim to prove that C-BML is a suitable tool for the exchange of orders and reports between C2 systems and constructive simulators [6]. Paper [17] aims at the way in which to implement lexical functional grammar based approaches into object-oriented class hierarchies with the conclusion that "Lexical Functional Grammar approach combined with object-oriented representation is a good practice in order to represent grammar in BML."

Computer science and software engineering view on the topic of languages in the context of the military application domain have been discussed in [18, 19]. In [18], a category of a Domain-Specific Language [20] that can be utilized as a programming or specification language in the military application domain (C2) has been introduced, together with the principles of syntactic as well as semantic processing of the language. At the same time, a certain parallel between traditional programming languages and DSLs in the area of C2 from both processing and utilization point of view, where the role of a DSL-trained military commander in relation to a DSL can be analogical to the role of a computer programmer in relation to a programming language. In [19], the concept of DSL has been extended in the direction of application of techniques of both abstract and concrete syntax and semantic processing as a means to achieve multilingual support and a certain form of semantic interoperability in the context of C2 systems. This approach enables to study and design families of DSLs with different concrete syntaxes (each of them tailored for a specific usage, audience, and/or nation), but with mutually related semantic processing, which should simplify the development of interoperable systems. The ideas behind abstract syntax and semantic processing are similar to the ones presented in [8], where the abstraction and association relationships between classes are used to represent a model of the grammar.

In this paper, we have followed the ideas presented in [18, 19] and tried to join them with the C-BML standard [7] by introducing an example of a grammar describing simplified structure of a language for the control of combat operations intended for Slovak (human) environment (so-called Slovak language representation, SLR), together with a language processor transforming sentences in SLR into standardized XML data structures of C-BML. The idea has been presented for the first time at the conference [21] and now we bring a more detailed view on its aspects: We start with an introduction to C-BML principles and data model; next we describe the basis for our language representation followed by its specification in the form of a parsable context-free grammar; next follows the section devoted to the lexical, syntactic, and semantic processing of the SLR with the utilisation of Flex and Bison tools [22, 23].

2. C-BML Principles and Data Model

Basic information components of C-BML can be described by so called "5Ws paradigm" (Who, What, When, Where, Why). The information obtained from 5Ws is

essential for the expression of orders, requests, and reports for any doctrine, unit, as well as nation [7]. Now let us introduce the 5Ws model description:

1. “Who” is an information component of C-BML designed to identify the object:
   - Intended to carry out an activity (TaskeeWho in orders)
   - Ordered the execution of tasks (TaskerWho in orders)
   - Affected by the task to be performed (AffectedWho in orders)
   - Who asked or was asked to perform specific actions (RequesterWho, RequestedWho specified in requests)
   - Which observed or was observed or executed some action (ReporterWho, ReportedWho specified in reports)
   - To whom a report is addressed (AddresseeWho)
2. “What” is the information component of C-BML describing the action that will be or was performed (What in orders, ReporterWhat and ObservedWhat in reports).
3. “When” is an information component of C-BML describing the time frame when the action should be or was executed:
   - When the order was issued (OrderIssuedWhen)
   - Start/end time of the action (StartWhen, EndWhen in orders)
   - Time of the event (When, WhenTime, WhenEvent, WhenDetails in reports)
   - Time relative to another when (RelativeWhen in orders and reports)
4. “Where” is an information component of C-BML providing the exact location of the object on the battlefield, the place where the action is carried out or the place where a particular action or event occurred:
   - Where an action is done (AtWhere)
   - A route to be followed in an action (RouteWhere)
   - Initial or final position (StartWhere, EndWhere)
   - Where defined as a control feature (ControlFeatureWhere)
5. “Why” is an information component of C-BML describing the reason or the purpose of the actions, or desired end state of the action:
   - Reason for executing an order (Why in orders)
   - Perceived or observed reason (ReporterWhy, ObservedWhy in reports)

The 5Ws model concerns doctrinal perspective of C-BML. The basic elements of abstraction in the doctrinal content greatly facilitate the description of orders, requests, and reporting for all organizations and national forces, which will use the C-BML. These basic components of the language are used to construct C-BML terms (orders, requests, and reports) which are designed in accordance with the content of information exchange and structure specification [7]. Thus, each "W" of the 5Ws model is used in expressing orders, requests, and reports. For example, "Who" in an order can be the unit who gave the order, whilst the other "Who" would represent the unit that will carry out the order.

