# The Supporting Role of Ontology in a Simulation System for Countermeasure Evaluation

Nelia Lombard
Council for Scientific and Industrial Research, Pretoria, South-Africa
nlombard@csir.co.za

**Abstract:** Military aircraft in the battlefield are under constant threat from missile attacks. Technology such as infra-red countermeasures are used to protect against these threats. Measurements, modelling and simulation are used by the military to develop, implement and evaluate these countermeasures. The ability to simulate what happens in real life scenarios is a powerful way to evaluate countermeasures. These simulation systems are complex and lack a common vocabulary. Ontology is an emerging discipline providing a mechanism to provide common domain vocabularies. Ontologies in information systems and engineering is maturing after years of use and the research described in this paper investigates the possible advantages that ontology might have in the simulation environment. We also investigated how ontologies can provide a solution to some of the challenges to be dealt with in a countermeasure simulation system. The construction of an initial ontology showed that the ontology not only provides a high-level description of the concepts in the simulation system, but it also supports the validation of the use of the simulation models in the system.

**Keywords:** Ontology, Simulation, Countermeasure, Model, Concept.

## 1. Introduction

Weapons guided by optical radiation present a serious threat to military aircraft participating in various types of military operations. Today, even the smallest terrorist group can get hold of shoulder-launched surface-to-air missiles and use it to attack military aircraft. This threat is not only applicable in times of war but also when they are involved in peace missions carried out across the globe and especially in Africa. In order to protect aircraft against these missiles, defensive countermeasures have been developed and implemented on the aircraft (Figure 1).

**Figure 1: Missile threat against target**

The design, development, implementation and testing of these countermeasures are a complex and costly exercise. It is not viable to test every scenario in field trials but by utilising simulation software, it is possible to simulate what happens in the real world and to test various scenarios that may not be feasible or desirable to conduct with live tests [2]. Lord Kelvin said “To *measure* is to know”. The knowledge gained by measurement can be used to construct models that represent real-life objects. Simulation systems can, by using the *data measured* and the *models built*, simulate what happens in the real world. Simulation therefore enables the saving of time and money without having the environmental impact of live tests.

Optronics Scene Simulation (OSSIM) is a computer-based scene simulation system environment that provides a wide spectrum of applications such as scenario analysis and signature modelling [10]. The focus of this research is the application of OSSIM to do evaluation studies of infra-red countermeasures. Infra-red countermeasure evaluation assists in optimising the deployment strategy of countermeasures on aircraft and provide input to the operational doctrine of a specific client. The simulation environment is a complex system that involves many stakeholders and needs the knowledge of domain experts to be successful. One of the most important factors for success is adequate communication between the developers of the system, the users and these domain experts. Ontology is an emerging discipline providing mechanism to provide common domain vocabularies.

An ontology provides a shared, common understanding of the concepts and terminology in a domain, thereby ensuring that all the stakeholders communicate effectively [1]. In addition, an ontology captures the meaning and constraints of objects of the domain formally. Formal ontologies thus provide a machine-readable vocabulary that can be used by a computer system to communicate between software modules and act as an interface between human and machine.

Given the complexity and challenges of the simulation environment as mentioned, and what ontology technologies provide, it is feasible to argue that an ontology might be able to address the identified shortcomings of the simulation system. In this research the focus was on how ontologies can be used to capture the domain knowledge and present it on a high-level of detail in order to use it as a shared, common vocabulary of communication.

This paper is structured as follows. Section 2 provides background of firstly, the simulation system and secondly a description of ontology technologies. Section 3 describes the process followed to develop an ontology for the simulation system. While constructing the ontology for the simulation system, lessons were learned and experiences gained and these are discussed in Section 4. In Section 5 the contributions of this research are discussed. Section 6 concludes this paper by discussing the benefits to the simulation environment provided by formal ontologies and how this will indeed makes a positive contribution to the countermeasure evaluations.

