The Construction and Use of an Ontology to Support a Simulation Environment Performing Countermeasure Evaluation for Military Aircraft

by

Orpha Cornelia Lombard

(Student Number: 0709-006-4)

submitted in accordance with the
requirements for the degree of

Magister Technologiae

in the subject

Information Technology

at the

UNIVERSITY OF SOUTH AFRICA

Supervisor: Dr. Aurona Gerber
Co-supervisor: Prof. Alta van der Merwe

May 2014

Abstract

This dissertation describes a research study conducted to determine the benefits and use of ontology technologies to support a simulation environment that evaluates countermeasures employed to protect military aircraft.

Within the military, aircraft represent a significant investment and these valuable assets need to be protected against various threats, such as man-portable air-defence systems. To counter attacks from these threats, countermeasures are deployed, developed and evaluated by utilising modelling and simulation techniques. The system described in this research simulates real world scenarios of aircraft, missiles and countermeasures in order to assist in the evaluation of infra-red countermeasures against missiles in specified scenarios.

Traditional ontology has its origin in philosophy, describing what exists and how objects relate to each other. The use of formal ontologies in Computer Science have brought new possibilities for modelling and representation of information and knowledge in several domains. These advantages also apply to military information systems where ontologies support the complex nature of military information. After considering ontologies and their advantages against the requirements for enhancements of the simulation system, an ontology was constructed by following a formal development methodology. Design research, combined with the adaptive methodology of development, was conducted in a unique way, therefore contributing to establish design research as a formal research methodology. The ontology was constructed to capture the knowledge of the simulation system environment and the use of it supports the functions of the simulation system in the domain.

The research study contributes to better communication among people involved in the simulation studies, accomplished by a shared vocabulary and a knowledge base for the domain. These contributions affirmed that ontologies can be successfully use to support military simulation systems.

Keywords: ontologies, knowledge base, military simulation, design research, adaptive methodology, OWL, Protégé

Declaration

I declare that

The Construction and Use of an Ontology to Support a Simulation Environment Performing Countermeasure Evaluation for Military Aircraft

is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references. I further declare that I have not previously submitted this work, or part of it, for examination at Unisa for another qualification or at any other higher education institution.

Orpha C Lombard

Acknowledgments

I would like to give thanks to the following people for their support in assisting me to complete this research project:

- Aurona and Alta, my supervisors, for guidance, patience and for always being available to assist me in any way.
- My colleagues in the simulation team for teaching me about simulation and explaining military terms and technical detail.
- The CSIR for financial support and creating an environment to study.
- My family who never cease to encourage me and for them just assuming I will continue and persevere.

Contents

List of Tables

List of Figures

Terms and Concepts

Several terms and concepts are used throughout the document and are defined here for clarity and readability.

- **Business Object Reference Ontology (BORO)**
  Approach to developing ontological or semantic models for large complex operational applications that consist of a top ontology and a process for constructing the ontology [5].
- **Countermeasures**
  A a collective term for electronic measures implemented by the military to protect personnel and assets against enemy attacks.
- **Computer Science (CS)**
  Used throughout the document to refer to the study of computer hardware, software and technology and covers all aspects of computing and information sciences.
- **MANPADS**
  Man-portable air defence systems are surface-to-air missiles used to attack aircraft. Can be fired by a single person (shoulder-launched).
- **OSSIM**
  Optronic Scene Simulator is a computer-based engineering tool that performs image-based rendering of visual and infrared scenes [97].
- **Protégé**
  An open-source editor that is used to construct, edit and maintain the ontology.
- **Scenario**
  A scenario describes a single simulation that is performed in the simulation system to test an aircraft against a threat, using specific countermeasures defined in the scenario file.
- **Simtology**
  The name given to the ontology constructed for the simulation system environment.
- **Simulation System Environment**
  This describes the simulation software, procedures, documentation that the simulation system operates in.
- **XML**
  A self-descriptive mark-up language designed for data exchange ([16]).

CHAPTER 1

Introduction

This document describes a research study conducted to determine the benefits and use of ontologies to support a simulation environment that evaluates countermeasures implemented as protection on military aircraft.

Aircraft are important and expensive assets for military forces. Huge investments are made into these assets and especially into protecting them. Protection is needed from a range of threats such as anti-aircraft missiles fired from the ground to the airborne aircraft. These missiles, known as man-portable air defence systems (MANPADS), are widely available in current as well as old war-zones, are relatively cheap and easy to operate. Although not expensive weaponry, their mechanisms are complex and continually being upgraded to withstand improvements and developments in aircraft countermeasures [14]. The military needs to better understand these missile systems in order to develop successful protection against them. The missile systems vary from one another and an understanding of how each type of missile reacts in aircraft engagement is crucial in the development of aircraft protection [11]. It is therefore inevitable that effective protection against these missiles can only be achieved by understanding their behaviour in as many as possible engagement scenarios.

MANPADS operate by tracking a heat source on the aircraft. The operator, located on the ground, points the missile to the aircraft to enable the missile to track and intercept the aircraft by engaging on a heat source located on the aircraft, typically the engine of the aircraft. To protect an aircraft against these kinds of attacks, countermeasures are implemented on the aircraft. There are different types of countermeasures in use and one such type is infra-red flares. Flares are hot, burning metal with temperatures equal to or hotter than the engine of the aircraft, thereby acting as a decoy to misguide the missile. Figure 1.1 illustrates a military aircraft ejecting infra-red flares to misguide the approaching missile in an effort to protect the aircraft against the attack. The smoke from the burning flares is clearly visible in the scene.

In order to effectively design, develop and deploy the different countermeasures, computer-based simulation systems are used by the military. The development and simulation of computer models are important aspects of assessing the capabilities of

Figure 1.1: Apache Helicopter Releasing Flares [89]

the systems and are fundamental in the development of aircraft countermeasures. Simulation systems utilise computer science technology to model real-world objects and simulate them in an artificial world. One such system is the Optronic Scene Simulator (OSSIM) that is used to model and simulate several models in a military scene, such as that illustrated in Figure 1.1. The research described in this document involves the construction of an ontology to support a specific application of the OSSIM system.

Ontology as a technical term denotes an artefact that is designed and constructed for the specific purpose of modelling knowledge about a domain of interest. Ontologies and ontological engineering feature extensively in investigations to find possible technologies that support modelling within computing systems. One of the original definitions of the term ontology by Gruber defines an ontology as a formalisation of a shared conceptualisation [34]. A formal conceptualisation is the representation of the concepts in a domain in a formal language. Formal ontologies are therefore ontologies constructed using a formal representation language such as Description Logics (DL) [4]. Given the characteristics and purpose of ontologies, it was proposed that the use and application of an ontology to address the identified needs in a specific application of OSSIM are investigated.

The aim of the study was thus to determine, through a rigorous research process, if an ontology can be constructed that will play a supportive role in the environment wherein the simulations are being performed. That is what the benefit of its construction and use will be and whether it will provide a solution to issues in the

environment.

The following section provides more information on the simulation environment and the issues that lead to this research project.

1.1 Background: Countermeasure Evaluation

The Optronic Scene Simulator (OSSIM) is a set of software applications that is used to simulate scenes in the visual and infrared bands. Electro-optical systems involve components and devices which concern the interaction between the electromagnetic, the optical, and the electrical, or the electronic states of materials, and concentrating mainly on the infrared region of the electromagnetic spectrum. There is a specific application of OSSIM that employs specific libraries to undertake the radiometric, imaging, target and flare countermeasure simulation.

To better understand the research background, a short introduction into the OSSIM system is necessary. The OSSIM system is able to perform the following tasks in the domain of electro-optical sensor systems [97]:

- Assist in the design and development of optronic sensor systems that are part of countermeasure systems or thermal imaging cameras by providing performance evaluation functions.
- Assist in the preparation of military flight missions by running prepared flights in simulation. By running the flight tests in simulation, problem areas can be identified prior to real world testing, thus saving time by eliminating unnecessary tests.
- Provide increased understanding of the complexities of the real world domain as a result of experience with building models and testing systems against these complexities in the simulation.
- A good simulation reduces the dependence on extensive field tests that may involve expensive hardware. It provides simulation tests which are more flexible than field tests.

One of the applications in OSSIM is used to perform research on the protection of military aircraft against attacks from MANPADS. This self-protection application simulates what happens when a missile is fired at an aircraft. This scenario happens when an enemy missile is fired at a military aircraft flying at a certain speed, or hovering at a certain height in the case of a helicopter. The aircraft has a missile warning systems installed that will detect, warn and possibly trigger countermeasures installed on the aircraft. There are many events that influence the probability of the missile hitting or missing the target. The countermeasures might be successful

in stopping the missile from hitting the target by guiding it away from the aircraft. Another scenario may occur if the countermeasure does not have an effect on the missile and is therefore unsuccessful in deflecting the missile. Figure 1.2 shows a close-up view of a cargo aircraft ejecting infra-red flares as a countermeasure.

Figure 1.2: Cargo Aircraft Ejecting Countermeasure Flares [79]

The countermeasure simulation system simulates a missile attack, as described in the previous paragraph, by using software models of all the objects that are part of the simulation scene set up for the evaluation. Models of every component are build and implemented in the system. The aircraft, missile and countermeasures are presented by separate models in the system. Model parameters describe the characteristics of each model. By simulating their behaviour and interaction in a specific scenario, a prediction can be made as to the effectiveness of the countermeasure against a missile attack. The use of models and simulation minimises the necessity of field tests for every possible scenario. Field tests are very expensive and missiles cannot be fired at aircraft in tests. Computer-based simulation makes it possible to evaluate and predict much more but with less expense and more safety. This application of OSSIM is a software tool used mainly to evaluate the countermeasures deployed on military aircraft to protect them against missile attacks. Simulation makes it possible to experiment with actual systems, in a cost-effective way.

In summary, the application provides functionality to:

- Simulate the real-world behaviour of models such as aircraft and missiles.
- Simulate interaction between models as a result of certain events.

- Accelerate countermeasure research by making it possible to run various scenarios.
- Assist in planning field trials by eliminating unnecessary tests.
- Simulate what might not be possible or desirable in field trials.

The system has been in use for some time and has been successfully applied to undertake these kind of evaluation studies. There are, however, certain requirements for improving and enhancing the system. The next section highlights some of the requirements that led to the research study described in this document.

1.2 Problem Statement

OSSIM has been successfully applied in countermeasure evaluation studies to protect military aircraft. There are, however, improvements that can be made that will result in a better use of the system, as pointed out in the remainder of the section.

1.2.1 Need for a Common, Shared Language

Undertaking a simulation study is not a solitary effort; a group of people are involved and Carson [18] calls such a group the Simulation Team. People with different skills and expertise make up the team. The person developing and building new models or setting up a simulation is not necessarily the same person analysing and writing the resulting report. All team members need to work closely together to perform a successful evaluation study. Unfortunately there is little common technical language to describe specific concepts and often different terminology are used to describe the same concept. A need therefore exists to have a single source of definitions to describe each concept and a shared language for use by the members in the team. Such a shared resource can also be used to communicate the concepts in the system to people unfamiliar with the contents of the system and help with presenting realistic models.

1.2.2 Repository of Objects in the Simulation Environment

The simulation environment consists of a rich set of models and information about the models. Physical and behavioural properties are applied to describe these models. Every model in the simulation has several parameters that can be set to determine the behaviour of the model in the scene during a simulation run. These properties and parameters determine how the models behave in a specific scenario and how they interact with each other. The success of a simulation depends firstly on how good the models are. If the behaviour of the models is not implemented or used correctly, the simulation results will not be accurate or usable. New models are also constantly

being added to the simulation system by different clients. It is thus important that these models adhere to existing standards and rules. If a repository of models and parameters is available it can be used by clients who wish to develop their own models. The integration of new models into the simulation system will be easier if the developers of these new models have guidelines to adhere to. Furthermore, while setting up simulations, all the model parameters are not readily available. The simulation practitioner that is performing the simulation set-up might not have specialist knowledge of how the models interact, and could set up scenarios that are syntactically correct but do not make sense in the real world. In Chapter 5 examples are provided to elaborate such a scenario.

1.2.3 Description of Scenarios

Evaluation studies consists of many different possible scenarios and over the years a rich set of simulations has been built up. However, there is no repository of descriptions of previously run simulations. Such a repository would ideally provide a high-level description of a specific scenario in a consistent terminology.

1.2.4 Validation of Scenarios

Models are required to behave in a certain way to reflect their true behaviour. Although it might be possible to set up a simulation that will not produce errors at runtime, 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. There is, therefore, a need to validate how the models are specified in the scenario.

By meeting the above-mentioned requirements, the simulation environment will improve and these issues will be resolved. Thus, the problem statement is to find a tool that can provide a solution to address these requirements. The next section provides background to ontologies to present a picture as to why an ontology might offer a solution to meet the requirements as stated above.

