Semantic Enhancements when Designing a BOM-based Conceptual Model Repository

Vahid Mojtahed
FOI, Swedish Defence Research Agency
Division of Command and Control Systems
Department of Decision Support Systems
SE-164 90 Stockholm, Sweden
vahid.mojtahed@foi.se

Eric-Oluf Svee, Jelena Zdravkovic
Department of Computer and Systems Sciences
Stockholm University
FORUM 100, SE- 164 40 Kista, Sweden
eric-sve@dsv.su.se, jelenaz@dsv.su.se

Keywords:
Ontology, BOM, Repository, OWL

ABSTRACT: The Defence Conceptual Modeling Framework (DCMF) is the Swedish Defence Research Agency's (FOI) proposal for conceptual modeling in the military domain. DCMF enables the conceptualization, composition, visualization, and reuse of knowledge for modeling and simulation. To achieve these aims, DCMF requires that its final products—conceptual models expressed as Base Object Models (BOMs)—-are embedded with semantics. In this study, this is accomplished by formalizing them through the use of an ontology. These semantically enriched models are better able to achieve key requirements of DCMF, mainly conceptualization and reuse of knowledge. Such requirements are crucial when conceptual models are stored for later use in a repository.

1 Introduction

The Defence Conceptual Modeling Framework (DCMF) is a project at the Swedish Defence Research Agency (FOI) meant to develop a framework for creating and representing conceptual models [2]. Those conceptual models are formalized descriptions of real world processes, entities, associated relationships, and interactions that constitute military missions, operations, or tasks. They are the final artifacts of the DCMF process, intended as requirements models for simulation development.

To enable a satisfactory analysis of domain knowledge and its shared understanding among people and software systems, DCMF requires that conceptual models are formalized with semantics. In this paper we discuss one such formalization developed by utilizing the Base Object Model (BOM). BOM was created by Simulation Interoperability Standards Organization (SISO) to enable composability and reuse of concepts intended for designing High Level Architecture (HLA) simulation models, called *federations* [5]. The IEEE-standard Federation Development and Execution Process (FEDEP) defines the process for developing and supporting federations conforming to HLA [5]. FEDEP proposes six basic steps: Define Federation Objectives, Develop Federation Conceptual Model, Design Federation, Develop Federation, Integrate and Test Federation, and Execute Federation and Prepare Results.

When using BOM to conceptualize the artifacts in DCMF, the framework can be seen as a well-defined approach for designing Step 2 in FEDEP, i.e., for guiding development of conceptual models for federations.

However, when considering the BOM as a storage model in the DCMF process, a need arose to find a way to enrich BOM with more semantics than it is capable of carrying via its standard representation in XML. This became especially important when considering the reuse of the DCMF artifacts stored in a repository. While several approaches have been offered, among others our earlier

proposal of BOM++ [6] they have, as the end result, extended the BOM specification with new elements, thus altering the integrity and structure of the original BOM.

This paper presents the outcomes of a theoretical consideration and implementation of a repository to house semantically enhanced, BOM-based concepts. In our approach, we have specified BOM using Web Ontology Language (OWL) [3] while keeping the BOM specification unaltered for three major reasons: a) to maintain the integrity of the original BOM, b) to preserve the compliance with the methods and concepts that are aligned with the original BOM specification, and finally, c) to explore in detail the capabilities of BOM expressed using OWL, before considering any extensions to the specification. The proposed solution, based on the results of the DCMF process, provides a method for not only eliciting and creating ontology-based BOMs, but has the concomitant benefit of increasing the conceptual model's capabilities for use and reuse from the semantic perspective.

The paper is organized as follows: Section 2 gives overviews of the basic concepts underlying our work. In Section 3, we describe the design of the BOM ontology for the repository purpose. In Section 4, we explain the usage scenarios of the repository. The final section concludes the study and describes further research.

2 Basic Concepts

In this section we give an overview of the works related to our proposal of an ontology-based conceptual repository.

2.1 DCMF

The DCMF process is an iterative process spanning four major phases governed by different roles of responsibilities (see Figure 1).

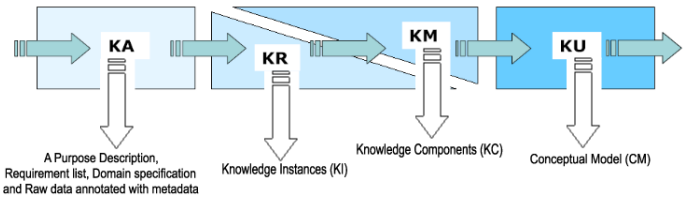

Figure 1: Four Phases of DCMF [10]

Information is first gathered within the Knowledge Acquisition phase. The Producer role processes unstructured knowledge and transforms it into represented knowledge. To accomplish this, a parsing method must be used. DCMF is agnostic where methods are concerned, though Subject-Predicate-Object as used in the Resource Description Framework (RDF) [3] has been discussed in the literature. Under current review is the DEMO-based modeling method proposed by Dietz [10].

During the Knowledge Representation phase, smaller sections of this data are structured as Knowledge Instances (KI) and validated for storage in the repository by the Controller role. KIs are useful for some purposes, but they are not reusable since they are specific to the scenario data. To get reusable knowledge, KIs are abstracted to the type level, modeled as Knowledge Components (KC) and then validated in the third phase, called Knowledge Modeling. These components are, upon Consumer requests, composed to form Conceptual Models (CM) in the fourth and final phase, Knowledge Use. All the described artifacts are stored in a repository for use and reuse.