As a central reference model for C-BML data model, JC3IEDM [2] has been chosen. It is sufficiently robust to cope with the amount of data that should be interchanged among systems for which C-BML is proposed (C2, robotic, M&S). The model is based on a set of concepts, their attributes, relations and business rules to check data consistency and is described using XML schemas. Fig. 1 illustrates the component “Where” describing a route to be followed in an action (RouteWhere). The route can be specified by its starting location (StartWhere), ending location (EndWhere) and a list of passing-through locations (Via). To express a location, the principle of generalization known e.g. from object-oriented methods has been utilized.

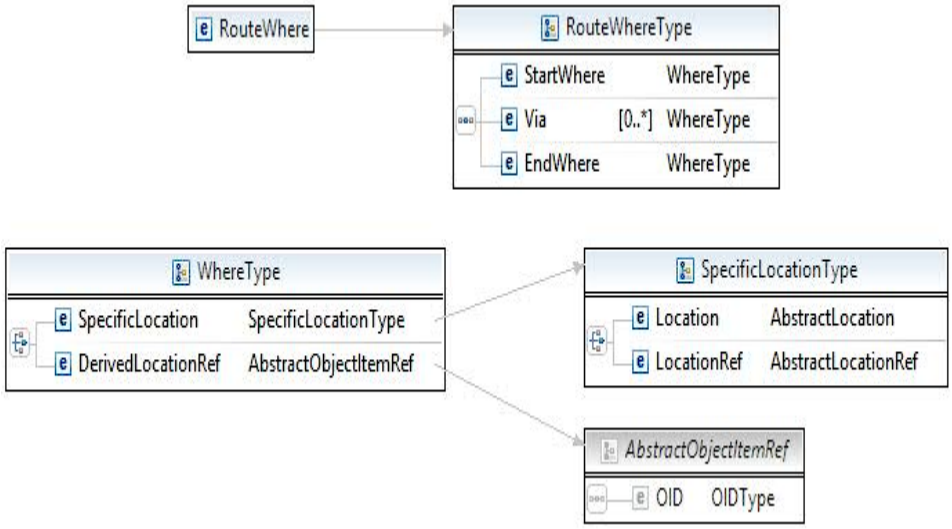

Fig. 1 C-BML data structures

3. Grammar for Slovak Language Representation

Since battle management languages are still computer languages, their syntax can be described by context-free grammars (CFGs) [20, 24]. From a theoretical computer science of view, a context-free grammar is a 4-tuple G = (N, T, P, S), where

- N is a finite set of nonterminals (or variables depicted using <…>),
- T is a finite set of terminals (lexical elements, depicted in bold),
- nonterminal S plays the role of the starting symbol of the grammar from which each derivation starts, and
- P is a finite set of productions (or rewriting rules) of the form $B \rightarrow \alpha$, where nonterminal B is the left-hand side and string (or sequence) α consisting in general of both terminals and/or nonterminals is the right-hand side of the production. The right-hand side of a production might also be the empty string, denoted further by Ɛ.

For our experimental grammar, we have selected four basic commands (Attack, March, Occupy, DelayEnemy). It is now necessary to construct a formal context-free grammar describing the syntax. From the selected basic commands, we can see what we need to describe by our grammar [8]:

- Task (Attack, March, Occupy, DelayEnemy)
- Tasker (Unit who ordered the execution of task)
- Taskee (Unit who perform the task)
- Where (Place where the task will be performed)
- Route (Route where the unit will be moving)
- Start (Start time of the task)
- End (End time of the task)
- Why (Reason of the task)

When creating the grammar, we have to take into consideration linguistic aspects of the Slovak language (punctuation, word order in phrases, addition of some key-words, data formats, etc.) in order to ensure better understanding of the language phrases for Slovak-speaking users and mainly (since our grammar is experimental) to show how these aspects can be reflected in a concrete formal context-free grammar.