1.3 Background: Ontologies

A background to ontologies could start at the coining of the term by ancient philosophers. Aristotle and early philosophers analysed objects in the world and studies existence [34]. These philosophers concerned themselves with the study of 'being' or 'existence' and used the term 'ontology' to describe concepts in the world.

Aristotle was one of the first philosophers to ask questions concerning 'To be..' and 'What is being?'. Kung [48] discusses the question Aristotle asked and the answer that he provided: all beings in the world must have some characteristic, and that

characteristic makes it something that exists in the world. Smith [87] explains that ontologies provide a classification of concepts. The classification should be definitive in the sense that it can serve as an answer to such questions as: What classes of entities are needed for a complete description and explanation of the domain of interest?

In practical terms, an ontology is both a controlled vocabulary of things in the real world as well as capturing the relations between them. It is more than a taxonomy because it captures relationships as well as the meaning of words. Ontologies use the terms concepts or classes to refer to concrete things, such as 'animal', or more abstract things, like 'program'. An example of a basic ontology is shown in Figure 1.3. This is an ontology of animals wherein the concepts in the domain are plants and animals, herbivores and carnivores, antelopes and lions. From the animal ontology, it is possible to derive the fact that a lion is an animal, even though this was not directly stated.

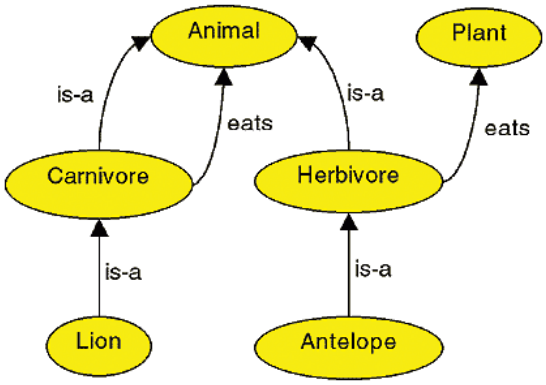

Figure 1.3: Animal Ontology [96]

The evolution of ontology technologies from philosophy to Computer Science (CS) was greatly influenced by the different fields of Artificial Intelligence, Software Engineering and database communities [74]. The demand to share knowledge and to have the same meaning for concepts, played a role in the use of ontologies in CS.

1.3.1 Ontologies in Computer Science

Ontologies have been widely used in CS since the '90s and Gruber [34] presents a definition that was adapted from the definitions in philosophy:

'An ontology is an explicit specification of a conceptualisation.'

The definition from Gruber states that an ontology is a description of the set of the concepts and relationships that can exist in a domain. Ontology thus formally rep-

resents knowledge of a domain as a set of concepts within that domain, and the relationships among those concepts. It is not only a description of the domain, it can also be used to reason about the concepts, deriving logical consequences from the information captured in the ontology.

Noy and McGuiness [62] describe the purpose of ontologies in CS to be:

- Sharing a common understanding of the structure of information among people or software agents.
- Reusing the domain knowledge among different applications.
- Publishing the domain assumptions explicitly.
- Separating domain knowledge from operational knowledge.
- Analysing the domain knowledge.

The function of sharing a common understanding is an especially important role of ontologies in CS. The World Wide Web created a certain expectation of how information is shared. This requirement to share information also impacted on the fields of Artificial Intelligence, Software Engineering and database development, as described by Sanchez *et al.* [74]. In order for the scientists in these fields to expand their capabilities, they need to represent their knowledge by having a model of a conceptualisation to present the concepts they use. An example of how an ontology is utilised to successfully share knowledge is the Gene Ontology [3] that plays a central role in making controlled vocabularies available for sharing biomedical information.

Such presentations of concepts are potentially beneficial for all fields of CS, especially modelling and simulation.

1.3.2 Ontologies in Simulation Systems

A specific application of ontologies in computing is in the applications that build models and use the models in simulation systems. A simulation system is a specific application that executes a model, represented by a computer program that gives information about the system being investigated [7].

According to Benjamin, Patki, and Mayer [8] ontologies can support the simulation modelling and analysis life cycle from as early as the design phase. They also play an important role in the integration of simulation parts and simulation composability. Porzel and Warden [68] sum up the use of ontologies in the different stages of modelling and simulation: naming the stages design, runtime and analysis. During **design**, ontologies can, according to Benjamin *et al.* [8], assist in providing detail analysis of the data and help in understanding problem descriptions. They can assist in defining consistent terminology to describe the domain. Ontologies provide

consistency checking to verify that the information captured does not contradict information in the domain. Completeness checking ensures that there are enough information captured so that the simulation model will be able to provide answers that are meaningful in the domain. During **runtime**, complex simulations can be made more accessible for interaction among humans, computers and agents because of the explicit and formal representation in the ontology. Finally during the **analysis** phase, ontologies can assist in interpretation of the simulation results and act as a source to understand the meaning of the results. The reasoning capabilities of ontologies can be used for validation of data.

1.3.3 Ontologies in the Military Environment

The military uses computer systems and simulation extensively to develop and deploy systems. In the book, On War, Clausewitz [19] writes about the need to develop situational awareness and understanding through the practice of mereology (the logical theory of part and whole): '...three quarters of the information upon which all actions in war are based on are lying in a fog of uncertainty....in war more than any other subject we must begin by looking at the nature of the whole; for here more than elsewhere the part and the whole must always be thought of together...'

Bowman and Lopez [15] discuss the complexity implied above further by stating that although the basic nature of war is constant, the means and methods of combat evolve through time. Military operations of the future will be conducted differently than they are today, and present operations employ means and methods that are radically different to those of the past. According to Clausewitz [19] there are three levels of war - tactical, operational, and strategic. Ontologies can be utilised in all three of these levels. An ontology can, for example, be used to capture and integrate human knowledge that military professionals use to solve problems.

Military systems and the knowledge therein are complex and often layered. According to Mandrick [54], ontologies applied in military information systems contain the following:

- Concepts to present the objects, properties, events and relations in military domains.
- A representation that reflects reality.
- A vocabulary that is a shared, common and consistent description of a given domain that can be used by military personnel.
- A shared resource that can be used for communication between different communities.

- A tool that can be used to support collaborative development and integration of information between different military applications.

The points made here illustrate that ontologies play a role in a wide spectrum of military applications. It varies from the simulation of a complete battlefield to a very specific study of only a part of a weapon. Confidence in the results of such simulations is often debated and their validity questioned. Ontologies can play a role in better understanding the models and systems and building confidence in the results.

The previous sections highlighted the early history of ontology in philosophy and the use of ontologies. The advantages that ontologies brought to the scientists in CS, simulation systems and military information systems led to the suggestion that an ontology might support the simulation system in performing successful evaluation studies.

1.4 Research Questions and Purpose of the Study

After considering ontologies and their advantages against the requirements for enhancements in the simulation environment, a suggestion was formulated that the construction and integration of ontology technologies into the simulation system could enhance the system in fulfilling the listed requirements.

To find out if this can be supported by evidence, the following research question was formulated: *How can ontology technologies be used to support a countermeasure simulation system environment?*

The above-stated question is broken down into three sub-questions in order to lead the research process to a stage where an answer to this question can be concluded. The following research sub-questions were formulated:

1. What are the requirements or concerns of a countermeasure simulation system environment that ontology technologies support?
2. How can an ontology be constructed that will capture the knowledge of a countermeasure simulation system environment?
3. Does the use of ontology technologies support and enhance the functions of the simulation system in the domain?

The broad statement made by the main research question will be supported by addressing these sub-questions following a research approach. The research approach, discussed in the following section, is initialised with a literature study covering ontologies and ontological engineering, ontologies in modelling and simulation and ontologies in the military.

1.5 Research Approach

This section describes the approach that was followed in conducting the research. The first step was to perform a literature study to determine the use of ontologies in CS, in simulation systems and by the military. The history of ontologies and the road of ontologies from the philosophers to CS are discussed.

The construction of the artefact, the ontology, must be carried out by following a research methodology. Design research was chosen as the research methodology. The steps in design research were followed from determining the scope of the ontology until the maintenance and use of the ontology. The research was concluded by evaluating the role of the ontology in the simulation environment and a discussion of the research contributions.

Figure 1.4 provides an overview of the research process that was followed.

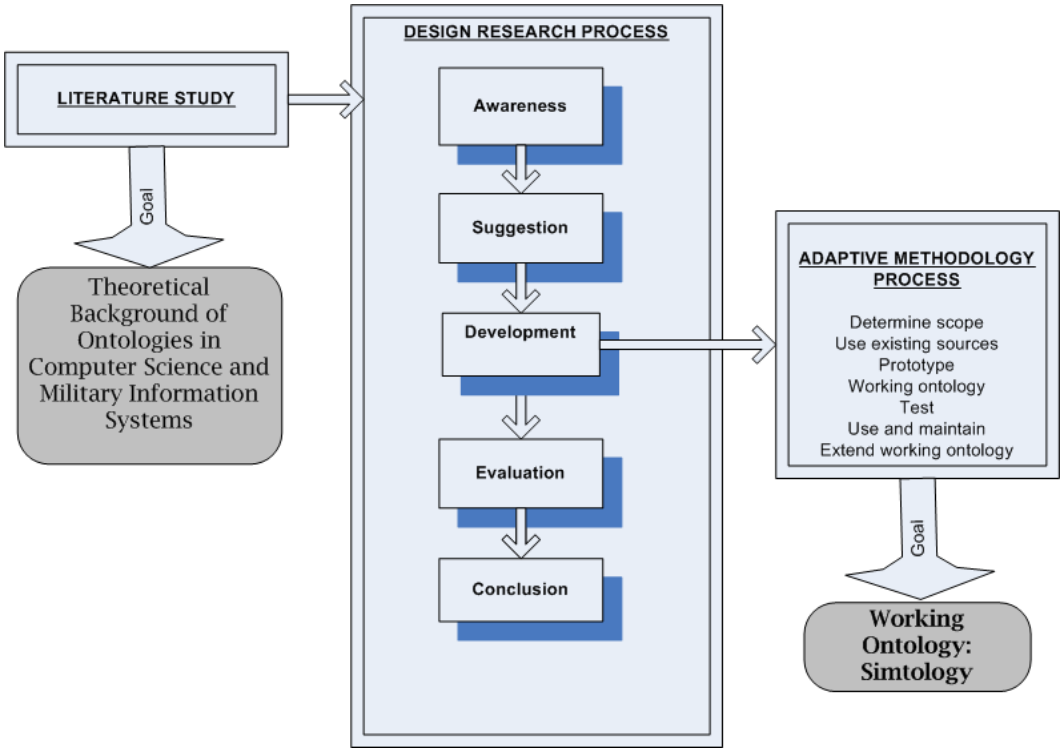

Figure 1.4: Overview of the Research Process

The next section gives an overview of the research methodology and design research which was followed in undertaking the research study.

1.5.1 Design Research

Design research is a research methodology applied to answer the research questions through the design and creation of innovative artefacts, thereby contributing new

knowledge to the body of scientific evidence [39].

Design research can be applied to a domain in different ways. The methodology that was going to be followed was described by Vaishnavi and Kuechler [92], who propose the following stages:

- Awareness of the problem: The problem at hand must be analysed to reach a proposal for research.
- Suggestion: Possible solutions are investigated.
- Development: Construct the artefact using a formal development methodology.
- Evaluation: Evaluate the artefact against the criteria set in the awareness stage.
- Conclusion: Concludes the research process by consolidating the results and the knowledge gained.

The design research approach is applied to the problem statement as stated in Section 1.2. An ontology is the artefact that is going to be constructed in the development phase of the design research process.

1.5.2 Constructing the Ontology

The goal of the development phase in the design research process is to construct an artefact. To ensure an artefact of quality, a formal development methodology must be followed. The adaptive methodology process, designed by Bergman [10], is an agile development methodology suitable to construct ontologies and is the development methodology adopted for this study.

The commencing step in adaptive methodology, as proposed by Bergman [10], is to determine the scope of the ontology by analysing the domain. Existing sources of information available in the domain are examined and inspected to utilise existing information. A prototype ontology is constructed to illustrate the concepts of ontologies. The prototype ontology is expanded to a working ontology by adding additional concepts and properties. After the construction of the working ontology, the ontology will be tested by using a set of criteria. The structure as well as the contents of the ontology are validated by ensuring the ontology is complete, consistent and a true reflection of the domain. The ontology is used and maintained to ensure that it is relevant for the domain. The final step is to extend the working ontology. Extending the ontology is an iterative process of improving the ontology, testing new additions and putting it to use. At every point of a development iteration, the ontology is tested and evaluated before use. This development process ensures that the artefact grows as scope and complexity grows.

The development stage is iterated until a satisfactory point is reached where the

artefact can be evaluated according to the criteria set in the initial stages of awareness and suggestion. The goal of evaluating the artefact is to decide if the development of the artefact provide a solution to the initial problem defined. The final stage in the design research process will be to conclude the research process by communicating and presenting the results, within the domain and scope of the research done.