2.2 BOM Overview

The BOM was created by the SISO to encourage and support reuse, interoperability, composability, and to help enable rapid development of HLA simulations. Conceived in 1997, BOM was standardized by SISO in 2006 [6].

2.2.1 What is BOM?

At a high level, BOMs are reusable packages of information representing independent patterns of simulation interplay and are intended to be used as building blocks in the development and extension of simulations. These components can also be composed in larger models e.g., BOM Assemblies. Additionally, interplay within a simulation or federation can be captured and characterized in the form of reusable patterns. These are sequences of events between simulation elements. Implementation of these patterns using HLA object model constructs is also captured in the BOM. [1][5]

2.2.2 BOM Structure

Structured in five major parts, a BOM is an XML document that encapsulates the information needed to describe a simulation component.

The first part is Model Identification, where metadata about the component is stored. This part includes Point of Contact (POC) information, as well as general information about the component itself (e.g., Type,

Security Classification, Purpose, Application Domain, Use Limitations, and Keywords). These facts describe what it simulates, how it can and has been used, as well as descriptions aimed towards helping developers find and reuse it.

The second part is the Conceptual Model. The Conceptual Model contains information that describes the patterns of interplay of the component. This part includes what types of actions and events that take place in the component, and is described by a pattern description, a state-machine, and a listing of conceptual entities and events, which, when taken together, describe the flow and dependencies of events and their exceptions.

The third part is Model Mapping, and is where conceptual entities and events are mapped to their HLA Object Model representations. This part bridges the Conceptual Model with the HLA Object Model that is described in the fourth part of the BOM.

The fourth part, the Object Model Definition, contains the information that is found in a normal Federation Object Model (FOM) or Simulation Object Model (SOM)-–objects, attributes, interactions and parameters—and should conform to the HLA Object Model Template (OMT).

The fifth section is called Supporting Tables and consists of two parts, namely Notes and Definitions. These contain semantic information about events and entities as well as actions that is specified in the Conceptual Model, and are used to provide a human-readable understanding of the patterns described in the BOM.

Because DCMF's focus is on developing activity-based conceptual models, this paper will focus on the second part of the BOM as detailed above. While the other sections bear some importance, they are more data-centric and not germane to the purposes of this article.

2.2.3 Conceptual Models in BOM

BOMs Conceptual Models are further divided into four different sub-models:

- the Pattern of Interplay;
- the State Machine;
- the Entity types; and
- Event types.

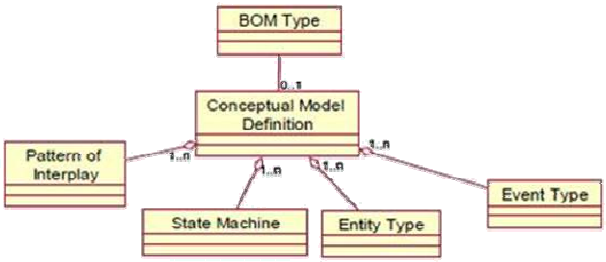

Figure 2: Structure of BOM Conceptual Models [5]

Pattern of Interplay

The Pattern of Interplay (POI) describes the recurring behavior used to accomplish a common objective, capability, or purpose, as carried out by a real world entity, phenomenon, process or system [5].

It contains actions, entity types, and event types that take part during one interaction between two components. It also contains variations and exceptions that could happen as alternatives to the normal flow of events. These actions form a flow that is sorted by the Sequence Number associated to each of them. Along with the Sequence, a Name, a Sender, and a Receiver are also specified.

State Machine

The State Machine model is a mechanism for modeling how entity types move from one state to another via actions. State Machines are made of states and conditions that must occur to hop to the next state, and are related to a specific conceptual entity.

Entity Types

Entity types represent real-world objects; they are used in POIs in order to define Senders and Receivers. Entity types also describe the conceptual entities that are used within State Machines, and have Attributes associated with them.

An important part of Entity Types is Model Mapping, which describes the mapping between Entity and Event elements from BOMs to their counterparts in HLA's Object Model. [7]

Event Types

Event types are used by conceptual entities to make transitions from one state to another and are employed within the POI model. There are two kinds of Events: Triggers and Messages. Triggers are sent by some entities

without a specific receiver, while Messages do have a specific recipient.

The Sender and Receiver identified for a Pattern Action, Variation, or Exception are used to help understand the relationship of the POI among the conceptual entities that are to be exhibited within a simulation or federation. [7]