## 2. Background and Technology

### 2.1 The Countermeasure Simulation System

The OSSIM simulation system is a software environment wherein software components use realistic military models and their behaviour to simulate what happens in the real world. The models are brought together to run interactively to offer the following functionality:

- Simulate real-world behaviour
- Simulate interaction between models as a result of certain events
- Accelerate countermeasure research
- Assist in planning field trials
- Simulate what might not be possible in field trials

The models are the main components of the system and adds real value to the system. Aircraft and infra-red countermeasure flares are measured in the field and the models built by using the measured data. These models are geometrically and radiometrically accurate to present real-life objects and their behaviour. Atmospheric and terrain models are built to simulate the effect that the environment will have on model behaviour in the simulation. Mathematical models of the missiles are built to simulate the threat against aircraft.

An evaluation study typically consist of a request from a client having a need, for example, to evaluate a surface-to-air missile against an aircraft using a specific type of countermeasure such as a flare. To run a simulation, the following inputs are needed:

- Type of aircraft: e.g. a Mirage
- Mission flight plan: How will the Mirage fly?
- Type of missile: e.g. AS30
- Type of countermeasure and the dispensing logic: The dispensing logic determines how and when the flares must be ejected
- Atmospheric conditions: e.g. clear skies or fog
- Terrain model

Once the specific input requirements are obtained, the simulation is set up by creating a set of XML files that contains all the information required to run a simulation. The XML files are hierarchically linked to each other by means of model names. In the scenario file, for example, it will be specified that a Mirage will be used, and a tag will indicate which file contains the information of the Mirage model.
The different software components inside OSSIM use the information in the XML files as input to the simulation modules. Figure 2 displays an example of a scenario file. The different XML elements in the file describe the models that will take part in the simulation run for this scenario.

```
<?xml version="1.0" ?>
<!-- $Id: tp06a.xml 77 2009-06-04 13:30:20Z nelis $ -->
<!-- $HeadURL: svn://146.64.246.11/dpss_cms/trunk/dpss/dpss_test/TestPoint06/tp06a.xml $
-->
<Scenario Name="TestPoint06a" ID="" Version="" Latex="true">
   <TestPoint Basename="tp06a" />
   <PathCache PrintPaths="false" >
            <DataPaths COMMENT="Optional, only used if present" >
                  <DataPath SearchOrder="1" Path="../.." COMMENT="SearchOrder 1
means first"  />
   </PathCache>
   <TestFiles>
      <TestFile Type="rad" COMMENT="rad thm stat m" />
   </TestFiles>
   <DebugLog Level="warn" Filename="dpss_test\TestPoint06/AllLogsProfile.xml"  />
   <LeakDetector Active="0" COMMENT="1=true,0=false" />
   <Clock Filename="dpss_test\TestPoint06/Clock.xml" />
   <Observers>
      <Observer Type="CBaselineMisl" Filename="dpss_test\TestPoint06/AS30.xml" />
   </Observers>
   <Movings>
      <Moving Type="CBaselineFlare" Filename="dpss_test\TestPoint06/FlareA.xml" />
      <Moving Type="CBaselineFlare" Filename="dpss_test\TestPoint06/FLAREMA.xml" />
      <Moving Type="CBaselineFlare" Filename="dpss_test\TestPoint06/FLAREMBB.xml" />
   </Movings>
   <GroundAltASL Height="0.000000e+000" />
   <LatLongLocation Latitude="-2.544000e+001" Longitude="-2.811000e+001"
COMMENT="latitude [-90;90],longitude [-180;180]" />
   <Radiometry SunComponent="1" COMMENT="1=true,0=only thermal" />
   <Atmosphere Filename="dpss_test\TestPoint06/Atmo.xml" />
   <Renderer   Superres="true" Comment="true or false" />
   <Rasteriser Method="Hecker" ZBuf="0" COMMENT="(SIMIS Hecker) (0=no, 1=yes)" />
   <Cyclops Connect="0" COMMENT="0=no/1=yes" />
</Scenario>
```

**Figure 2: Example of scenario input file**

The output of the simulation is written as XML files. These XML files are used to construct visual output as requested by the client, and can be processed in a number of ways, one of which is illustrated in Figure 3. This example displays a snapshot from a 3D viewer, showing how the missile tracks the target in the scene.

Although the simulation system is applied successfully to do countermeasure evaluation, there is always a need to improve and enhance. The following is a list of needs that were identified by users as well as developers of the system.