1.6 Scope of the Research Study

The countermeasure simulation system is an application that uses libraries and models to undertake countermeasure simulation. The research described in this document is confined to concepts in the self-protection application only, even though the OSSIM system covers a wide range of applications. A specific set of models (aircraft, missiles and countermeasures) are used in the self-protection application. These models are client specific and only available to the specific part of the simulations, being of a sensitive nature.

The models in the core OSSIM system are available to all the applications of OSSIM and describe the environmental parameters of the scenario. These models are:

- Environmental models describing the atmosphere.
- Terrain models.
- Background models.
- A model representing a clock to determine the date, time and duration of the simulation.

The models in the self-protection application present the aircraft, countermeasure and the different types of missiles. The models are:

- Geometric and radiometric models of the aircraft, as well as models of the missile warning systems implemented on the aircraft.
- Models of the countermeasures such as the specific type of flare designed to protect against a specific type of missile.
- Missile models - the specific threats that are encountered by the specified aircraft.

The most important factor to take into account when determining the scope of the research was that the working ontology has to include a presentation of all the models that can possibly be used in a scenario of a countermeasure evaluation study.

A typical design research process involves feedback from the evaluation stage that led to more suggestions to be investigated. For the purpose of this research, only one development cycle was performed to limit the amount of work.

The main goal of the research is to support the self-protection application of OSSIM and if successful in that, the research can be expanded in future to include other applications of the OSSIM system.

1.7 Publications of the Research

During the course of this research project, two publications were published.

1. The first publication by Lombard [52] was conducted during the initial stages of the project and describes the initial study and the prototype ontology.
2. In a second publication, Lombard, Gerber, and van der Merwe [53] discuss the construction of the working ontology and especially the lessons learned during the construction of the ontology.

1.8 Chapters Overview

The document is organised as illustrated in Figure 1.5.

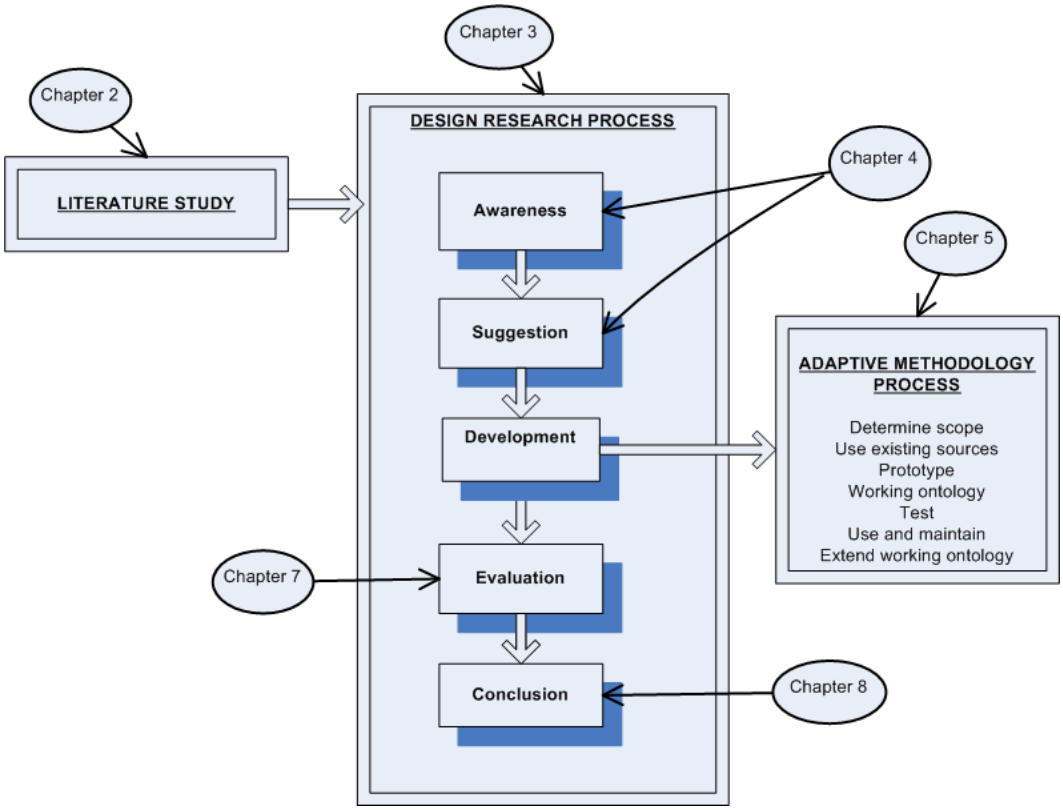

Figure 1.5: Chapter Map of the Research Process

Chapter 2 follows on the introductory chapter, Chapter 1, and contains the review of the literature on ontologies and the role of ontologies and ontological engineering. General ontologies are discussed and thereafter the role of ontologies

in Computer Science is discussed. Subsequent sections describe ontologies in different applications of the military. Examples of ontologies in military simulation are also included.

Chapter 3 describes the design research process that was chosen to perform the research. It explains design research and why it was chosen as the research methodology for constructing and evaluating an ontology although design research is not widely used. The stages of design research are listed and discussed.

Chapter 4 provides insight and background to the environment wherein the simulation system is used. The issues and shortcomings experienced by the people involved in the system are discussed, thus this is the awareness stage of design research. A solution to be developed and evaluated is suggested.

Chapter 5 covers the development process of the ontology. The third stage in the design research process is the development of the artefact. Adaptive methodology is followed as the developing methodology and all the steps involved in the construction, evaluation and use of the ontology are described in this chapter.

Chapter 6 addresses the evaluation of the ontology and presents the proof that an ontology can indeed support the simulation system in a positive way, addressing the issues in the awareness stage.

Chapter 7 discusses the finale stage of the design research process by elaborating on the research outcomes and contributions. In design research, this stage is completed when the development of the artefact reaches a stage where the artefact is in such a state that conclusions and contributions can be made from the research.

Chapter 8 concludes the document by summarising the content of the preceding chapters. An overview of the full process provides a summary of the research process as well as future recommendations.

CHAPTER 2

The Literature Review

2.1 Introduction

The literature study described in this chapter investigates the use of ontologies in Computer Science (CS) and particularly in systems in use by the military. The concepts of traditional ontologies and ontology engineering are also discussed.

The concept of an ontology is not something new but it is only as recently as 1998 [35] that the notion of ontologies were widely accepted in the field of CS. It is often the case that a computer scientist gets introduced to the term ontology only when encountering it in CS, without realising its origin in the field of philosophy.

Traditionally the term 'ontology' refers to its use in philosophy but recently the term is more commonplace in CS [100, 87, 94]. Ontologies in CS are built upon the principles of the ontology of the philosophers and those early ontologies laid the foundation for ontologies in CS. According to Zúñiga [100], the role of ontologies in CS evolved over a period of time. The different needs of the scientists in the fields of artificial intelligence, software engineering and the database community were the original drivers for ontologies to be introduced to the CS community. One of the subfields of CS is modelling and simulation, as extensively described in literature by Benjamin *et al.* [8], Fishwick and Miller [28] and Silver, Hassan, and Miller [80]. Modelling and simulation build a representation of the real world and simulate the behaviour of objects presented by the models. Ontologies are used in modelling and simulation during design and development of models and assist in the understanding of the models [8].

Ontologies are used in the military domain as a source of knowledge, as described by Schlenoff, Washington, and Barbera [77] and Valente, Holmes, and Alvidrez [93]. The main purpose is to have a knowledge source that can be shared between researchers in the military and the developers that build military applications. The simulation environment applicable to this research project is an example of modelling and simulation techniques applied in a military domain. There are several examples of ontologies used by the military in simulation systems [54, 49]. These ontologies in CS and in the military information systems are artefacts and need to be designed

and constructed using the same principles as artefacts in other CS fields.

The process of constructing an ontology is called ontology engineering [22] and comprises all the activities in the life-cycle of an ontology. Design principles are applied to all the steps from designing an ontology, constructing it and finally implementing it. Ontology editors are front-ends for ontology engineering that are used to create and manage an ontology. Ontology editors often use graphical environments wherein ontologies are constructed by using an appropriate ontology language. The editor used to construct the ontology in this research project is the Protégé 4 OWL editor [70], which is widely used to construct and edit ontologies in different domains [22, 77, 60].

Formal ontologies are represented in a specific formal knowledge representation language: the specification language that will be able to store domain information as a formal conceptualisation [4]. This means that domain information is presented in such a way that it can be processed by computer applications. The Web Ontology Language (OWL 2) by the W3C is one of the standardised ontology languages used for Web or computing ontologies [41] and is provided by the chosen ontology editor, Protégé. OWL 2 is based on description logics (DLs) as the representation language [4], and therefore there is reasoning support for these ontologies: information can be derived that was not explicitly stated in the knowledge base [64]. Furthermore, syntactically, OWL is based upon the Resource Description Framework (RDF) and XML, expanding RDF with the ability to add meaning to the concepts in the ontology [41].

The background study described in this chapter was conducted as shown in Figure 2.1 and the chapter is organised accordingly.

The course of the background study of ontologies is as follows:

- Firstly, the theory of ontologies are described with a discussion of the traditional ontologies of the philosophers.
- The second section elaborates on the use of ontologies in CS.
- The third section describes the use of ontology technologies in modelling and simulation systems in CS.
- The fourth section explains how ontologies are used as effective technology for various applications in the military domain.
- The fifth section concentrates specifically on military modelling and simulation systems, the domain applicable to the research described in this document.
- The chapter concludes with a description of what ontology engineering entails and some of the tools available to build and maintain ontologies.

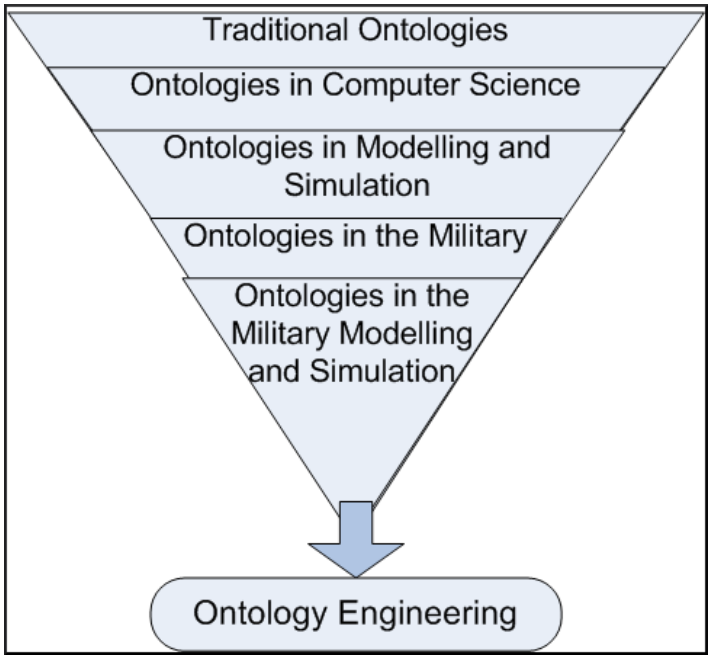

Figure 2.1: The Course of the Background Study of Ontologies

The section which follows looks at the traditional ontology of the philosophers in order to understand the role of ontologies in CS.

2.2 Traditional Ontology

From early on, philosophers studied the meaning and existence of objects in the world [87]. They argued about what exists and how objects that exist relate to each other. They reasoned about what there is and what it means. This study became to be known as the study of ontology [74, 67], based on the term “ontos” being the Greek word for “being”. Originally used in early Greece [36, 87], a form of ontology had been defined in the Western world as early as 1606 and by 1721 the word had been used and published in an English dictionary. Now, ontology is generally used to note the description and concepts of a domain and the “Ontology of music” or the “Ontology of history” can be read as “A description of the terms in music”or “What history is all about”.

Figure 2.2, adapted from terminology used by Poli [67], shows some of the terms describing ontology in philosophy. These different terms complement and overlap each other but present the intention of an ontology in philosophy.