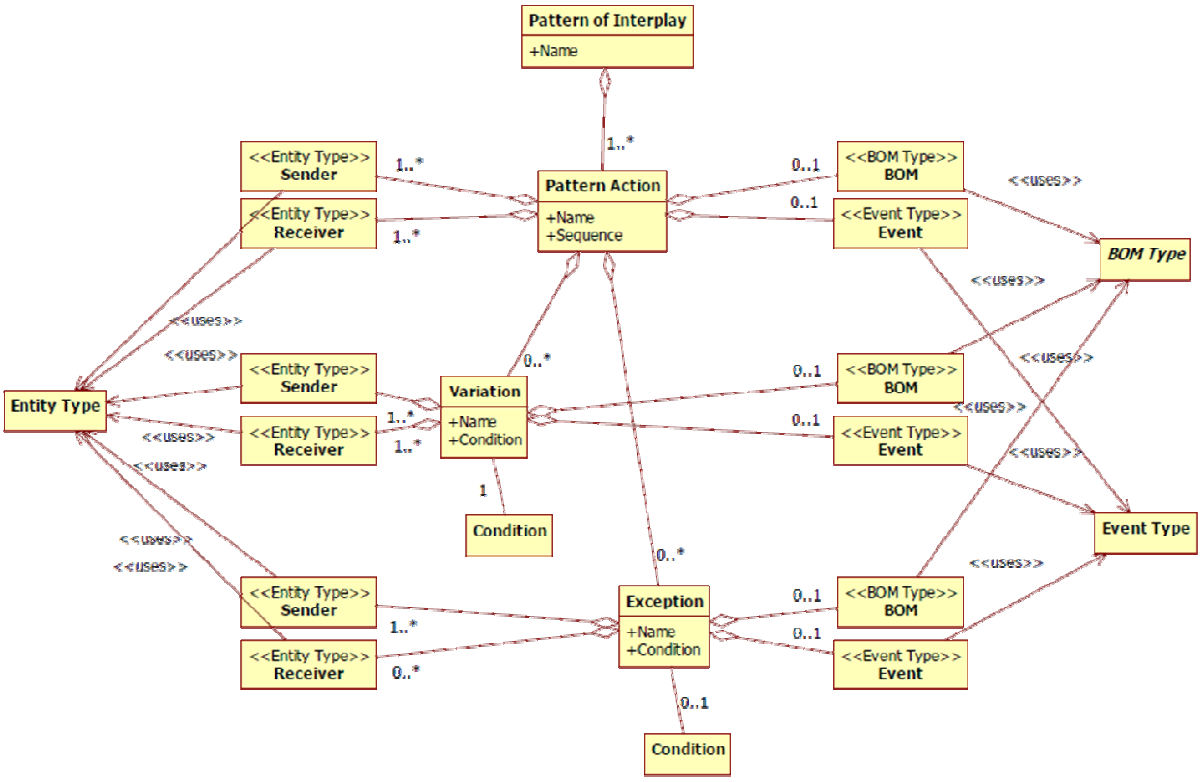

Figure 3: BOM Pattern Action Relationships [5]

2.2.4 Ontology and OWL

Ontology refers to an engineering artifact, constituted as a specific vocabulary used to describe a certain reality, by logical axioms designed to account the intended meaning of the vocabulary.

Their importance has long been recognized in a number of research fields and application areas, including knowledge engineering and representation, information retrieval and extraction, and so forth. In these, ontology has been used mainly to facilitate a semantically controlled vocabulary of concepts and to enable automated reasoning and inferences concerning those concepts, their relationships and instances.

The Web Ontology Language (OWL) was proposed to provide a language for ontology design. It is used to describe the Classes, Instances (Individuals) and Relationships (Properties) between them. OWL facilitates greater machine interpretability of content than that supported by XML, or RDF, by providing additional vocabulary along with a formal semantics. Ontology differs from an XML schema in that it is a knowledge representation, not a message format.

The language is built on formalisms that conform to Description Logic (DL) forms and therefore allows for reasoning and inference. Reasoning is the act of making implicit knowledge explicit. For example, an OWL knowledge base containing descriptions of students and their parents could infer that two students exhibited the `brother` relationship if there were both male and shared

one or more parent. No explicit markup indicating the `brotherhood` relationship need ever have been declared.

To perform such tasks, a reasoning engine is required. These are computational machinery that use facts found in the knowledge base and rules known *a priori* to determine the following:

1. *Class membership*: If `XYZ` is an instance of class `Restaurant`, and `Restaurant` is a subclass of `Business`, then we can infer that `XYZ` is an instance of `Business`.

2. *Equivalence of classes*: If class `A` is equivalent to class `B`, and class `B` is equivalent to class `C`, then `A` is equivalent to `C` too. In addition to the transitive-centric definition of equivalence, in OWL, two classes can be considered as equivalent if they instantiate exactly the same individuals.

3. *Classification:* If we have declared that certain property-value pairs for `Restaurant` class should satisfy the condition that `Restaurant` should be a 10-letter word for membership of `Restaurant` class, then if an individual `BurgerKing` satisfies such a condition, we can conclude `BurgerKing` is an instance of `Restaurant` class.

4. *Consistency* determines if the model is consistent. For example, an OWL model contains these facts:
   ```
   (a) cows are vegetarian
   (b) sheep are animals
   (c) a 'mad cow' is a cow that has
       eaten sheep brain
   ```
   From these facts a computational reasoning engine can infer that `mad cows` are inconsistent since any `cow` eating `sheep` violates `a`.

5. *Subsumption* infers knowledge structure, mostly hierarchy, or the notion of one artifact being more general than another. For example, a model incorporating the notions
   ```
   a) drivers drive vehicles
   b) bus drivers drive buses and
   c) a bus is a vehicle
   ```
   Subsumption reasoning allows the inference that `bus drivers are drivers` (since `vehicle` is more general than `bus`).

Tools for Working with OWL

Protégé is an open source application created by Stanford University [4]. It is a Java-based standalone application with an extensible architecture. The tool offers capabilities for graphically-oriented ontology development using the Protégé Editor, as well as other services, such as merging, aligning, various types of visualization, and exporting/importing, among others [4]. In its latest version, Protégé 4, the tool supports designing OWL 2 ontologies, as well as a number of Plug-Ins, such as Reasoner Manager, the SWRL engine [15], among others.