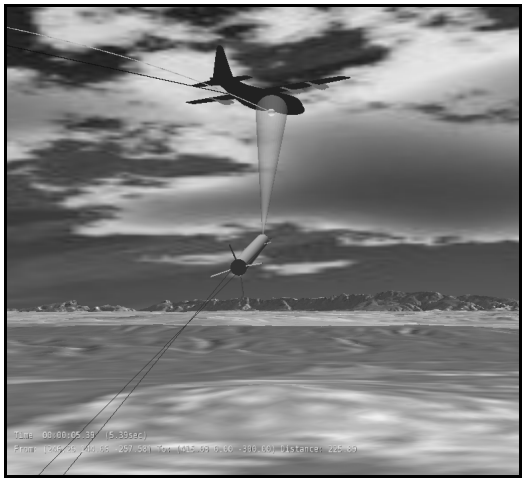

**Figure 3: Example of scenario output**

**A need for a high-level description of a scenario**

The ability to describe a simulation scenario on a high-level, using consistent terminology, will greatly improve the communication between developers, users and domain experts. Carson [5] names this group of people the *Simulation Team* and describes how important it is that people with different skills and expertise are part of the team. It is not necessarily the same people or person that develop and build new models, set up a simulation, write the report or use the result but they need to work together and speak the same technical language.

**To know what is available in the system**
During the setup of a specific simulation study, it is possible to set several input parameters for each model. It will be very useful to have all the possible models and model parameters available when creating setup files. This will allow for the setup files to be validated even before the simulation is run and will be valuable for users creating scenario files as well as the software modules that read the scenario files.

**Guideline for specification of new models**
The success of a simulation depends firstly on how good the models are. If the models are not behaving correctly, the simulation results will not be accurate and the simulation results useless. New models are constantly being added to the simulation system by different clients or users. It is important that these models adhere to existing standards and rules. A client who wants to develop its own model to be integrated into the simulation system, needs that set of rules to adhere to.

**Must be possible to verify and validate model interaction in a scenario**
A function is needed to validate model interaction in the simulation setup according to the rules of the simulation system. Models behave in a certain way and although it might be possible to set up a simulation that is programmatically correct and produce no errors, it might not be correct according to the rules of behaviour for that specific model, thus the specified behaviour might be unrealistic or even impossible.

**Function to reverse engineer previous simulations**
A method is needed to construct high-level descriptions of previously run simulations by investigating the simulation files and match the objects to classes in the domain.

In this paper, we investigate how ontology can be used to address the issues mentioned above. In order to do that, ontology is explained in more detail in the next section.

## 2.2. Ontology explained

Ontology was originally more used in the social sciences than in information sciences. Therefore it is appropriate to use an illustrative example from a social perspective to explain the concepts. Imagine a group of children playing a game. The playing children have certain words, phrases and rules they use so that each of them knows how to take part in the game. If an intelligent robot is built to play the game with the children, it will have to know the terminology and the rules. If the robot can be provided with a formal description of the concepts of the game and how to play, it will be possible for it to take part and interact with the children during the game. The robot cannot read the information on a human level, it needs to be provided with a machine-readable language that is also understood by the children. Ontology is what the robot needs to in order to play. It is a term used to describe the formal conceptualisation of the knowledge in the domain and in information systems it is the knowledge that is necessary to complete the tasks of the system.

Although ontologies for many domains exist, it can still be applied in a much more practical way in real-life systems [8] and especially in simulation systems such as OSSIM. There are several advantages to use ontologies in information systems. It provides a structured way to analyse the domain knowledge. It further enables reuse of that knowledge and provides a way to share it between software modules and users. Ontology engineering provides reasoning to check consistency of classes and to provide a query mechanism that can find, for example, individuals matching a given criteria [4].

Ontologies use XML language to describe the underlying structure of the information captured but do not replace a descriptive language such as XML Schemas or a modelling paradigm such as object-orientated design. XML Scheme definitions are XML documents and provide structure, but not meaning, and as explained by [6], XML Schemas is useful to define valid document structures but cannot be compared to ontologies. Object-orientated design on the other hand, focuses on the tasks to be done and structure the design around methods of classes that fulfil a specific task as described by [9], whereas ontologies are based on logic.