Over the years there have been several philosophical debates around the term ontology and the meaning thereof. One such example is the famous debate between the two philosophers William Quine and Rudolph Carnap, described by Smith [87] and Sandholm [75]. William Quine in Quine [72] searches for the answer to the ques-

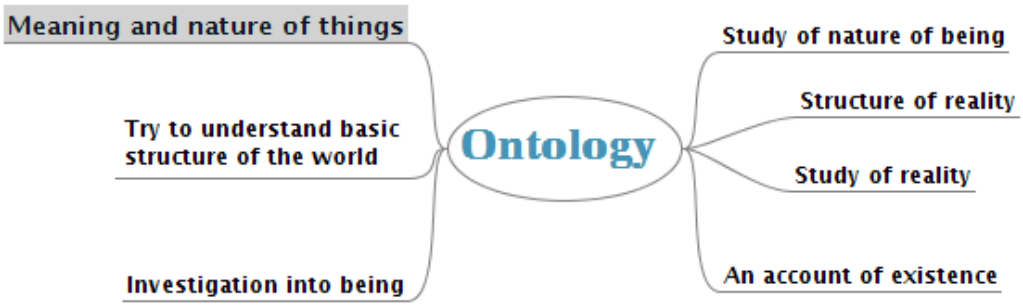

Figure 2.2: Traditional Ontology Terms

tion what exists by saying that "to be is to be a value of a bound variable", meaning that a description of what exists must arise from scientific theories of what exists. Carnap [17] on the other hand claims that ontology is more a function of how a specific language is the conceptualisation of the objects in a given domain. According to Sandholm [75], and that is why the debate is discussed, Carnap's view on ontology fits the view of ontologies in CS better. Although it seems that these philosophical debates are of no value to computer science and provide mere interesting reading, Sandholm [75] and Smith [87] emphasise that there are some lessons to be learned. Computer scientist might end up with a better conceptualisation, and thus a better ontology, if they use a more philosophical approach to building ontologies.

Ontology is not beyond criticism and misuse and both Santini [76] and Poli [67] argue for the careful use of ontologies. In a critical analysis of ontology, Santini [76] gives an overview of the use and abuse of ontologies. The accusation he makes is that the most widely used definition tells what an ontology is used for, but not exactly what it is. Although Santini [76] concentrates on the use of ontology in documents, his work does contain an important discussion about the use of ontologies to present meaning. He warns that care must be taken when deciding to use an ontology because it is not a single solution to all problems.

Although ontology is the study of what exists, there are some philosophers like Poli [67] that argue about the ontology of what does not exist. Poli [67] argues that to exist, and to be, is not the same. He mentions several cases to proof his point. He claims for example that past and future entities do not really exist and that entities can only be partially determined because some of its attributes might be hidden. Everything is thus not known, so the presentation of the object is not complete. In the context of this research project, a situation like that might occur in the simulation system when the exact behaviour of objects cannot be determined beforehand; for example the influence of environmental factors in the simulation.

The traditional ontology paved the way for the use of ontologies in CS today. The next section explores the use of ontology in CS in more detail by firstly looking at the period in history when ontologies moved from only being used by philosophers to being applied in a new application domain. Why did it happen and furthermore what role does it play in CS systems today?

2.3 Ontologies in Computer Science

As the different fields and domains of CS grew, new technologies brought new demands. Guizzardi [36] gives an account of how ontology was first mentioned in CS literature in 1967, and how the field of artificial intelligence (AI) began to use ontologies in its respective domains. The following three sections describe the road that ontologies followed to be used in CS, discuss the different definitions of ontologies in CS and compare ontologies with traditional software tools.

2.3.1 The Road Towards the Use of Ontologies in Computer Science

The World Wide Web and the information available for everyone that use it created a certain expectation of how information is shared [56]. This requirement also impacted on the fields of artificial intelligence (AI), software engineering and database development, as described by Sanchez *et al.* [74], Smith [87] and Sandholm [75]. There was an increased need to share and present knowledge.

In order for scientists in these fields to expand their capabilities, they needed to represent their knowledge through a model of conceptualisation that would present the concepts they use. The model of conceptualisation that presents the concepts in these fields, needs to be formally presented. This enables the automatic processing of the knowledge by computer systems [74]. For the AI systems, agents need to make decisions based on knowledge and a formal presentation enable the agents to use the knowledge in a way that is understood in a computational environment.

The next requirement was to have a high-level description of the concepts in the respective domains. In the field of Software Engineering, systems became more complex and object-oriented modelling and design began to be popular [86]. The software engineers need to have a high-level description of the concepts they are working with to be able to understand communication between objects. Sanchez *et al.* [74], furthermore, describe how the Database community also needed a high-level description to display objects in the database and Smith *et al.* [86] highlight that conceptual modelling is an important part of databases that often leads to problems of integration if not properly done.

The previous paragraphs explained how the AI engineer, the software developer and

the database modeller all have certain requirements to fulfil. These requirements can be summarised as:

- The need to represent knowledge;
- The need to share that knowledge; and,
- The requirement that the presentation, or model, must be formal.

These requirements can be seen as the main contributing factors that influenced the road towards the use of ontologies in CS. Sharing the same requirements, the different fields of AI, software engineering and databases each need a representation of knowledge through a logical structure that describes all the objects in their respective domains. This representation also needs to be formal. In CS, there is no debate about the existence of objects and every object in the domain can possibly be presented by a concept that capture its meaning. This is in contrast with philosophy, where the ontology is about "what exists" [94] in the real world.

The advantages of having shared knowledge in AI can be seen in the use of intelligent agents. The Active Framework, described by Guzzoni, Baur, and Cheyer [37], is a framework that utilises ontological technologies to develop an assistant that is able to schedule meetings by incorporating language interpretation, dialogue management and web services.

The Gene Ontology [3] is an example of successful knowledge sharing [88] and plays a central role in having controlled vocabularies available for shared use across different biological domains. The goal of the developers of the Gene Ontology is to produce a structured, precisely defined, common, controlled vocabulary for describing the roles of genes and gene products in any organism. The Gene Ontology brought considerable benefits to a range of different types of biological and biomedical research. For the database community, ontologies play a role in the semantic integration of databases by providing controlled vocabularies which have the goal of making it possible to search different databases secure in the knowledge that the same terms, wherever they are encountered, will always represent the same entities [88]. In another example, the British National Mapping Agency has large databases of geo-spatial information, for example survey data. Dolbear, Goodwin, Mizen, and Ritchie [21] describe how it develops ontologies to map to the information in the database.

In CS, an ontology is an artefact that is designed for a specific goal and the concepts confined to the domain. The next section looks at definitions of ontologies in CS.

2.3.2 Different Definitions of Ontology

There are several different definitions for the term ontology and this section looks at those used in CS and the differences between ontologies and the traditional tools used in CS in order to understand why ontologies add benefit to this field.

By the 1980's, ontologies were widely used in CS and Gruber [34] formulates the following definition for an ontology:

> *An ontology is an explicit specification of a conceptualisation.*

This original definition was expanded by Guarino [35] into a more formal definition. Guarino [35] aims to emphasise the difference between an ontology and a conceptualisation because he believes that an ontology is dependent on the language used, whilst a conceptualisation is not dependent on the language used. Guarino then defines ontology as follows:

> *An ontology is a logical theory accounting for the intended meaning of a formal vocabulary.*

Furthermore, Guarino [35] broadens the definition by stating that the formal vocabulary prescribe the logical language use, and the models are constrained by this prescription. An ontology reflects the commitment by the models to adhere to the logical theory.

Thus, Guarino [35] clarifies the use of the word ontology in CS systems by the following distinction: ontology in philosophy is a conceptualisation and ontology in CS is an ontology. This difference between ontology in CS and ontology in philosophy was further emphasised by Zúñiga [100]. He felt, however, that the two aforementioned definitions were not adequate and formulates the following definition for ontologies in CS:

> *An information science ontology is an axiomatic theory made explicit by means of a specific formal language.*

At the time Guarino [35] wrote the article, ontologies were not widely used but he already realised the possible advantages of ontologies in CS as follows:

- ***Databases***
  Ontologies can assist in the design phase to undertake requirements analysis and cooperate with the database when doing queries.
- ***User Interfaces***
  Ontologies can provide a view of the terminology used in the interface so that the terms can be better understood. It can also act as a repository for the fields in the interface or assist in mapping the terminology used in the interface to

the terminology used in the software.

- ***Application Programming***
  Software applications often consist of different sources of knowledge that might be hard-coded into the program or reside outside the program in external sources. Ontology can make software more transparent by capturing all the knowledge inside the ontology.

The reason for the definition of Gruber [34] being so widely used and the value of the discussion of Guarino [35] is the fact that an interdisciplinary audience is addressed, thereby reaching a wide spectrum of different users. Not only did Zúñiga [100] uses these two definitions, it appears in almost all writings about ontology and ontology in CS. Fensel [27], for example, elaborates on this very thoroughly by explaining how ontology fits into information systems. Fensel [27] also argues that although ontology promise to be a solution to knowledge presentation in electronic commerce, it is not yet the silver bullet that will solve all problems. According to Fensel [27], ontologies provide a way to exchange meaning, by requiring consensus and by reaching consensus as a result. Figure 2.3 by Fensel [27] sums up the difference between ontologies in philosophy and ontologies in CS.

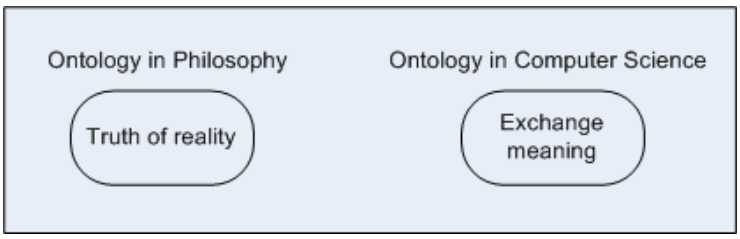

Figure 2.3: Fensel [27] on Ontology

But when can something be called an ontology? According to McGguinness [56], an ontology must have a set of minimum properties to be called an ontology. To simply make a list of terms is not an ontology because it lacks meaning. A list of terms with meaning is not an ontology because it is not machine readable. A true ontology must at least have an extensible vocabulary, a hierarchical structure of unambiguous interpretation of classes and relationships between the classes. The goal of ontologies in CS systems must be to make the domain knowledge useful in a computing environment.

The previous paragraphs explained the difference between ontologies in CS and ontologies in philosophy, as well as the criteria set by McGguinness [56] for an artefact to be an ontology. The next section compares ontologies with traditional software tools.

2.3.3 Ontologies and Traditional Software Tools

Several tools are available in CS to assist in the development of systems. Why is it then beneficial to use ontologies? What are the advantages of using ontologies?

Ontologies and traditional software engineering tools such as object models are different but should complement each other, as emphasised by Siricharoen [83] and Tran, Lewen, and Haase [90] and in a later paper, again by Siricharoen [82]. According to Siricharoen [83] and Siricharoen [82], it may appear that ontologies and objects fulfil the same role and there are indeed some overlapping qualities, but they do differ. Object models consists of collections of objects that have attributes. The same types of objects are grouped into classes. Classes can have methods and relationships between them. Ontologies describe the classes in a domain, and a specific object in a class will be an instance in the ontology. Ontologies are however found on logic [82], and because of that, use properties to reason (See Section 2.7.4). Therefore properties play a different and more important role than in object modelling. Relationships between classes are limit to class hierarchies but in ontology modelling they are allowed without restriction. Siricharoen [83] very importantly points out that ontologies should not replace procedural software technologies but must be use in conjunction with these technologies because this provides additional functionality. He provides three examples to illustrate his claim:

- Data mining: Ontologies can help to make data-mining more efficient by organising knowledge. The idea is not to replace the data-mining algorithms, but enhance functionality.
- Software Engineering: Ontologies can help to validate models by building meta-models of the models.
- Database Technology: Storing large-scale datasets in an ontology can provide a conceptual view of data sources thereby showing an integrated view without replicating datasets.

Siricharoen [83] also claims that the techniques used in object-oriented modelling can be adjusted and applied in constructing ontologies. The object model consists of objects with structural data (properties, relationships, events and processes) between objects and the ontology offers a controlled vocabulary of concepts. Both of them contain information of the objects in the problem domain. The ontology is a formal, explicit specification of a shared conceptualisation and can link with the object models in creating an enriched view of the related objects in the domain.

According to Tran *et al.* [90], ontologies do not have to adhere to all formal definitions in order to be useful. There has to be a balance between the amount of

knowledge presented on the one hand and the performance on the other hand. The decision on how much detail information to present in the ontology is determined by the application for which the ontology is created. Tran *et al.* [90] furthermore discuss the difference between ontologies and other formalisms. The conclusion is that ontologies will never replace any of these technologies but be used in conjunction with them. Table 2.1 gives a summary of the discussion of the difference between ontologies, XML schema, ontologies and database schema and ontologies versus object models.

Ontologies	**XML Schema**	**Database Schema**	**Object Models**
Focus on concepts		Focus on data	Focus on objects
Share information	Share data		
Meaning	Structure	No meaning	
Multiple viewpoints		Single view	
Behaviour not defined			Specify object behaviour

Table 2.1: Ontology and Traditional Software Tools

The purpose of discussing the difference between ontology and other traditional tools in CS is to explain what makes ontologies different, how each has its role, but that ontologies are best used in conjunction with the traditional software development tools.