2.2.5 Previous Research Attempts

Although BOM has many adherents, many people have found its semantic capabilities deficient, specifically that the current BOM standard lacks the required semantic information to avoid ambiguity [14]. There have been several attempts to change this, primarily through altering and extending the BOM.

The approach shown in this work concerns expressing BOM as an ontology through the use of OWL. The other approaches taken have focused on end goals, such as composability, and chose methods accordingly, specifically syntactic and semantic, which allowed for the most expeditious solution to their task.

Syntactic Focus

Several methods for dealing with BOM composition and reuse focus solely on procedural, or syntactic, matching. One such attempt linked BOM processes to ontologies through the use of Simulation Reference Markup Language (SRML) [7], while another focused on matching BOMs in order to create compositions [8], with yet another approach being offered where SRML documents were automatically parsed to produce BOMs. [13]

Semantic Focus

Several approaches have focused on working with BOMs on a semantic level. One such proposal, called Semantic BOM Attachment (SBA) used features in OWL to link ontologies to BOM [14]. A similar approach was offered where BOM was extended, becoming BOM++, through reference to an external ontology [6].

In contrast to the current proposal, these previous approaches all altered the BOM in some way. The goal of the current proposal was to leave the BOM unaltered, focusing solely on expressing its features using an ontology created using OWL.

3 Designing DCMF-BOM Repository

3.1 Requirements for the DCMF-BOM Repository

The artifacts, which are identified during the DCMF process (see Section 2.1), such as Knowledge Components (KCs) and Knowledge Instances (KIs), need to be stored in a repository. The repository is recognized as a central component in DCMF because it is intended to store all the artifacts for use and reuse. As was briefly reported in the Section 1, the process of identifying KCs and KIs and their storage in a repository corresponds to Phase 2 in the FEDEP process.

The major functionalities of the repository concern:

- *Adding data,* which encompasses capabilities for creating and modifying artifacts, such as KCs and KIs;
- *Searching data,* which requires a semantic-level search for KCs and KIs to ensure their correct use and reuse, in single forms, or in compositions; and
- *Visualizing* the artifacts found in the repository. KIs, KCs, or whole Conceptual Models need to be presented to the user in a comprehensible format, a task for which images are particularly well-suited.

To facilitate the above requirements, we have proposed to design the DCMF repository as OWL ontology; we have also represented DCMF concepts (i.e. KCs and KIs) as BOM elements, i.e. well-established components for modeling of simulation requirements.

Finally, when conceptualizing KCs and KIs with BOM, we have taken a process-centric approach rather than the more common object-orientation. In this way we have established a natural alignment between the business requirements cases commonly considered as process-oriented and the computational model, i.e. the ontology structure. Modeling concepts from the process perspective facilitates considering them as possible services, when needed.

3.2 BOM as an Ontology

A BOM is comprised of a group of interrelated elements specifying metadata information, conceptual model information, class structure information defined using HLA OMT constructs, and the mapping between conceptual model elements and object model elements that identify the class structure information.

Setting aside the simple addressing and metadata that occupy several of these, DCMF and by extension, this paper, instead focus solely on the conceptual model. Thus, following DCMF's guidelines, the BOM specification was translated into an OWL-Full ontology using Protégé 4.

3.2.1 Class-level Mapping in BOM

Ontologies are based on `Is_A` relationships, and in OWL, the base for these is the concept `Thing`; all entities must possess an `Is_A` relationship to `Thing` and every individual must belong to at least one class (even if only `owl:Thing`) [3] Therefore, the uppermost level of the ontology consists entirely of `Thing`, and the secondary level contains all concepts which can relate directly to `Thing` via an `Is_A` relationship.

While at first it might make sense to create a secondary level that contains all basic concepts in BOM, which would be incorrect, due to the necessity for all secondary level items to have a direct, `Is_A` relationship with `Thing`. For example, a `Conceptual Model Definition Is_A Thing`. Placing its components at a tertiary level is not possible, since they do not have complete relationships with the secondary level; while a `State Machine Is_Part_Of` a `Conceptual Model`, it does not possess the all-important `Is_A` relationship. Therefore, all components must exist at a level that fully expresses their `Is_A` relationship. When this is done, the following structure is created, as seen in Figure 2.

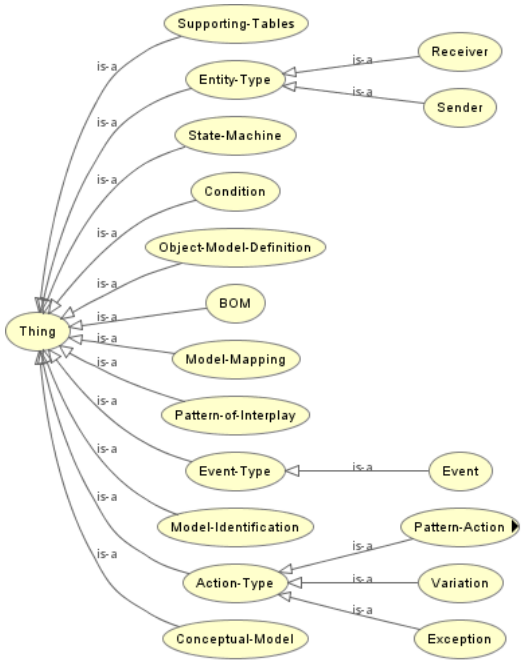

Figure 4: Entire BOM represented as an ontology

Under this basic structure, individuals can be asserted, providing they possess an `Is_A` relationship.