## 3. Construction of the ontology

As mentioned previously, the focus of this paper is to reflect on how an ontology can assist in the capturing of specific domain knowledge. In the first part of this section, we look at how the ontology was constructed and in the second part, the constructed ontology is discussed.

### 3.1 Modelling procedure to construct ontology

The procedure to construct the ontology is described by [7]. The most important rule when doing ontology construction is to always keep in mind the purpose of the ontology. “Where and how will it be used?” “Will it add any real benefit?”
The following five steps were used to construct the first version of an ontology for the simulation system domain.

**Decide on the purpose and scope of the ontology**
The main purpose of this ontology is to provide a high-level description of the simulation system and therefore the ontology covers the concepts as well as the way the concepts relate to each other in the system. The scope of the ontology is determined by those concepts making up a scenario as described in Section 2.1. The intended use of the ontology is focused on successful setup and description of a simulation so by including this, adequate information will be available to fulfil the purpose.

**Identify the classes**
Classes are the main building blocks of an OWL ontology, and are the sets that contain the individuals. Because of the scenario being the main components of the system, the classes were identified by making a list of all the possible participants in a scenario, but limiting it to the high-level concepts, those concepts that have independent existence, apart from them begin part of a scenario. Each of these classes were then looked at in more detail to identify a class hierarchy. The names of the classes were chosen to present the name of a group of models build for the simulation. The following step was to organise these classes in a hierarchical taxonomy by deciding if a class is a

kind-of other class, for example, every `Helicopter` is a `Target`, therefore the `Helicopter` class is a subclass of class `Target`.

**Define object properties**
Object properties are what links the objects in the domain to each other. How does a flare relate to an aircraft? Important to note is that properties are defined on class level, but it describe a relation between two specific individuals.

**Define data properties**
Classes alone are not enough to describe all the information that needs to be captured. Data values are linked to classes, giving them internal structure. Every class need to have enough properties to define that class in adequate detail to be able to use the class successfully in a simulation.

**Create individuals**
These are the objects in the domain that the users are interested in. The specific aircraft that might be flying and the instance of a real terrain it is flying over. Instances of a scenario will be unique to a specific simulation study but those instances that are part of a scenario will be reused in different scenarios. For example an instance of `Terrain` class, `Ocean`, will be used in more than one simulation study.

### 3.2 Discussion of the constructed ontology

Using the process described, a Web Ontology Language (OWL) ontology was created using Protege 4.1, which is used successfully in simulation modelling in the medical field, as described by [3].

The classes created in the ontology are displayed in Figure 4. The `Scenario` class is the main class that contains all other classes. The other classes are not a type of scenario, but they are defined as being part of a scenario.