Our experimental grammar for SLR consists of 37 rewriting rules. The names of relevant nonterminals reflect the 5Ws principle. Rules 1-2 describe the sequence of commands. Rule 3 describes a structure of a single command (prikazuje). Rules 4-7 describe four particular tasks – Attack (útoč), March (pochoduj), Occupy (obsaď) and DelayEnemy (zdrž nepriateľa). Rules 8-13 describe a formation (sila) of the enemy unit from squad (družstvo) through platoon (čata), company (rota), battalion (prápor) to brigade (brigáda). Rules 14-16 describe Taskers and Taskees; rules 17-27 time phrases, particularly at (o), not later than (najneskôr o), immediately, not later than (ihneď, nie neskôr ako), after (po), immediately after (ihneď po), before (pred), not before (nie pred); rules 28-29 not compulsory Why data and rules 33-37 Where data using phrases from (z), to (do), and via (cez). The starting symbol is <commands>.

1. <commands> → <command> <commands>
2. <commands> → Ɛ
3. <command> → <TaskerWho> **prikazuje** <TaskeeWho> <TaskWhat> <StartWhen> <EndWhen> <Why>;
4. <TaskWhat > → **útoč** <AtWhere> <Strength>
5. <TaskWhat> → **pochoduj** <RouteWhere>
6. <TaskWhat> → **obsaď** <AtWhere>
7. <TaskWhat> → **zdrž nepriateľa** <Strenght> <AtWhere>
8. <Strength> → **sila** <formation>
9. <formation> → **družstvo**
10. <formation> → **čata**
11. <formation> → **rota**
12. <formation> → **prápor**
13. <formation> → **brigáda**
14. <TaskerWho> → <unit>
15. <TaskeeWho> → <unit>
16. <unit> → **string**
17. <StartWhen> → <time data>
18. <time data> → **o** <time>
19. <time data> → **najneskôr o** <time>
20. <time data> → **ihneď, nie neskôr ako** <time>
21. <time data> → **po** <time>
22. <time data> → **ihneď po** <time>
23. <time data> → **pred** <time>
24. <time data> → **nie pred** <time>
25. <time> → **time** <timezone>
26. <timezone> → **timezone**
27. < EndWhen > → <time data>
28. <Why> → **string**
29. <Why> → Ɛ
30. <AtWhere> → **coordinates** <Precision> <WhereCategory>
31. <Precision> → **precision**
32. <Precision> → Ɛ

33. <WhereCategory> → **category**
34. <WhereCategory> → Ɛ
35. <RouteWhere> → **z** <AtWhere> **do** <AtWhere> <via>
36. <via> → **cez** <AtWhere> <via>
37. <via> → Ɛ

Regarding lexical elements, some of them represent a unique sequence of characters (**útoč**, **ihneď**, **o**, **z**, **cez**, **družstvo** etc.); others (namely **string**, **time, timezone, coordinates, precision, category**) cover sets of data of a particular format.

4. Processing of the Language Representation

Development of the language processor consists of two parts, namely the creation of the lexical analyzer and the parser, along with semantic routines. For both there are specialized software tools which allow the developer to concentrate on substantial, creative activities (description of lexical elements in the former case and description of a grammar accompanied with semantic routines in the latter case). In our case, for the creation of the lexical analyzer the Flex tool has been used and the Bison tool has been used for the creation of the parser. The process is depicted in Fig. 2. Both tools were originally developed to support compiler construction and run under the Unix operating system; in the case of Microsoft Windows we can use the Cygwin environment that provides functionality similar to the Linux operating system.

Flex [22] is a tool for generating lexical analyzers or, more generally, text processors based on regular expressions. Flex processes an input file (with the suffix .l or .lex) and generate a lexical analyzer in the form of source code in the C programming language (file lex.yy.c), which is accessible via the function `yylex()`. The description of the lexical elements is created in the form of pairs consisting of regular expressions and related codes in the C language. Regular expressions describe individual lexical elements and after recognizing a particular one of them, the corresponding program code executes. For example, the lexical element **time** can be described by a regular expression

```
(([0-1][0-9] | [2][0-3])[:][0-5][0-9]) return(TIME);
```

where `TIME` is a symbolic constant returned in this case by the function `yylex()`.