3.2.2 OWL Properties

OWL provides several methods for expressing information about components, primarily through Properties. However, because ontologies are relation-centric, the commonly accepted definition of 'property' does not apply exactly. It is easiest to think of OWL properties as representing relationships; of the three types of properties, the two most important—Object Properties and Data type Properties—are relation-centric. The third type, known as Annotation Properties, can be thought of simply as data fields.

OWL distinguishes between the two main categories of properties that an ontology builder may want to define, stating that Object properties link individuals to individuals while Data type properties link individuals to data values.

Object Properties

Object properties link an individual to another individual. For example, when expressing one of the precise relationships between the Entity `NavalCruiser` and the POI `Report-Coordinates`, the OWL statement would be `NavalCruiser isSenderOf Report-Coordinates`. Object properties can also have many characteristics (e.g., Functional) that could provide additional depth to the relationships, allowing for greater precision when implementing automated reasoning solutions. A functional characteristic—where there can be at most one individual that is related to the individual via the property—for the object property `sendsOrder` could be limited to the `Headquarters` class.

Data Properties

Data type Properties link an individual to an XML Schema Data type value or an rdf literal, thus describing relationships between an individual and data values. These relationships can then be used to enforce class membership; we could create data type properties such as `hasRange` or `hasBore` to restrict membership to the class `BattleCannon` from individuals whose range and bore were outside of the set parameters, for example an M-16.

Annotation Properties

Annotation properties can be used to add metadata to classes, individuals, and object/data type properties. OWL Full does not put any constraints on annotations in an ontology and has five predefined annotation properties:

```
owl:versionInfo
rdfs:label
rdfs:comment
rdfs:seeAlso
rdfs:isDefinedBy
```

To extend these basic forms, Dublin Core properties can be used directly as annotation properties, using special tags. For example, the class BattleCannon could be linked to the data literal(string) Grumman as part of the annotation property dc:creator.

3.2.3 Using OWL Properties with BOM

While creating the BOM ontology, it was essential to review the BOM Template Specification. This defines the format and syntax for describing the elements of a template for representing BOMs and also specifies the semantics of the elements of a BOM and the syntax for constructing BOMs from these elements. In addition, it provides a data interchange format (DIF) for the representation of BOMs using eXtensible Markup Language (XML). This BOM DIF should enable tools to exchange and reason about BOMs.

However, in reviewing the BOM DIF, it was noted that certain outcomes desired by the implementation of the BOM DIF could be more effectively carried out if it were expressed as an ontology using OWL, rather than as pure XML, in particular automated reasoning. For the purposes of implementing a repository within DCMF, this last capability was crucial. The ability to effectively compose and repurpose KCs and KIs depends on the ability for semantic discovery to occur, something that is most directly accomplished through the use of an OWL ontology.

Using part of the POI as an example, the mapping between category and property can be seen in Table 1. Each decision was based on the relationships contained within the ontology: if a category's property was expressible primarily in terms of a relationship, then it was considered an object property and if a property was expressible using an XML Data schema type, then it was assigned to an OWL Data property.

Figure 6: Mapping BOM Categories to OWL Properties

POI Category	OWL Property
POI	Data
Name	Data
Pattern Action	Object
Sequence	Object
Name	Data
Sender	Object
Receiver	Object
Event	Object
BOM	Object

4 Utilizing the Repository

The following scenario has been selected for possible submission into the DCMF repository. In the following sections we will walk through several possible benefits that would accrue from having an OWL ontology.

> A naval defense group, consisting of headquarters, a battle cannon and a radar station, was deployed to defend against incursions by enemy naval forces, primarily naval cruisers.
>
> Recently, the radar station detected an enemy naval cruiser and relayed that information to headquarters for processing. After ascertaining that the threat was legitimate, headquarters issued orders to the radar station and battle cannon to attack the enemy naval cruiser. The radar station reassessed the situation and plotted the coordinates again, relaying location to the battle cannon. Simultaneously, the battle cannon performed a readiness check, awaiting the coordinates from the radar station. When the radar station received the coordinates from the radar station, a further check was performed to gauge distance and projectile yield. After this check was performed with affirmative results, the battle cannon requested final confirmation from headquarters and, after receiving it, commenced firing, damaging but not destroying the enemy naval cruiser.
>
> During post-incident analysis it was learned that the particular battle cannon's limited range caused problems with targeting. This sub-optimal capability was deemed the primary cause for the failure to destroy the enemy naval cruiser. To increase range, changing the battle cannon and radar system from being land-based to being ship-based was proposed.

In this study, we do not address the parsing methods and their outcomes, as applied in the DCMF knowledge process (see Section 2.1.). Parsing is a complex activity, and for the sake of this study, we briefly state that first the parsed information is mapped to KC terms, such as process, entity, relationships; secondly the components are added to the taxonomy tree of BOM ontology through sub-classing. For instance, the processes elicited in the text, such as Locate-Carrier, Manage-Coordinates, or Attack-Carrier are parsed first as top-level processes that will correspond to BOM-level subclasses. After, the protocols (i.e. behavior), the entities and entity relationships utilized in those processes are elicited. Once the candidate terms are extracted, it is up to the ontology designer to add them to the ontology developed from the BOM specification, as presented in Section 3.