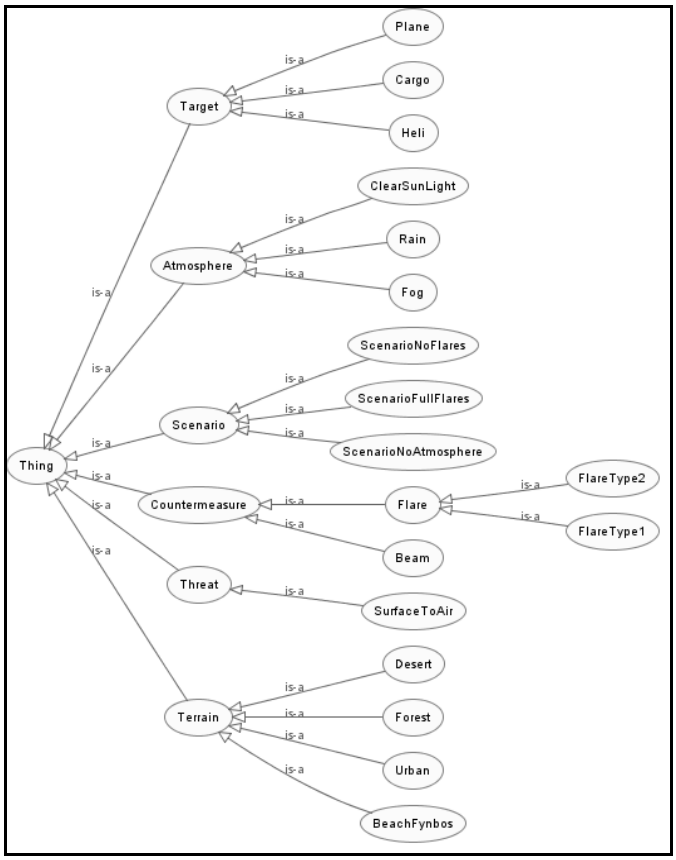


**Figure 4: Classes created in the ontology**

**Object Properties**

Object properties were defined to indicate those classes that are part of a scenario. A `Helicopter` or an `Atmosphere` cannot exist in the simulation on their own, it must be part of a `Scenario` in the simulation. Other properties defined were for example the `hasCountermeasure` property that define the relation between a countermeasure and a target, as shown in Figure 5.


**Figure 5: Example of object property**

**Data Properties**

Data properties were defined for all the parameters describing a specific class. Figure 6 illustrates an example of a data property, the Position property, created to hold the position of a Target or a Threat and consist of three sub properties for indicating position of an object by specifying three values (xPosition, yPosition and zPosition).

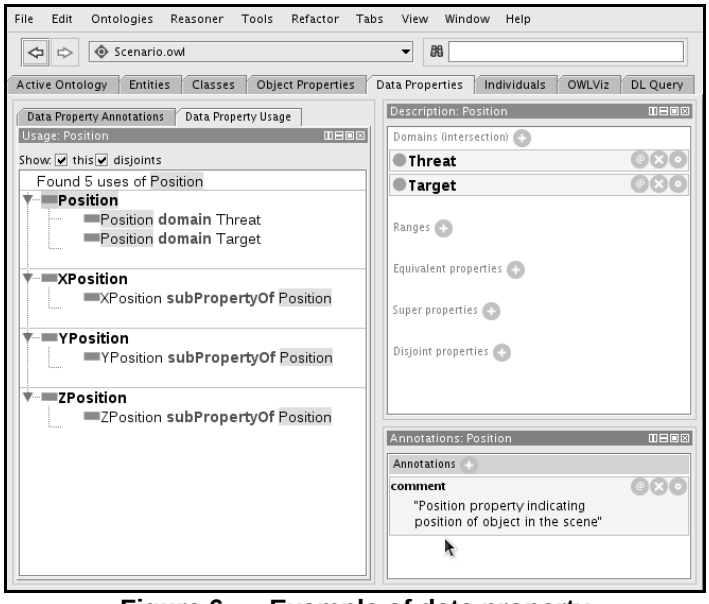


**Figure 6: Example of data property**

## 4. Discussion of Experiences and Lessons Learned

During the ontology construction, there were some choices to be made, difficulties encountered and interesting findings that are discussed in more detail in the next few paragraphs.

**Naming of classes**
In order to construct a common vocabulary it is important to have consistency of class names. A concept of a platform can have a different meaning, depending on the way it is used. An agreement must be reached and class names used that will be meaningful to everyone using the ontology.

**Classes versus instances**
In Section 3, there is a description of how the classes were identified but the choice of something being classified as a class versus a specific instance of a class is not always easy. The purpose of the ontology must be used as a guideline. The top-level classes present no problem, but going deeper down the hierarchy it is not always clear as to what must be defined as a class or rather an instance. An example of this is the concept of a flare that is mounted on the left side of an aircraft. This can be defined as a class, `leftFlare`, or an instance of the class `Flare`, `leftFlare.` Another example is the `Atmosphere` class. The purpose of the class in the simulation is to capture the effect of the environment on the simulation. A typical instance is `MidAlitudeSummer` that describes a certain condition, but it might be necessary to create a new class.