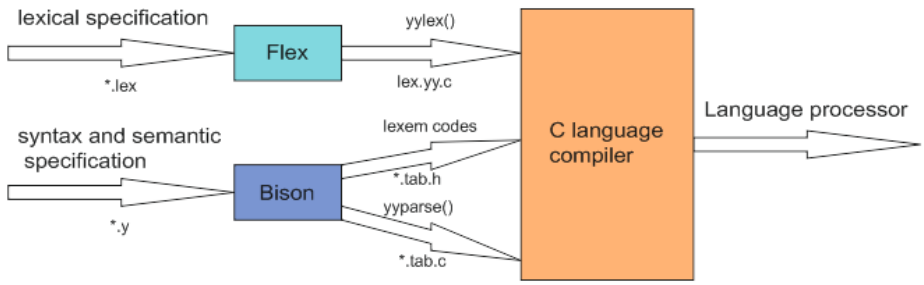

Fig .2 Creation of language processor

Bison [23] is a LALR(1) parser generator of language processors. Bison reads the specification of the language in the form of a context-free grammar combined with semantic actions expressed as blocks of the C programming code (input file with the suffix .y) and generates a parser (again in the form of the C programming language code) that is able to read a sequence of tokens (lexical elements recognized by the lexical analyzer) and decide whether a sequence (sentence) corresponds to the syntax specified by the grammar. If a particular sequence of symbols (usually right-hand side of a particular rule) is recognized, then the corresponding semantic action (programming code) is executed. The tool itself offers support for the information exchange among semantic routines by means of semantic records and a semantic stack. For example, let us consider rule 35 describing the structure of a route from a starting point (Z) to an ending point (DO) via an optional sequence of passing-through points. Its syntax as well as semantic processing can be described in the following way:

```
RouteWhere:
      Z AtWhere DO AtWhere Via
      {
        $$=(struct Route *)malloc(sizeof(struct Route));
        $$->from = $2->AtWhere;
        $$->to = $4->AtWhere;
        $$->via = $5->Via;
      }
      ;
```

When the parser recognises the right-hand side of this production, the corresponding semantic routine creates a new semantic record (of the type `struct Route`) representing the whole route and associates it with the left-hand side of the production (`RouteWhere, $$`). The starting point of the route is collected from the semantic record associated with the first nonterminal AtWhere (accessible as $2), ending point from the semantic record associated with the second nonterminal AtWhere ($4) and the list of passing-through points from the semantic record associated with the nonterminal Via ($5).

As we could see, the main role of semantic routines was to collect information from lexical elements and/or nested structures and output this information in the form of XML data structures as it is shown in the next example. The generated parser runs only relevant semantic actions and ensures communication between them using a semantic stack. The Bison tool is compatible with the Flex tool and they are often used together for the development of language processors.

In the next step a language processor for the grammar for SLR has been developed. Its role is to transform syntactically valid expressions in SLR into standardized C-BML data structures. For example, the command

"2.Division" prikazuje "1.Platoon" zdrž nepriateľa sila družstvo 13.548789 22.632541 10MTR TARGET najneskôr o 11.04.2016 08:15:00 CET nie pred 11.04.2014 10:30:00 CET;

is transformed by the language processor into:

```
<?xml version="1.0" encoding="UTF-8"?>
<CBML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.sisostds.org/schemas/c-bml/1.0"
xsi:schemaLocation="http://www.sisostds.org/schemas/c-bml/1.0 ../../example-
expressions-schema/example-cbml-expressions.xsd">
<Task>
<What>
<ActionTask xsi:type="OtherActionTask">
<ActivityCode>DELAY ENEMY</ActivityCode>
</ActionTask>
</What>
<FormationLevel>SQUAD</FormationLevel>
<TaskerWho>
<OrganisationRef xsi:type="UnitRef">
<OID> "2.Division"</OID>
</OrganisationRef>
</TaskerWho>
<TaskeeWho>
<OrganisationRef xsi:type="UnitRef">
<OID> "1.Platoon"</OID>
</OrganisationRef>
</TaskeeWho>
<AtWhere>
<SpecificLocation>
<Location xsi:type="GeographicPoint">
<LatitudeCoordinate>13.548789</LatitudeCoordinate>
<LongtitudeCoordinate>22.632541 </LongtitudeCoordinate>
<PrecisionCode>10MTR</PrecisionCode>
<WhereCategory>TARGET</WhereCategory>
</Location>
</SpecificLocation>
</AtWhere>
<When>
<StartWhen>
<AbsoluteTime>
<SpecifiedTime>
<Datetime>20160411081500</Datetime>
<StartQualifierCode>NLT</StartQualifierCode>
<Timezone>CET</Timezone>
</SpecifiedTime>
</AbsoluteTime>
</StartWhen>
```

```
    <EndWhen>
        <AbsoluteTime>
          <SpecifiedTime>
            <Datetime>20160411103000</Datetime>
            <EndQualifierCode>NOB</EndQualifierCode>
            <Timezone>CET</Timezone>
          </SpecifiedTime>
        </AbsoluteTime>
    </EndWhen>
</When>
</Task>
```