4.1 Adding

Using the text parsing methods outlined in Section 2.1, several interesting things could be discovered about the scenario.

1. *Classification:*
   An unclassified weapon possesses these and only these, characteristics: `hasRange` and `hasYield`. The class `Battle-Cannon` and `Missile-Warhead` both can have these as properties, although only `Battle-Cannon` requires them and only them. Therefore, the reasoner could conclude that the unclassified weapon was a member of the class `Battle-Cannon`.

2. *Subsumption:*
   When given these facts:

   ```
   leaders command forces
   commandants command cruisers
   cruisers are a force
   ```

   Therefore, commandants must be leaders.

3. *Class membership*:
   If `Howitzer-M198` is an instance of class

`Battle-Cannon`, and `Battle-Cannon` is a subclass of `Artillery`, then we can infer that `Howitzer-M198` is an instance of `Artillery`.

4.2 SEARCHING

Once the repository is filled with a number of KCs and related KIs, the Consumer will consider possibilities for reusing the repository content. In the DCMF context, two major usage scenarios are relevant: Searching for KCs and KIs of interest, and Searching for possible compositions of KCs and KIs to retrieve the conceptual models corresponding to complex requirements scenarios.

From the perspective of ontologies, searching for a containing concept or a set of them satisfying given constraints requires deductive reasoning to provide a correct answer. As explained in Section 2.2.4, conceptually, OWL can provide a number of reasoning capabilities, which can also be performed in Protégé via the Reasoner plug-in. However, for more complex inferences, the repository user often needs to complement OWL with more expressive rules, such as those supported by Semantic Web Rule Language (SWRL) [15]. The rules in SWRL are written in terms of OWL concepts (i.e. classes, properties, individuals and data values) to provide more powerful deductive reasoning mechanisms than with the core OWL. At its essence, OWL facilitates a number of mathematical and string operations that are used to evaluate truthfulness of given, user-defined predicates.

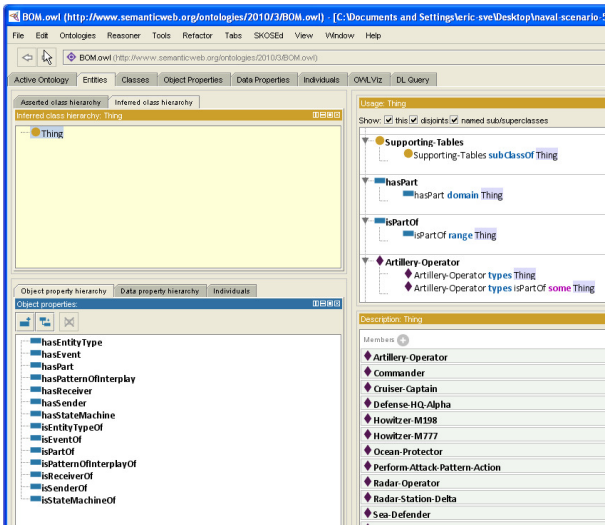

Figure 7: Protégé interface, showing inferred relationships

4.2.1 Searching for Knowledge Components and Instances

When searching the repository for a KC or a KI with desired features for use in a new requirement scenario, it is necessary to be able to search the repository for contents that could meet the requirements. In such a scenario, the benefits of implementing BOM in an ontology would be apparent, owing to the inherent capabilities of an ontology.

1. *Equivalence of classes*:
   If class `Naval-Cruiser` is equivalent to class `Naval-Frigate`, and class `Naval-Frigate` is equivalent to class `Naval-Carrier`, then `Naval-Cruiser` is equivalent to `Naval-Carrier` too.

   Although on many levels, these instances are quite different, the focus of the ontology might simply be whether they share the common characteristic of floating. This focus or ontological commitment is another key benefit of ontologies, allowing for very disparate objects to find commonalities among their relationships.

2. *Consistency:*
   Given the following information a reasoner could discover a logic inconsistency:

   ```
   Naval-Cruiser isPartOfDefense-Force
   Defense-Force defendsCoastline
   Naval-Cruiser attacksCoastline
   ```

   Such a construction could be made logically consistent by creating a new property `isHostileTowards` so that the different types of `Naval-Cruisers` (both those hostile and friendly) can be captured.

4.2.2 Searching for Compositions of Components and Instances

A number of requirements scenarios require combining two or more KCs or KIs to realize the given requirements with a single conceptual model. Here, one may distinguish two types of needed compositions: *vertical* and *horizontal*.

Vertical Composition

The *vertical composition* of KCs and KIs is used when an action contained in the component needs to be further decomposed. The purpose is to refine the final conceptual model to capture more level of details, by following a given simulation requirement case.

A vertical composition can substitute an ActionType class, i.e. PatternAction, Variation, or Exception with an entire KC class (which is itself a BOM). In the requirements scenario in Section 4, `Perform-Attack` pattern action of the interplay `Attack-Carrier` and the corresponding KC are typical examples of the activities that may need to be decomposed further to model their more detailed tasks and entities, such us e.g., navigation, firing, and so forth. Once the rules for examining the ability of replacing an ActionType of a KC class with an entire KC class are established, they can be further checked on the instance level, to see which individuals can obey the vertical composition rule.