Traditional software tools play an important role in the field of modelling and simulation. Modelling and simulation are traditionally known as complex systems. The following sections focusses on how ontologies are applied in the specific domain of modelling and simulation. The nature of the OSSIM system revolves around modelling and simulation and it is therefore important to concentrate on modelling and simulation systems in CS.

2.4 Ontologies for Modelling and Simulation

By 2004, a number of ontologies had been published for several domains [58]. By then the modelling and simulation community had begun to investigate the benefits that ontologies might have for modelling and simulation applications. The system used in this research and discussed in Section 4.1, simulates a real world scenario of a missile attacking an aircraft and the countermeasures that the aircraft deploys. Ontology can assist in the modelling process and make it easier to understand what to model and how to model it. According to Benjamin *et al.* [8], ontologies play a role in modelling and simulation at all levels of simulation. Not only during design but also at run time when simulation models are integrated and communication take place between the models and other systems.

2.4.1 Ontology at Simulation Design Time

Benjamin *et al.* [8] discuss the role and benefits of ontology during the design of the simulation models. During the design phase, the problem must be identified and the scope of the simulation determined. Each domain has its own terminology and the problem statement is determined by domain experts who is familiar with the terminology. Ontologies provide a way to interpret the technical terms and their usage. Ontologies assist as follows during the design function:

- Conceptual design - provides system descriptions.
- Data collection phase - provides an understanding of the data, ensuring correct data is collected.
- Detail model design - provides information of the model constraints.

2.4.2 Role of Ontologies during Construction of Simulation Models

Silver *et al.* [80] and Fishwick *et al.* [28] describe how ontologies are used during the simulation process to build simulation components from ontology repositories. Fishwick *et al.* [28] provide two examples, discussed in this section, of research done to illustrates the benefits of using ontologies during the simulation process.

2.4.2.1 A Model Ontology

This section describes the use of an ontology as a model repository to build simulation models. Silver *et al.* [80] present a method to use ontologies to develop simulation models and acknowledge the role that domain experts play in the capturing of domain knowledge in the ontology. Domain experts play an essential role in ensuring that the models are accurate representations of the real word. Once the knowledge is captured in the ontology, simulation models can be built from the ontology. Silver *et al.* [80] propose a method wherein a second ontology is created - a model ontology. The output model, built from the domain ontology, is translated by tools using the model ontology to get the simulation model ready for use in the simulation engine. The process followed is to first translate ontology instances to the mark-up language and then to generate simulation models from the mark-up language. The advantage of this approach is that several domain ontologies can be mapped to the model ontology. The implementation detail of the different ontologies does not need to be known; only the mapping tools have to be updated. By having the ontologies in an appropriate language, repositories of models can be built and re-used.

2.4.2.2 Manageable Components

Benjamin *et al.* [8] stated that one of the strengths of simulation models is that it can be divided into smaller, more manageable models. These models have to communicate and share data between them to run a successful simulation. There are may challenges involved in the integration of different simulation models and components. The models normally do not publish their intended meaning and the semantic content of the concepts that are presented. A term or concept used in one model might have a different meaning in another model. The challenge is to make a system aware of the presence of these problems. One solution to this is to use a processing language to map between the ontology of the models and the ontology of the system. Often these different components were not initially developed to work together. Different systems have different purposes and put different emphasis on certain aspects. By using an ontological approach, some of these challenges can be overcome. Benjamin *et al.* [8] propose that one should firstly create an ontology for each component, put all the components and their ontologies into a repository and allow access to that repository by modelling and simulation builders. The domain and goal of two simulation components might be the same, but they are defined on different levels of detail. One component is possibly defined on a high level of detail, modelling small components, but another model might concentrate on a low level of detail. A user selecting a component has the difficulty of choosing the correct model but by having access to information, such as the level of detail and the intended use a better choice of model can be made.

The first example by Fishwick *et al.* [28] describe how a domain ontology is updated from three-dimensional models created in a graphical interface. Researchers at the University of Florida built a framework that links the ontology with the simulation building tools. The framework consist of a graphical display that enable a user to build a simulation scene by selecting models from the display. The ontology is updated by adding the created scene so it acts as a knowledge base to hold information of the simulation scene.

Fishwick *et al.* [28] describe a second example of how a team at the University of Georgia built a prototype to investigate the use of ontology in discrete event modelling. Several ontology languages were investigated and the outcome was that modelling and simulation systems will have to adopt a combination of OWL and the Semantic Web Rule Language (SWRL). Modelling and simulation systems use terms from mathematics and statistics and to redefine those in an ontology will be a redundant effort. The use of higher level ontologies will allow the use of already defined concepts. They build a taxonomy for the ontology, deciding on the appropriate scope

that will be covered by using existing XML definitions.

Although the discussion of Fishwick *et al.* [28] does not go into much detail, it gives a good overview of what is possible and the research that has been done in the field. They also raised important issues regarding ontologies in simulation and asks if there are ways that different ontologies can be related to each other automatically. For example, if one ontology defines the word Atmo and another ontology uses the word Atmosphere, will it be possible to relate that to the same concept? The conclusion was that the modelling and simulation communities add their own flavour to ontologies to make them applicable to a specific domain.

In conclusion, Eriksson, Morin, Ekberg, Jenvald, and Timpka [26] illustrate a practical example of the role that an ontology plays as part of a simulation system architecture that separates modelling and execution, thus provide flexibility and run-time efficiency. Figure 2.4 illustrates the separation. The ontology editor was extended by extra functionality that enables construction of a scenario by using ontology models. The simulation parameters are thus defined on a high level. The second function that was added manages simulation runs. The simulation runs can be configured to use the scenarios already defined in the ontology, creating simulation jobs that are also part of the defined classes in the ontology. By separating the scenarios from the simulation jobs, the management of scenarios are more flexible. The work described by Eriksson *et al.* [26] is of particular importance for this research project and the construction of an ontology for the simulation system environment.

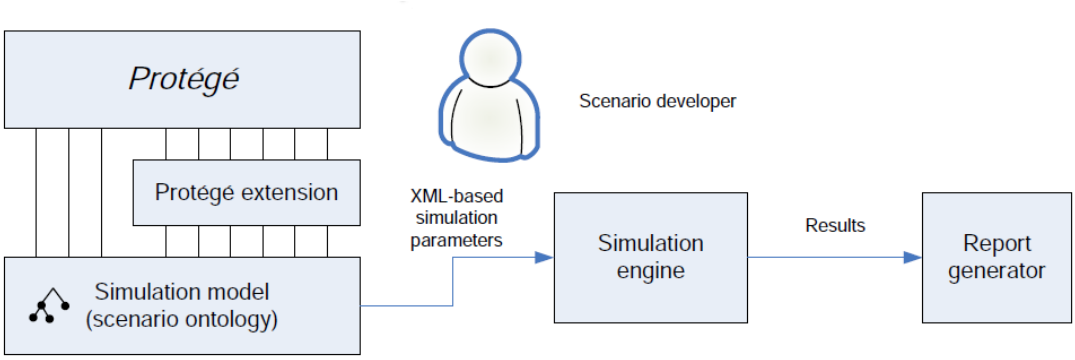

Figure 2.4: Role of Ontology in Simulation Architecture [26]

The main advantage of ontology in modelling and simulation is the ability to integrate different models. Ontology enhance the benefit of simulation composibility and enable added benefits such as saving time when developing models. The next section discusses how the military utilises ontologies in the military community and its applications.

2.5 Ontologies in the Military

The military community and the systems used by the military cover a wide spectrum of uses. Two main areas are command and control in the battlefield and the management of assets [77, 93]. Military practitioners are often on the forefront of the newest technology but the systems are complicated by the necessity to integrate legacy systems with these new developments [98, 84].

To support this view of complexity Clausewitz [19] in his book, *On War* writes about military information as follows:

> *...three quarters of the information upon which all actions in War are based on are lying in a fog of uncertainty... and,*
> *...in war more than any other subject we must begin by looking at the nature of the whole; for here more than elsewhere the part and the whole must always be thought of together...*

Furthermore, Mandrick discusses the use of ontologies to model information in the military environment. According to Mandrick, ontologies in the military must adhere to the same requirements as ontologies in other domains, as described later in Section 4.3.2. Important aspects to highlight is the ability of the ontology to provide a common vocabulary between planners, operators and commanders in the different military communities [54].

The following paragraphs describe how the use of an ontology can be beneficial in the military domain. The discussion covers three main areas:

- Sharing of knowledge.
- Integration of different data sources.
- Communication between military and civil organisations.

2.5.1 Sharing of Knowledge

There are several advantages to having military information in a shared knowledge source, as stated by Schlenoff *et al.* [77] and Valente *et al.* [93]. The main goal is to have a knowledge source that can be shared between domain experts in the military and the developers that must build applications.

One of the important research areas in the military focusses on how to build intelligence into combat vehicles in order to make them run more autonomously. To accomplish this, the operation of these vehicles and the environment where they operate must be fully understood. Schlenoff *et al.* [77] describe the process followed to create an ontology, called the Standard Intelligence System Ontology, that captures

knowledge about the behaviour of combat vehicles.

Another development is that of the Military Information Ontology (MilInfo), as described by Valente *et al.* [93]. An ontology (MilInfo) has been developed in the airspace systems domain. The purpose of the MilInfo ontology is to serve as a base ontology for the military information used by the system architects in the UN Department of Defence. In developing MilInfo, the developers investigated a number of top-level ontologies such as OpenCyc and SUMO and decided to use the ideas in OpenCyc ontology and apply these to the development of MilInfo [54]. The most important aspect they adopted from these sources is what they called 'abstract information' versus an 'information bearing object'. Information plays a double role: as content and as the bearer, or carrier, of the content. As a carrier of content, an 'information bearing object' is for example an XML file, and the 'abstract information' is the XML element inside the XML file. This was taken further by defining a concept called MilitaryInformation and another concept named MilitaryInformationBearingObject. The following requirements were drawn up by the developers of MilInfo to specify the extent of every piece of military information in the ontology:

- Description of the content.
- Users of the information.
- Different components.
- Source of the information.
- Quality of the information.
- Analysis and inspection information.
- Access restrictions.

The military generates a lot of data and MilInfo can assist in organising that data into usable sources of information. The future of MilInfo is to integrate into other ontologies that have been developed, namely Military Communications and Military Organisations.

The final discussion of military knowledge sharing is the work of Preece, Gomez, Mel, Vasconcelos, Sleeman, Colley, and Porta [69] that describe the development of an ontology as a solution to the challenges faced when deploying sensors for a military mission. Sensors are military intelligent resources comprised of intelligence, surveillance and reconnaissance used to detect enemies or guide missiles. One of the goals of sensors is to gather information during a military mission. The International Technology Alliance undertook a project to ensure that these sensors are deployed in the best possible way to support military missions. The ontologies in this domain can be used to:

1. Assist in identifying the requirements specification of a mission.
2. Specify what the ISR resources provide.

It is possible to solve these requirements with other technologies but the motivation to use an ontology to satisfy the requirements is because of the reasoning capability of ontologies, further described in Section 2.7.4. It is possible to specify the requirements or rules for a mission and subsequently use the ontology to deduct suitable sensors.

2.5.2 Integrating Different Data Sources

Military information systems use many different sources of information. Legacy systems are such a source that must be integrated with new technology, as pointed out by Bailey *et al.* [5] and Holmes and Stocking [42]. Thus one of the main activities of military information systems is to integrate different data sources. Bailey *et al.* [5] describe how an ontology was used to solve the problem of different naming conventions arising from the integration of different sources. The UK Ministry of Defence launched a project to investigate the use of four-dimensional ontologies to address the problem of multiple naming conventions. Various sources of country codes (NATO, UK Government Taxonomy and CIA World Factbook) must be integrated. An extensional ontology was developed using the Business Object Reference Ontology (BORO) Methodology [5], which concentrates on the semantics of concepts. The concept of space was added to the ontology by means of location data. Bailey *et al.* [5] highlight the fact that even with experts involved one might not end up with the same ontology. Semantics mean the real entities, not their representations. The different uses of the word semantics lead to problems when ontologies must operate together. The work done was criticised for not being adequate to demonstrate the usefulness of an ontology, but the possibility to extend the ontology did indeed make it a worthwhile effort.

2.5.3 Military and Civil Communication and Integration

The military is not always at war and often peacekeeping and humanitarian missions are undertaken. The research project discussed in this dissertation is applied to such an area. These missions are almost always undertaken together with non-military organisations and effective communication is a very important part of the success of these operations, as described by Winklerova [98] and Smart *et al.* [84].

Command and control operations in the military are defined by Education and Doctrine Division [25] as the processes of directing and controlling forces, and rely heavily on accurate information being available when needed. The information used for

command an control operations is a fusion between different systems and agents are used to communicate between these systems. The Message Content Ontology, described in Winklerova [98] is used to provide definitions in a domain. These definitions are made available to the different agents. The ontology has an additional role in acting as background knowledge for each agent. Traffic between agents can therefore be reduced.