5. Conclusion

In this paper, we have tried to extend our computer languages view on Battle Management Languages and combine it with the current C-BML standard. We have introduced an example of a formal context-free grammar describing simplified structure of a language for the control of combat operations intended for Slovak (human) environment, together with a description of the language processor transforming sentences in SLR into standardized XML data structures of C-BML that has been developed using the Flex/Bison tools. The grammar described in the paper has been designed for research and demonstration purposes only: The aim of the paper was to point out at a certain approach to the topic, not to find a detailed complex solution which the authors did not have enough resources for. When creating a more complex grammar, it would be necessary to pay attention to its parsability by one of the known parsing techniques (e.g. LL(1), LR(1), LALR(1)), but this is, in the case of artificial computer languages, a solvable problem.

The specific of our approach to Battle Management Languages is the fact that our primary starting concept is the formal language as one of the key concepts in computer science. We consider this approach to be more general, and together with respecting principles of language engineering and separation of abstract and concrete syntax and semantics [19, 20, 24] more adaptable to various conditions, as the approaches where the language is considered just as a means to solve a particular problem.

By this example we also wanted to point out how it is possible to construct concrete language representations tailored for particular environments. Different environments can represent either different nations and/or the fact that the language should be used primarily for communication with humans (either in graphical or in textual form) or machines. For this reason, we find it interesting to study and design languages with different concrete syntaxes, but with mutually related semantic processing. The principles of language design and processing presented in the paper could lead to the design of the whole family of "BMLs", each tailored for particular audience or purpose, together with their language processors. Considering the principles on which C-BML has been designed, it seems to be possible. This goal can be accomplished by utilizing an abstract syntax of the language and defining the great majority of semantic processing on it [21].

In our work we have also come to two main conclusions [21]:

- We have shown that computer languages and techniques of their processing can be utilized in the area of semantic interoperability (e.g., data and command conversions) among different military systems.
- We have shown that the Flex and Bison tools that have been originally developed for the support of compiler construction can also be utilized in a completely different application domain.

Acknowledgement

This work has been supported by the Ministry of Defence of the Slovak Republic (research project VV1-2015 "Cyber Threats and Defence of Military Information Systems").

References

[1] NATO STANAG 2014, *Formats for orders and designation of timings, locations and boundaries.* North Atlantic Treaty Organisation, 2000.

[2] NATO STANAG 5525, *Joint C3 information exchange data model – JC3IEDM.* North Atlantic Treaty Organisation, 2007.

[3] BLAIS, C., HIEB, M. R. and GALVIN, K. Coalition Battle Management Language (C-BML) [Study Group Report]. In *Proceedings of the Fall Simulation Interoperability Workshop*. Orlando: SISO, 2005. 13 p.

[4] SCHADE, U. and HIEB, M. R. Formalizing Battle Management Language: A Grammar for Specifying Orders. In *Proceedings of the Spring Simulation Interoperability Workshop*. Huntsville: SISO, 2006. 13 p.

[5] REIN, K., SCHADE, U. and HIEB, M. R. Battle Management Language (BML) as an Enabler. In *Proceedings of the 2009 IEEE International Conference on Communications ICC 2009*. Dresden: IEEE, 2009. 5 p.

[6] HEFFNER, K., BROOK, A., REUS, N., KHIMECHE, L., MEVASSVIK, M. O., PULLEN, M., SCHADE, U., SIMONSEN, J. and GOMEZ-VEIGA, R. NATO MSG-048 C-BML Final Report Summary. In *Proceedings of the 2010 Fall Simulation Interoperability Workshop*. Orlando: SISO, 2010. 11 p.

[7] SISO-STD-011-2014, *Standard for Coalition Battle Management Language (C-BML) Phase 1*. Simulation Interoperability Standards Organization, 2014.