To derive the rule for a vertical composition, there must be a Knowledge Component X ($KC_{x)}$ with an ActionType class such as PatternAction with the sequence number n ($KC_x\_PA_n$) examined for refining by another Knowledge Component. As discussed in Section 2.2.3, a pattern action is determined by its associated event; the event (of message type, for instance) in turn is defined by its characteristics, such as Sender, Receiver and Content. Thereby, we define the composition requirement as:

> *Search for the Knowledge Components having an Event class equivalent to the Event class of KCx_PAn, i.e. where the Sender, Receiver and Content associated with both events are equivalent.*

The requirement is examined using a sequence of OWL inferences, and complementing SWRL statements:

1. Running the class equivalence reasoning between the Sender and the Receiver classes of the $KC_x\_PA_n$, event, and the events of all other KCs to find possible matches.
2. Running a SWRL statement to examine the equivalence of Content data properties of the characteristics of the two events selected in 1, i.e. if they concern the same entity and if the entity changes as specified in the Content fields are the same.

   For instance, if the events matched in 1. are concerned with reading the location of the carrier, where the Sender is `Officer` and Receiver is `Cannon`, and if in both events it is specified in Content Characteristic `Location.Destination` where `Location` is the Entity and `Destination` is a Data field within it, then those events are considered as equivalent.

After running the above rules, if one or more candidate KCs are found, it is further possible to automatically check which KIs (i.e. individuals) can fulfill the established class-level rule.

Horizontal Composition

A *horizontal composition* of KCs occurs when two or more components are chained to model a more complex simulation requirement process. Chaining means imposing certain flow order between the KCs. For instance, from our requirement scenario, the identified KCs (i.e., `Locate-Carrier`, `Attack-Carrier`, `Manage-Coordinates`) should be chained to satisfy the whole requirement. The flow order is commonly based upon some data or object dependency requirements; in the previously given example, `Manage-Coordinates` must precede `Attack-Carrier`, because the latter requires the coordinates of the carrier.

Assuming that two Knowledge Components, KC_x and KC_y should be examined for chaining in the way that KC_y follows up KC_x, the composition requirement can be formulated as:

> *Find the equivalent entities of KC_x and KC_y and check if the KC_x provides the data/objects in those entities required by KC_y by investigating the content characteristics of the events associated with the entities in both KC_x and KC_y.*

In OWL terms, this requires the following steps:

1. Selecting the entities from KC_y and matching them against the entities in KC_x using the class equivalence inferring (by checking if the all containing individuals are the same).
2. Compare the Content Characteristics of the entities matched in 1 for equivalence by running a corresponding SWRL rule.

As discussed for the vertical composition, once the candidate KCs are determined, automated reasoning can be use to find the individuals of those components that can be chained.

4.3 Visualization

The ability of a framework to utilize current mainstream practices is critical to its successful adoption. Particularly when working with business users, it is important that processes and tools speak a language they are able to quickly understand and accept.

Protégé offers several useful visualization tools, among them OWL2UML [12]. As shown in Figure 8, classes

along with their object and data properties are easily rendered into UML diagrams.

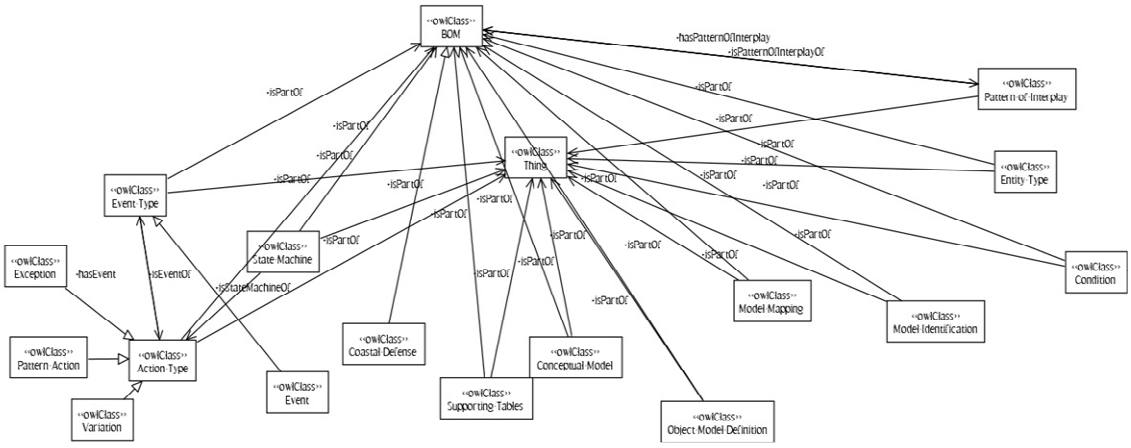

Figure 8: BOM Ontology, visualized in UML

5 Conclusion and Future Work

The goal of this study has been the design of a repository for the concepts from the Defence Conceptual Modeling Framework (DCMF) of the Swedish Defence Research Agency (FOI), to facilitate use and reuse of the concepts. In this task, we have been interested in using the Base Object Model (BOM) to shape the conceptualization of the DCMF elements. For the effective use of a DCMF repository, the most significant features concern storing and retrieving of the contained concepts (i.e. Knowledge Components and Knowledge Instances) in clear and unique ways and with the capabilities to compare those concepts, elicit their various relationships and finally, correlate them in compositions to model more complex requirements for simulations.