AktiveSA, found in Smart *et al.* [84], is a technical system to demonstrate scenarios where military and civilian information is used together to assist in relief efforts. Information about geographical, transport and military entities are captured in different ontologies and made available on a web interface. There is functionality available on the web interface that allows for graphical views of the scenarios built, thus allowing users to build various scenarios using the information that the different ontologies have provided.

From the discussion in the previous sections it can be concluded that ontologies play various roles in the military domain. The military community however also uses modelling and simulation to enhance its operations and the next section therefore describes applications of ontologies used in military modelling and simulation applications.

2.6 Ontologies for Modelling and Simulation in the Military Environment

Modelling and simulation in the military environment are not about war games but play a role in a wide spectrum of military applications. This can vary from the simulation of a complete battlefield to a very specific study of only part of a weapon. Confidence in the results of such simulations is often debated and the validity of results questioned. Ontologies can play a role in understanding the models and systems better and building confidence in the results. The following sections describe the use of ontologies in military simulation in the following domains:

- Trajectory simulations.
- Distributed simulations.
- Integrating different data sources.
- Military and civil integration.
- Mobile route planning.
- Military software agents.

2.6.1 Ontology for Trajectory Simulation

Trajectory simulations are used in the military to calculate, for instance, the flight paths of a weapon ([23], [24]). The purpose of simulating trajectories is to assist in the testing and the designing of weapon systems. The simulations themselves cover a wide spectrum of simulations, from simple to more complex simulations.

Durak *et al.* [24] explain how an ontology specifically designed for this purpose, the Trajectory Simulation Ontology (TSONT) [23], was implemented and used. The purpose of this ontology is to provide a solution to problems experienced in the simulations. Users have different requirements and every new simulation scenario was designed from the start, no previous simulations were re-used. It takes a lot of expert input to prepare a simulation and even then the quality of each study must be verified, which takes a certain amount of effort.

The ontology was built from an existing object-oriented framework used for simulations. The framework was expanded by building onto it and implementing it into Matlab Simulink [55]. A special conversion tool was developed to convert models in TSONT to Matlab Simulation blocks. DAVE-ML is used to implement the aerospace models. All functions in TSONT are implemented.

Matlab Simulink has built-in functionality that can be used to build models by using a scripting language. The script used to build the models reads information from the ontology to build the models. The ultimate goal is to have a methodology that will produce Matlab Simulink models from domain models.

Durak *et al.* [24] also describe the use of top-level entities. The top-level entities in TSONT were chosen to match the Suggested Upper Merged Ontology (SUMO). A group of scientists from computing, philosophy and engineering created SUMO as a base ontology to be used. It is a large ontology that is defined in first-order logic which means it is independent of the language used [61].

2.6.2 Ontologies in a Distributed Simulation Environment

Legacy systems exist in all domains and will be continued to be used for a long time. The value of using ontology in simulation systems often lies with the ability it provides to integrate and exchange information with other systems, which are often these legacy systems. There are difficulties when different systems must integrate and Benjamin and Akella [9] describe how ontologies are used to address some of these difficulties.

Each system implements the meaning of objects in a different way by using ontologies. Benjamin *et al.* [9] name and describe two main areas in describing the com-

plexity of integrating such sources: semantic inaccessibility and logical disconnectedness. These two terms refer to, firstly, inadequate knowledge of what exactly the content in the system means and secondly, inadequate knowledge of the constraints that exists when sharing information between systems. Can these problems be addressed by using ontologies? To answer these questions, Benjamin *et al.* [9] divide the operation between systems on two levels, namely during design time and during run-time. Inter-operation at design time is achieved by considering the systems during the life-cycle of development. Run-time inter-operation is not so predictable and systems that are already developed must be inter-operational. Several strategic steps were defined to ensure that systems can communicate with each other during a simulation run:.

1. Extract the ontology for each simulation.
2. Refine information of ontology by human intervention.
3. Map ontologies.
4. Determine the information of each system that must be part of the communication.
5. Set up the communication channels.

Benjamin *et al.* [9] conclude with an example of a system containing three different combat simulation systems exchanging information among each other. The first simulation simulates a training pilot with cock-pit display and a view of the surrounding environment. The second system is a battlefield management system while the third system simulates radar detection of aircraft. The simulation systems simulated engagements such as firing munitions and destroying aircraft as well as a view of the battlefield. An ontology was created for each of these systems. During simulation, the three systems communicate through the use of a gateway and trace files. In order to achieve effective communication, the ontologies were used to match entities and information among them. Benjamin *et al.* [9] successfully demonstrate how, by using the activities of ontology extraction and elicitation and information mapping, different simulation systems can operate in one environment.

2.6.3 Mobile Route Planning

Nagle *et al.* [60] report on the development of a prototype simulation system that creates an operational view of a battlefield to assist commanders in planning a safe mobile route during military operations. Several research efforts and technologies were brought together to create a successful system. In a simulation system, a scenario is built for a unit and the knowledge in the ontology used to conduct route

planning and possible events on the route. The ontologies are connected to the software to provide knowledge of possible events.

Nagle *et al.* [60] note several lessons learned while developing the ontology for mobile route planning. It is not a trivial exercise to create a useful ontology that will provide all the benefits hoped for. It is often best to start with a focused case of concepts and the refine the ontology at each iteration. Several iterations are often necessary to ensure a meaningful ontology. Although this ontology was relatively small, the complexity of the system was difficult to capture. Human judgements had to be implemented. The application could have been written in procedural code but the advantages of using an ontology was the ability to bring in the meaning of events and the data. As the ontology is expanded, more knowledge is captured and the human decision-making in these situations is improved. The developers used a bottom-up approach in developing the ontology, finding that by using that methodology, the necessary technical detail was better captured when starting at a lower level of detail. The developers of this ontology managed to put the intelligence of human decision-making outside the code of the simulation and thus decrease the risk of changing existing functionality.

2.6.4 Military Software Agents and Ontologies

The use of software agents in software systems is growing. The benefits of agents in modelling and simulation lie in their ability to take over the role of a complex part of the system, such as the human cognitive process. The following section describes the research done by Holmes *et al.* [42] and explains how an intelligent agent make use of an ontology to fulfil its role in the simulation of autonomous military vehicles (UAV). The application consists of three different components namely: intelligent agents, the ontologies and the simulation system.

JACK intelligent agents, from the company Agent Orientated Software, are designed to run autonomously [42]. The purpose of the agents are to achieve some goal by interpreting the environment and then act upon it.

The Ilium framework, described by Holmes *et al.* [42], is a set of ontologies that supports the operation of aeronautical analysis and simulations. The framework consists of three layers of ontology. A foundation layer on top, a middle layer for general ontologies and a third layer for the specific domain ontologies of the military environment. An example of an ontology on the third layer is the MilAsset ontology that models military platforms and command and control assets.

The simulation application creates an *Operational Scenario* from the ontologies in the framework. According to the description of this framework by Holmes *et al.* [42], this

scenario is an ontology and consists of components from many different ontologies. Reasoning can be used to make sure the ontology is consistent. The ontology and the JACK agents work together to simulate UVA behaviour. The agents act on the operation of the military aircraft in the simulation, starting with initial information about the simulated objects and retrieving information from the *Operational Scenario*. The agents get initial information from the ontology, but adapt that information according to its own rules.

Holmes *et al.* [42] describe how an ontology was integrated to work together with intelligent agents in a simulation system. The ontology created for the application plays a vital role in integrating legacy systems with the specific knowledge of the agents in the simulation of UAVs.

The previous sections describe applications where ontologies are applied in modelling and simulation in a military environment. These ontologies are created and maintained by utilising software tools. The use and availability of these tools are described next.

2.7 Ontology Engineering

Ontology engineering is described by Dragan *et al.* [22] and Gómez-Pérez, Fernández-López, and Corcho [33] as *'the activities that involve the life-cycle of an ontology'*. This life-cycle runs from the initial design phase, through implementation and evaluation, up to maintenance and use. During all the phases design principles, procedures and methodologies are applied.

Although ontological engineering shares properties with other design methodologies, there are also differences. To illustrate the differences, ontologies can be compared to object models and object-oriented analysis and design. Just as ontologies differ from object models, so ontological engineering differs from object-oriented analysis and design. Devedzic [20] gives an understanding of ontological engineering and highlights the tasks that an ontological engineer can apply that are traditionally used by other disciplines. The question was asked of whether ontological engineering can use the same methods applied by object engineering and the conclusion was that the steps taken by an ontological engineer to build and manage an ontology is not so different from traditional software engineering. The following practices in traditional software engineering can be applied in ontological engineering:

- Modelling and meta-modelling.
- Software architectures.
- Programming languages and compilers.

- Traditional software engineering.
- Object-oriented analysis and design.
- Design patterns.
- Component-based software engineering.

Gómez-Pérez *et al.* [33] describe a few of the ontology development methodologies in use by ontological engineers by 2004.

1. **The Cyc method**: A hand-built knowledge-base containing 'common-sense' knowledge about the world.
2. **The Uschold and King's method**: Follow steps to identify purpose and scope, build the ontology, and capture the ontology. Uses one of three strategies to identify concepts (bottom-up, top-down or middle-out), coding and integration, evaluation and documentation.
3. **The Gruninger and Fox's methodology**: A formal approach that uses first order logic.
4. **The KACTUS approach**: Based on application development. An ontology is refined every time a new application is build.
5. **MENTHODOLOGY**.
6. **The SENSUS method**.
7. **The On-To-Knowledge methodology**.

An ontology in CS is an artefact and not an abstract object. It is, therefore, necessary to have languages and tools available to build ontologies. Several ontology tools exist and they differ in the way information is captured, extracted or visualised. The next section describes these tools, called ontology editors.

2.7.1 Editors for Ontologies

Ontology editors are software applications that enable the construction, editing and maintenance of ontologies. These editors provide support through the life cycle of the ontology [45].

Alatrish [1] and Kapoor and Sharma [47] describe some of the ontology editors currently in use:

- **Apollo**: Open-source, user-friendly. Not extensible.
- **OntoStudio**: Licensed. Client-server based.
- **Swoop**: Open-source, lightweight. Not extensible, demand knowledge of specific language.
- **TopBraid Composer Free Edition**: Enterprise-class modelling environment.

Free version only for small ontologies.

- **Protégé**: Open-source, easy to use interface, flexible. Large number of third-party plug-ins that extends the functionality.

An editor must be chosen to suit a project according to the needs of the specific project. Protégé was chosen to create the ontology for this research project. It is widely used, as described in the papers by Dragan *et al.* [22], Schlenoff *et al.* [77] and Nagle *et al.* [60], making it one of the leading editors for ontologies. According to Kapoor *et al.* [47], the strength of Protégé lies in the support it provides for tool builders, knowledge engineers and domain specialists.

Protégé is an open-source, stand-alone program that runs on a variety of platforms [70]. It provides functionality to load, create, save and import ontologies. The editor consists of a GUI with different panels. Figure 2.5 depicts an example of an open ontology with the different panels in the GUI. There are different tabs on the panel for accessing the different elements in the ontology.

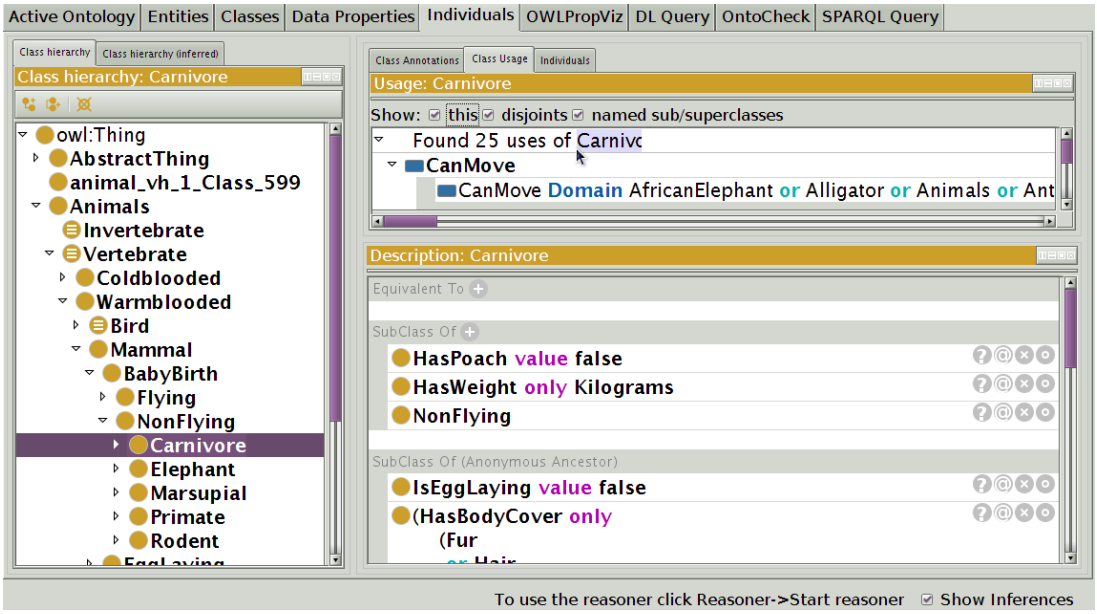

Figure 2.5: The Protégé Editor

It is possible to expand the functionality of Protégé by developing plug-ins, as illustrated by Eriksson *et al.* [26]. The work described by Eriksson *et al.* [26] demonstrate that Protégé can be extended by adding functionality for example a graphical editor. There are many plug-ins already developed and a list of these plug-ins are available from Protege [70].