**Events and relations**
An example of an event in the simulation domain is when a missile locks onto a target. There is thus an object relation between an aircraft and a missile, called `lockonTarget`, that model an event. According to [7], this is modelled as an object relation, because no timing is involved. For a simulation run in the system, the missile is always locked onto the target. If this is not the case, an error will occur. Modelling an event in this way pose a problem: if there is more than one target in the scene, there might be a platform without a relation with a missile. Alternatively, implementing this event as a relation, it implies that a missile must lock on to a target and thus enforcing a rule and prevent having missile and targets not related in a meaningful way. This demonstrates how initial modelling choices can influence the later application and use of the ontology.

**Modelling roles as classes**
A class can loose its role over time. A missile looking at a specific target at a specific height, has a property to indicate where it is looking. If that property changed, that role is lost. If however, an instance of a `Missile` was used in that role, it can be stored and retrieved later. A way must be found to solve these temporary roles of classes.
In some cases, although there exists a relation between two classes, instances of those classes are not allowed to have any relation between them. For example, only certain types of countermeasures can be used against certain types of missiles and to model that in the ontology, constraints can be used which are very effective in verifying the correct meaning of a simulation.
Ontology supports the management of a huge amount of instances created for different simulation runs. It can be reused in different simulations by different scenarios. The ontology constructed so far was only done as initial research and it will be expanded further. Although it is not yet completed, it already demonstrates support for the simulation system, as discussed in the next section.

## 5. Contribution

In Section 2.1, a list of issues was mentioned that will improve the simulation environment. To have a shared, common language of understanding for the domain was identified as the most valuable contribution. To create a useful ontology, it was necessary to have a thorough investigation into what exits in the domain and how to describe it. This process, as well as the created ontology, provides a high-level description of the domain, in a shared, common language which can now be used successfully as a tool for communication.

## 6. Conclusion

The idea to investigate the use of ontology in the countermeasure simulation system came from a need to have a clear, common language that describes the concepts in the domain of the system. Not

only in the form of proper documentation, but to have a technology that will capture the meaning of the objects that represent the models in the simulation.
In the simulation environment, it is possible to specify a scenario that is syntactically correct, but the behaviour of models and how they interact is incorrectly specified. Ontology prove to be a solution that enables the possibility to validate the simulation scenario in terms of meaning. Questions such as: “Are the behaviour of the models true to real life situations?”, can be addressed by making use of the rules specified in the ontology.
Research was done and as a result an ontology was constructed that provides a formal description of the terminology in the simulation domain. The important goals that were reached were the encapsulation of complex system descriptions into human-readable information as well as now having a formal way to present this information that was created.
The experience gained by the work done so far, will provide valuable input into future developments planned to enhance the simulation environment with the advantages that ontology engineering provides.

**References**

[1] A. Boury-Brisset, “Ontological engineering for threat evaluation and weapon assignment: a goal-driven approach”, in *Proceedings of the 13th International Command and Control in Research and Technical Symposia*, 2008.

[2] S Joseph, FPJ Le Roux, JP Delport and OC Lombard, “Infrared modelling and simulation”, SAAF Work Session, 2004.

[3] J. Ekberg, J. Jenvald, H Eriksson, M. Morin and T. Timpka, “Simulation modeling using Protege”, Technical report, Dept of Computer and Information Science, Linkoping University, 2008.

[4] I. Horrocks, “Description logic reasoning”,13th International Conference on Conceptual Structures, 2005.

[5] John S Carson II. “Introduction to modeling and simulation”, in Proceedings of the 36th conference on Winter simulation, 9-16, 2004.

[6] F. van Harmelen, M. Klein, D. Fensel and I. Horrocks, “The relation between ontologies and XML schemas”, 2001.

[7] Natalya F. Noy and Deborah L. McGuiness. Ontology development 101: A guide to creating your first ontology. Technical report, Stanford Knowledge Systems Laboratory, 2001.

[8] K. Akella, P. Benjamin and A. Verma, “Using ontologies for simulation integration”, in *Proceedings of the 2007 Winter Simulation Conference*, 2007.

[9] Dr. Waralak V. Siricharoen, “Ontologies and object models in object oriented software engineering”, in *International Journal of Computer Science*, 33:19–24, 2007.

[10] MS Wheeler and CJ Willers, OSSIM User Guide. Technical report, DPSS, CSIR, 2007.