These needs have motivated the consideration of ontology expressed through OWL as a tool, because it can facilitate the semantic-level classification for stored DCMF data as well as reasoning tasks using semantic-level query statements. OWL is a widely-used and established language, using which we have captured: a) the declarative knowledge of DCMF-centered BOM components, and; b) classified the components to enable an automatic search and composition using automatic inferences supported on the OWL level, as well as more complex reasoning enabled by SWRL, a rule language layered on top of the OWL. We have implemented and tested the proposal using Protégé ontology designer and several of its plug-ins to support reasoning.

The benefit of our proposal lies in the use of the original BOM standard to facilitate the semantic-level operations needed for the DCMF repository, by designing a BOM ontology to capture the semantics of the concepts without fostering extensions in the BOM notation. As for the future research, we plan to consider the OWL and SWRL capabilities for further refinements in the exploitation of the elements of the entire BOM ontology to obtain richer semantic reasoning and inferences.

6 References

[1] Simventions: "Homepage of Base Object Model" Available online at http://www.boms.info Last accessed 2010-03-11

[2] V. Mojtahed, M. Garcia-Lozano, P. Svan, B. Andersson: "DCMF – Defence Conceptual Modeling Framework. Methodology Report" FOI-R--1754—SE, ISSN 1650-1942, 2005.

[3] World Wide Web Consortium (W3C): "OWL Web Ontology Language Overview" W3C recommendation 10th February 2004. Available online at http://www.w3.org/TR/owl-features/ Last accessed 2010-04-20.

[4] Protégé Ontology Editor: Open source tool hosted by Stanford University. Available online at http://protege.stanford.edu/ Last accessed 2010-03-20

[5] Simulation Interoperability Standards Organization (SISO): "Guide for Base Object Model (BOM) Use and Implementation" SISO-STD-003.1-2006, 2006.

[6] V. Mojtahed, B. Andersson, V. Kabilan, J. Zdravkovic: "BOM++, a semantically enriched

BOM" Spring Simulation Interoperability Workshop, 2008.
[7] F. Moradi, P. Nordvaller, R. Ayani: "Simulation Model Composition using BOMs" Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '06), 2006.
[8] I. Mahmood, R. Ayani, V.Vlassov, F. Moradi: "Statemachine Matching in BOM based model Composition" 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, 2009.
[9] M. Garcia-Lozano, V. Mojtahed, P. Svan, B. Andersson,V. Kabilan: "Konceptuell Modellering inom det Svenska Försvart-DCMF" Swedish Defence Research Agency, FOI-R--2115—SE, ISSN 1650-1942, 2008.
[10] V. Mojtahed, E. Tjörnhammar, J. Zdravkovic, A. Khan: "The Knowledge Use in DCMF, Repository, Processes and Products" FOI-R—2606—SE, ISSN 1650-1942, 2008.
[11] J. Dietz: "Enterprise Ontology, Theory and Methodology" Springer–Verlag, Berlin/Heidelberg, Germany, ISBN 978-3-540-29169-5, 2006.
[12] OWL2UML Plugin, Available online at http://apps.lumii.lv/owl2uml/ Accessed 23 March 2010.
[13] F. Moradi, R. Ayani, S. Mokarizadeh, G. Shahmirzadi, G. Tan: "A Rule-based Approach to Syntactic and Semantic Composition of BOMs" 11th IEEE Symposium on Distributed Simulation and Real-time Applications, 2007.
[14] F. Moradi, R. Ayani, S. Mokarizadeh, G. Tan: "A rule-based semantic matching of base object models" Int. J. Simulation and Process Modelling, Vol. 5, No. 2, pp.132–145, 2009.
[15] World Wide Web Consortium (W3C): "SWRL: A Semantic Web Rule Language Combining OWL and RuleML" W3C Member Submission 21 May 2004. Available online at http://www.w3.org/Submission/SWRL/. Last accessed 2010-04-15.

Author Biographies

VAHID MOJTAHED is a senior scientist and deputy research director at the Swedish Defence Research Agency (FOI). He received a Master of Science in Computer Science and Engineering from Chalmers University of Technology in 1994. He has been working on modeling and simulation for the past 16 years and has led the Swedish Conceptual Modeling research at FOI since 2001 as well as the FOI research on Semantic Interoperability since 2006. He is also the Swedish representative in NATO's research group on Conceptual Modeling as well as project manager for the Swedish Conceptual Modeling project, DCMF. His research interests are in the area of Conceptual Modeling, Simulation Framework, Knowledge Representation, Semantic Interoperability and Information Operations and Warfare.

ERIC-OLUF SVEE is a research engineer, employed by Stockholm University and contracted to the Swedish Defence Research Agency (FOI). He received his Master of Science in Interactive Systems Engineering from the Royal Institute of Technology (KTH) in 2008. Prior to returning to academia, Eric-Oluf's work focused on the information management and design, as well as accessible design. His most recent position was with the Swedish Institute of Computer Science (SICS) where his research was focused on delivering personalized services to mobile devices in a privacy sensitive manner.

JELENA ZDRAVKOVIC has a doctorate in the field of process modeling and integration, and her current research continues to focus on business and process integration with e-services. In addition, Jelena holds a Masters in Business Administration, with a concentration on e-business. In recent years, Jelena has participated in several national and European research projects. She is a PC member/co-organizer of several international conferences and workshops. Besides academic qualifications, Jelena has experience as a consultant in systems modeling. Jelena now works as senior lecturer and researcher at the Department of Computer and Systems Sciences (DSV), Stockholm University/KTH in Sweden.