2.7.2 Ontology Languages

The information contained in ontologies is presented using a formal language, the specification language which gives meaning to concepts. The Protégé ontology editor provides the use of the OWL 2 Web Ontology Language, or OWL 2 [41].

An ontology language must store domain information as a formal conceptualisation. It provides a way to present information so that it can be processed by computer applications. In order for applications and agents to share and exchange information, a specific syntax is used to present concepts and properties. OWL 2 mainly uses RDF/XML, the Resource Description Framework (RDF). RDF provides structure to identified objects. RDF provides three elements to describe objects: resources; properties; and, classes. Hence, a statement in RDF will be:

```
<scenario> <hasTarget> <aircraft>
```

Although OWL 2 uses the basic elements of RDF, it expands RDF with the ability to add meaning to the concepts in the ontology, therefore adding the ability to express and use first order logic. The following statement is an example of expressing the logic:

```
All aircraft have wings and Puma is an aircraft therefore Puma
has wings.
```

OWL 2 was developed from its predecessor, OWL, and provides for the following components to be defined:

- Concepts in the domain can be defined as classes, e.g. aircraft.
- Data properties can be defined to describe the classes, e.g. name of aircraft.
- Relations between the classes can be defined, e.g. aircraft has a countermeasure.
- An individual or instance of a class, e.g. Puma is a type of aircraft.

For the purpose of this document, the following notation will be used for classes, individuals and properties:

- Classes: **`Terrain`**.
- Instances/Individuals: *Forest*.
- Properties/Relations: `Name`.

The OWL 2 profiles are based on various Description Logics (OWL Working Group [65]) and the knowledge base is made up of two statements named the ABox and the TBox. The TBox describes the ontology in the form of concepts and roles, while the ABox contains assertions about individuals using the terms from the ontology. A typical TBox statement will be: Scenario hasTarget Target and a typical ABox statement

will be: ScenarioPuma hasTarget Puma, and together they make up the knowledge base.

The next section explores the different components and how they are used to describe the concepts in the domain.

2.7.3 The Components of an Ontology

The basic components of an OWL 2 ontology are classes, properties, individuals of classes, and relationships among these individuals [65]. Classes are the concepts in a domain and describe naturally occurring sets of things in the domain. Instances are of a specific class, 'instance of a class', called individuals. Properties link the concepts to each other and describe the roles in description logics and relations in UML and other object oriented notions.

- **Classes**: Classes, also known as concepts, are the main building blocks of an ontology and a class is defined as a set of individuals. There are certain requirements for membership of the class. For example, the class Aircraft would contain all the individuals that are of type aircraft in the simulation system. Classes may be organised into a superclass-subclass hierarchy, which is also known as a taxonomy. Subclasses are subsumed by their superclasses. For example, consider the classes **`Moving`** and **`Aircraft`**. Aircraft is a subclass of Moving (so Moving is the superclass of Aircraft). All aircraft are of type Moving. Being an Aircraft implies that you are also of type Moving.
- **Individuals**: An individual, also known as an instance, is a specific member of a class. An individual presents the objects in the domain. For the simulation environment, individuals will be defined for specific aircraft or specific terrain data.

  Figure 2.6 illustrates some examples of individuals in the domain.

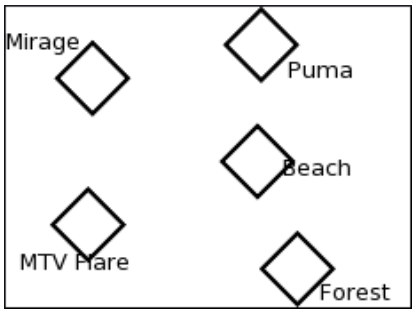

Figure 2.6: Examples of Individuals

- **Properties**: Ontologies use properties, also known as roles, to relate different individuals to each other. There are three types of properties in OWL 2 as

follows.

- **Object Properties**: Relationship between two individuals, for example Scenario hasTerrain SomeTerrain
- **Data Properties**: These properties describe individuals by parameters with data assigned to it. *Puma* `XPosition` 2000: gives the value 2000 to the data property `XPosition` of the individual *Puma*.
- **Annotation Properties**: Add additional or descriptive information to classes, individuals or properties. All classes have an annotation property called `Comment`, to describe the class.

Figure 2.7 illustrates examples of properties in the domain.

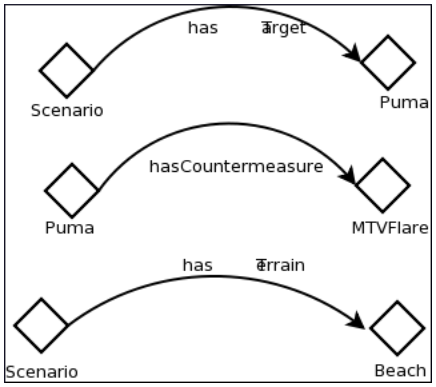

Figure 2.7: Examples of Object Properties

Two important properties in ontology engineering are the is_a and the part_of object properties. Object properties relate defined concepts but Smith [85] highlights an issue with the term 'concept' and how the definition of a concept can influence the properties that can be defined for that concept. The following paragraph illustrates his argument by using an example from the countermeasure simulation system.

Smith [85] claims that classes are normally a product of the human mind but classes defined in an ontology are based on reality to reflect the real truth in the domain. The traditional is_a and part_of properties in an ontology, cannot always reflect reality. The traditional definition stated that: for an is_a property, for example where *missile* is_a *weapon*, every member of the set *missile* must be a subset of a *weapon*. This is however not always true. In reality, a missile can be a target for a type of countermeasure deployed to destroy the advancing missile. In a case like that, the traditional definition of is_a property cannot present the scientific reality of a missile as a weapon and as a possible target. To solve this, Smith [85] proposes an improved definition which states that a *missile* is_a *weapon* if a *missile* and a *weapon* are universals, and for all times, t, if a *missile* is instantiated, a *weapon* is also instantiated.

The same principal can be applied to part_of relation.

Smith [87] warns that careful consideration must be taken when defining classes and the properties that relate to them. Defining a class incorrectly, or defining properties that are not a true reflection of relations in the domain can have an impact in how useful the final ontology will be for the domain for which it was created.

2.7.4 Ontology Tools: Reasoners

Because an ontology contains logic assertions, it is possible to apply logic reasoners to the ontology to computer inferences or logical entailments. This is a strength of ontology. Ontology editors have reasoners built into them, making it possible to test the ontology as it is built. A reasoner can be defined as:

> *A software application designed to examine sets of OWL statements and draw inferences from them... [6].*

The Protégé editing environment bundles or includes several reasoners as plug-ins with the software. The purpose of the reasoner is to derive additional information about the concepts defined in the ontology, check the consistency of ontologies, check for unsatisfiable classes and to see if axioms are entailed by an ontology. According to [64], the reasoner can perform the following functions:

- Consistency of class description - catch design errors, for example if an aircraft position and speed is set to hover but that type of aircraft cannot hover.
- Satisfiability of a class - determine whether a description of the class is not contradictory and that an individual can exist for the class.
- Subsumption of classes - determine whether class C subsumes class D, meaning that the description of C is more general than the description of D. For example: a helicopter is always an aircraft.
- Consistency of ABox with respect to TBox - determine whether individuals in ABox do not violate descriptions and axioms described by the TBox.
- Find classes that match known individuals: if Apache is an aircraft and it can hover, it is an instance of Helicopter.
- Check an individual - check whether the individual is an instance of a class.
- Retrieval of individuals - find all individuals that are instances of a certain class.
- Consistency of individual descriptions: Is the knowledge specified for an individual consistent with other known individuals and classes.

Bock, Haase, Ji, and Volz [13] split the reasoning tasks into something called TBox reasoning tasks and ABox reasoning tasks. The TBox reasoning tasks perform the

following functions:

- Satisfiability checks: whether a class C can have individuals according to the current ontology.
- Subsumption checks: whether a class D subsumes a class C according to the current ontology.

The ABox reasoning tasks are listed by Bock *et al.* [13] as:

- Consistency checks: whether the ABox is consistent with respect to the TBox.
- Instance checking: checks whether an assertion is entailed by the ABox.
- Retrieval problem: retrieves all individuals that instantiate a class C.

The Protégé editor allows for the implementation of different reasoners [70]:

- Pellet - open source, implemented in Java ([50]).
- Fact++ - open source, implemented in C++ ([91]).
- RacerPro - commercial.
- KAON2 - commercial, free of charge for universities and for non-commercial academic use.

The previous paragraphs discussed the ontology editor, the ontology language used in this research project and the reasoning mechanism. Ontologies can be constructed, edited and visualised in the editor and checked by using reasoners in the editor. The next section discusses the testing of ontologies.

2.7.5 Ontology Testing

There are many ways to test ontologies and three approaches are proposed by Yu, Thom, and Tam [99]:

- Gold standard evaluation that compares ontologies.
- Criteria-based evaluation. The ontology is tested for consistency, completeness, conciseness, expandability and extensibility. These tests can only be done with human interaction.
- Task-based evaluation. This approach evaluates an ontology based on the competency of the ontology in completing tasks. In taking such an approach, one can judge whether an ontology is suitable for the application or task in a quantitative manner by measuring its performance in the application.

Murdock, Buckner, and Allen [59] propose that testing is classified into three categories namely:

- **Structure**: Testing the structure of the ontology.

- **Functional**: Measuring if the ontology is suitable as a representation of the target domain. The gold standard approach also falls into this category.
- **Usable**: Testing the usability within the domain.

2.7.6 Different Types of Ontologies

Ontologies in CS are constructed to fulfil a purpose and therefore ontologies differ in application and size. Ontologies can therefore be categorised in a number of ways.

Sanchez *et al.* [74] list three different types of ontologies, categorised according to the scope of the ontology (the same classification proposed by Gruber [34]):

- Top-level ontologies which are at a high level of abstraction and describe general concepts that are independent of a specific problem or domain, for example the Basic Formal Ontology (BFO). These are also called upper or common ontologies and can be used in several domains.
- Ontologies that are adapted from top-level ontologies but are related to a specific domain or task, also called a task ontology.
- Application ontologies which are developed to solve a specific problem in a domain.

Fensel [27] proposes using the following classification:

- Domain ontologies.
- Meta data ontologies.
- Generic ontologies.
- Representational ontologies.
- Task ontologies.

Another classification method is presented by Gomez-Perez, as cited in [74], who classifies ontologies as light-weight and heavyweight. Light-weight ontologies contain concepts, relations between concepts and properties of concepts. Constraints and axioms are added to heavy-weight ontologies to make sure the intended meaning of concepts are captured in the ontology ([65]).

From all the different classifications listed above, domain ontologies are most widely used in CS. Ontologies cannot always be categorised and Dragan *et al.* [22] explain that it can happen that an ontology is classified using a combination of different types. For example, when an ontology that captures domain knowledge is expanded to be a task ontology. In CS, the most widely used application of ontology is to integrate different systems and enable effective communication among them.

2.8 Summary

Philosophers use the term ontology to describe the objects in the world and build a theory around what exists. Computer scientists use the term ontology as a specific conceptualisation, they design an artefact. Although these ontologies are different, the ontology of CS did evolve from the ontology of philosophy. The common need of computer scientists to have a shared representation of knowledge can be seen as one of the main reasons why ontology moved into CS. Although the use of ontologies in philosophy and CS differ, ontologies draw the fields closer together.

Ontologies have several advantages in their use in CS. They provide a structured way to analyse domain knowledge. They provide the ability to re-use knowledge and a way to share that knowledge between software modules and users. They also provide reasoning to check the consistency of classes and can therefore ensure that an ontology is meaningful and correct with minimum redundancy [44].

Modelling and simulation are traditionally complex to conduct but ontologies can be beneficial in understanding what to model and how to integrate different simulation models.

The scope of computer systems in the military environment covers a wide spectrum of complexity and uses and often these different systems have to share information. Ontologies have been demonstrated to be a viable solution to the problem of sharing information and to provide a formal knowledge base of military information. Although there have been a number of ontologies developed for use in military systems and a lot of effort put into developing ontologies in modelling and simulation, the full potential for military simulation system still has to be reached. Due to the sensitivity of the information the military community do not publish their work as openly as others so it is possible that there are very successful implementations of ontologies that have not been published or made available for public use. However, several success stories of ontologies used in military simulation systems act as examples and are of great value to future implementations in the military domain. Many of the techniques used and the experienced gained by these implementations were applied in this research project.