[8] SISO-GUIDE-00X-201X, *Guide for Coalition Battle Management Language (C-BML) Phase 1 Version 1.0*. Simulation Interoperability Standards Organization, 2014.

[9] GALINEC, D., STEINGARTNER, W. and MACANGA, D. Command and Control Information Systems Semantic Interoperability using a Canonical Messaging Approach. *Central European Journal of Computer Science*, 2012, vol. 2, no. 3, p. 316-330.

[10] SISO [on line]. *C2SIM PDG/PSG - Command and Control Systems - Simulation Systems Interoperation*. [cited 2016-06-16]. Available from: <https://www.sisostds.org/StandardsActivities/DevelopmentGroups/C2SIMPDGPSG-CommandandControlSystems.aspx>

[11] BLAIS, C. *Strategies for Application of the Coalition Battle Management Language (C-BML) with the Military Scenario Definition Language (*MSDL). Monterey: Calhoun, the Naval Postgraduate School Institutional Archive, 2012. 11 p.

[12] HEFFNER, K.., BLAIS, C. and GUPTON, K.. *Strategies for Aligment and Convergence of the Coalition Battle Management Language (C-BML) and the Military Scenario Definition Language (MSDL)* [on line]. [cited 2016-06-16]. Available from: <http://www.pegasim.com/CITT/4_Tasks/CBML_MSDL_Alignment/AlginmentStrategiesForCBMLandMSDL-part2_final.pptx>.

[13] REMMERSMANN, T., SCHADE, U., REIN, K. and TIDERKO, A. BML for Communicating with Multi-Robot Systems. In *Proceedings of the 2015 Fall Simulation Interoperability Workshop*. Orlando: SISO, 2015. 7 p.

[14] ÜNAL, Ö. and TOPÇU, O. Modelling Unmanned Surface Vehicle Patrol Mission with Coalition Battle Management Language (C-BML). *The Journal of Defense Modeling and Simulation Applications, Methodology, Technology*, 2014, vol. 11, no. 3, p. 277-308.

[15] BROOK, A. and MIFSUD, M. Using C-BML in a persistent Coalition C2-Simulation Experimentation Environment. In *Proceedings of the 20th International Command and Control Research and Technology Symposium*, Annapolis: International Command and Control Institute, 2015. 7 p.

[16] KHAYARI, R., LOTZ, H., KROSTA, U., KHIMECHE, L., CUNEO, X. and REMMERSMANN, T. Practical Use of BML and MSDL Standards for Supporting French German Training. In *Proceedings of the 2015 Fall Simulation Interoperability Workshop*. Orlando: SISO, 2015. 9 p.

[17] GUSTAVSSON, P. M., WEMMERGARD, J. and JONSSON, F. Object-Orientated Implementation of Grammar Based Battle Management Languages. In *Proceedings of the 2012 Spring Simulation Interoperability Workshop*. Orlando: SISO, 2012. 16 p.

[18] DEDERA, Ľ. Domain-Specific Languages for Command and Control Systems. *Science&Military*, 2010, vol. 5, no. 1, p. 40-46.

[19] DEDERA, Ľ. Semantic Interoperability by Means of Computer Languages. In *Military Communications and Information Technology. A Trusted Cooperation Enabler*. Warsaw: Military University of Technology, 2012, vol. 1, p. 209-220.

[20] FOWLER, M. *Domain-specific languages*. Boston: Addison-Wesley Professional, 2010. 597 p.

[21] DEDERA, Ľ. and BENČÍK, M. An example of a language representation based on C-BML. In *Proceedings of Communication and Information Technologies: 8th International Scientific Conference*, Liptovský Mikuláš: Armed Forces Academy, 2015, p. 1-7.

[22] THE FLEX PROJECT [on line] *Flex: The Fast Lexical Analyzer*. [cited 2014-04-09]. Available from: <http://flex.sourceforge.net/>.

[23] GNU.ORG [on line]. *Bison - GNU parser generator.* [cited 2014-04-08]. Available from: <http://www.gnu.org/software/bison/>.

[24] GRUNE, D., VAN REEUWIJK, K., BAL, H. E., JACOBS, C. J. H. and LANGENDOEN, K. *Modern Compiler Design*. New York: Springer, 2012. 822 p.