Ontology engineering covers the life-cycle of an ontology. Protégé was introduced in this chapter as the editor of choice for this research. Ontology tools were discussed as well as the different types of ontologies. The next chapter explains the theory of design research.

CHAPTER 3

Design Research

Design research was adopted as the research methodology for the execution of this research study. This chapter discusses the design research process and the application thereof in this research project.

One of the key factors of conducting successful research is adopting a solid and accepted research methodology. Such an approach will ensure a better research process, making the results more accurate and credible [63]. There are several recognised methodologies that can be followed when doing research, for example those described by Goddard and Melville [31] and Oates [63]. This chapter describes the methodology applied in this research project, namely *design research*, to address the research sub-question: How can an ontology be constructed that will capture the knowledge of a countermeasure simulation system environment?

In recent literature, Hevner *et al.* [39] describe how information systems play an increasing role in everyday life. Because of this growth in systems in Computer Science (CS), the pressure is continually increasing to design good software and robust systems. Design research is a research methodology that can improve the design and development of systems in CS [39]. The first part of the chapter describes the theory behind design research, the purpose of doing design research and how it is conducted. The different stages of design research are described and the guidelines for doing design research Hevner, March, Park, and Ram [38] discussed.

The simulation system environment described in this study is a complex systems capable of undertaking countermeasure evaluation. Although it is a productive working system, there are issues that needs to be improved in the use of the system. After an investigation, a suggestion was made that an ontology be constructed to support the simulation environment. Design research was chosen as the research methodology to be followed to achieve the desired outcome of the artefact which will be an ontology. The second part of the chapter discusses how the stages of design research are applied to the development of an ontology for the countermeasure simulation system and the steps followed to reach a stage where a conclusion could be made whether ontology does indeed play a supporting role in the countermeasure simulation system. Each stage of the design process is discussed briefly so as to give an overview of

what is going to be described in the rest of the chapters that will cover the process in full.

3.1 What is Design Research

Research generates new knowledge, but sometimes the only way to generate new knowledge is by creating a new product, which is often the case in the fields of CS [92]. The research effort then goes into the design and creation of an artefact, or some kind of product. The artefact is built whilst in the process of doing the research. This type of research is known as design research.

Two words make up the concepts of ***design research***: design and research. ***Design***, according to Friedman [29], is a process followed to reach a specific goal. The goal can be to solve a problem, to improve or to create something. Hevner *et al.* [39] notes that design is used in many different disciplines in different ways, for example by engineers and architects. Software engineers want to build quality software and design a solution to solve a problem.

Research, as defined by Oates [63], is done to create new knowledge or to contribute to the body of knowledge. There are many unknowns and the goal of research is to try to lessen the unknown by following a systematic approach. Hevner *et al.* [39] give a list of characteristics of research that can be summed up by saying that a research process starts with a **problem**, then follows an **iterative plan** towards a **goal**, while taking into account certain **assumptions**. Oates [63] also describes research by describing six essential elements of research, namely: purpose, product, process, paradigm, participants and presentation.

By consolidating the two terms described in the previous paragraphs, ***design research*** can be described as a process followed to reach a specific goal, the goal being to create new knowledge. The goal of design research in CS is to create a computer science artefact that improves or solves a problem. The purpose of design research is to improve, and according to Vaishnavi *et al.* [92], this emphasises the nature of the activity. The improvement must be grounded in theory, and whilst the artefact represents a solution, the generation of knowledge and adding to theory remains a core objective of design research. Hevner *et al.* [38] describe it as a research cycle that creates and evaluates artefacts intended to solve identified organisational problems. Samuel-Ojo, Shimabukuro, Chatterjee, Muthui, Babineau, Prasertsilp, Ewais, and Young [73] describe it as a process of gaining knowledge and understanding of a design problem and its solution during the building and application of an artefact.

To summarise, Hevner *et al.* [39] formally define design science research as follows:

> *Design science research is a research paradigm in which a designer answers questions relevant to human problems via the creation of innovative artefacts, thereby contributing new knowledge to the body of scientific evidence. The designed artefacts are both useful and fundamental in understanding the problem.*

This definition by Hevner illustrates that design is evident in the creation of artefacts and new knowledge is created doing research. The artefacts that are referred to are defined by Simon [81] as things that do not occur in nature but are produced by humans, thus something unnatural. Specifically in CS an artefact would be a software product that offer a solution to a specific problem or set of problems.

Design is a complex task and there is a lot of evidence relating to products and projects in CS that failed to offer an effective solution. The purpose of design research is to minimise the risks of complexity by offering a robust process to be followed.

3.2 Design Research Guidelines

This section discusses guidelines, as proposed by Hevner *et al.* [39], to follow while performing design research.

Design research is about solving a problem and gaining knowledge while doing so. The definition by Hevner *et al.* [39], highlight that a problem is solved by developing a product and knowledge is generated by the lessons learned during building and applying the solution. The two concepts here, the solution to the problem and the generation of knowledge, are the important parts of design research to be addressed during the research process. The knowledge gained must be identified, communicated and reported in some way.

To address these concepts and to ensure effective design science research Hevner *et al.* [38] propose seven guidelines to follow when doing design research as follows:

1. The first requirement is that the output of the research must be an artefact used in a field of CS. It is not essential that the artefact must be a completed product at the end of the research process, it might happen that enough knowledge is gained in the analysis and design of the artefact and therefore the artefact does not have to be completed or even implemented.
2. The goal of the research must be to solve a specific and relevant problem.
3. The resulting product must be evaluated by appropriate methods.

4. The research process and the creation of the artefact must add to the body of knowledge.
5. A proper research process must be followed.
6. A proper search and investigation into appropriate solutions must be done, taking into account the environment wherein the artefact will be applied and used. This will ensure that the most suitable solution is implemented.
7. The outcome of the research must be reported effectively to all involved parties.

By following these guidelines, Hevner *et al.* [38] aim to increase the value of design research in contributing to new knowledge. Researchers following and presenting their research according to these guidelines will have greater success in contributing to the use of academic research in product development. It furthermore sets a base to accentuate the difference between design research and product development.

Design researchers in CS must often defend the accusation that the design research performed is product development and not research. Oates [63] and Vaishnavi *et al.* [92] address this by proposing a clear distinction between design research and product development. Oates [63] explains the difference in developing a product and doing design research by focusing on the academic contribution of design research. The design research must follow a process of analysis and explanation. The research must also defend an argument and be evaluated. Oates [63] names six actions that must be addressed in any research, and calls it the 6P's of research: purpose, products, process, participants, paradigm and presentation. The presence of these six items ensures that research is indeed being performed. Vaishnavi *et al.* [92] propose that design research produce, for a specific group of people, new knowledge. They illustrate that successful products in CS often fail to produce new knowledge whereas projects that experience difficulties tend to be different - they add knowledge gained through difficulty. They emphasise that such knowledge is different from the knowledge put into the design of a product.

Considering the academic guidelines of Oates [63] and the proposal by Vaishnavi *et al.* [92], design research is different from product development if it follows an academic process and new knowledge is created. This is strongly supported by the guidelines proposed by Hevner *et al.* [38]. One of the academic research actions named by Oates [63] is to follow a formal process, and the following section describes the stages in the design research process.

3.3 Design Research Process

The design research process is often depicted as a series of tasks followed to solve a problem. According to Oates [63] and guideline number five proposed by Hevner *et al.* [38], the process must follow a systematic plan to ensure that the research is valid and trustworthy.

Most of the research processes to perform design research follow an iterative approach. There are several different methodologies used in computer science, for example the methodology used by Blessing and Chakrabarti [12] that is based upon prescriptivist and descriptive studies to understand the problem.

The design research process chosen for this research project is described by Vaishnavi *et al.* [92] who develop and describe a methodology for doing design research in CS. The methodology consists of the following five stages: awareness of the problem, suggestion for a solution, development of the solution, evaluation of the solution and the conclusion of the design process, as shown in Figure 3.1.

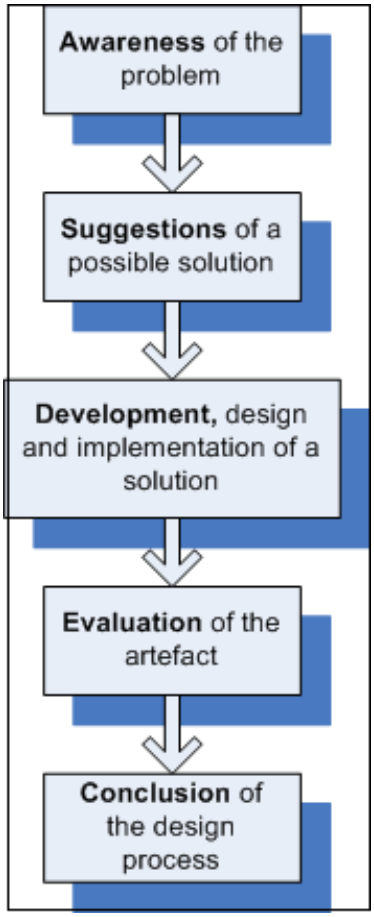

Figure 3.1: Design Research Process (adopted from Vaishnavi and Kuechler [92])

The following sections briefly describe the stages of the design research process.

3.3.1 Stage One: Awareness of the Problem

The first stage is to recognise and state that a problem does exist. There are various ways in which researchers become aware of problems to solve or improvements to make:

- Authors identify problems and future work in their works of literature.
- Clients experience problems.
- New findings.
- Technological developments.

The output of the awareness stage is a set of statements that describes the problems that need to be addressed. Although no solutions are proposed yet, the possible benefits and advantages that a solution will have can also be stated. These statements will assist in determining the goal of the intended research in finding a suitable solution.

These statements made during the awareness stage are very important for the rest of the research process and must be taken into account through all the following stages. In the development stage, the efforts will be designed towards finding solutions, using the statements as guidelines. During evaluation, the statements are used as criteria for evaluating the research output.

3.3.2 Stage Two: Proposing Suggestions

Following awareness of the problem, possible solutions must be formulated. Suggestions emerge from looking in detail at the current knowledge base and theories. It might be possible that researchers, when stating a problem in literature, also suggest possible solutions and these suggestions need to be investigated. If no solutions can be easily identified, it is then necessary to investigate novice ideas. A proposal in the form of possible solutions to the problems stated in the previous stage is produced.

3.3.3 Stage Three: Development

During the development stage, one of the proposed solutions is investigated, designed, developed and implemented either partially or in full. The output of the development stage is an artefact and therefore a full software development cycle must be followed, as emphasised by Oates [63]. The research methodology process does not replace the need for a proper development methodology to produce the artefact. The development process is one stage of the complete process of design research, as shown in Figure 3.2.

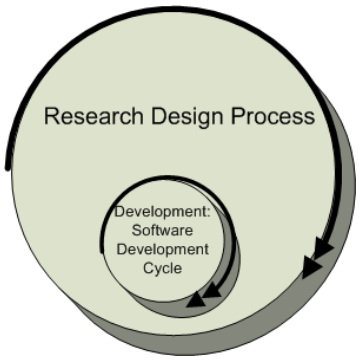

Figure 3.2: Development Process inside Design Research Process

To develop the artefact, traditional software methodologies that follow the proven method of requirements and analysis, design and implementation, testing and validation and maintenance, are often used. There are popular methodologies that are very successful applied in design research. Oates [63] describes the use of prototypes in the design research development process stage. Prototypes are very popular because of close user involvement and the delivery of an early working model Prototyping [71]. The object-orientated paradigm is another methodology and focusses on modelling entities as objects and presenting the real world as closely as possible. Alhir [2] explains the approach of object-orientated design as a focus on entities as complete units, taking into account both behavioural and structural properties. More recently, agile development methods became popular, aiming to solve the problems of traditional development methodologies [78]. It is also possible to use a combination of methods, for example using a prototype in the agile development phase.

It is beyond the scope of this research project to discuss all the development methodologies. The choice of development methodology will be greatly influenced by the type of artefact that is going to be created Vaishnavi *et al.* [92]. The artefact developed during this research is an ontology. The rest of this section will therefore focus on a development methodology that are specifically applied to the construction of ontologies and ontology engineering.

In Section 2.7, ontology engineering is discussed in detail. There have been several methodologies used in ontology engineering but they have been around for quite some time and ontology engineering is not a subject of recent research [10]. Those methodologies are outdated and their approaches either incremental and iterative or comprehensive. They share a series of logical phases comprising of conceptualise, deploy, test and refine. A group of researchers at AI3 developed a new methodology, called *Adaptive Methodology* for the development of ontologies [10].

Adaptive Methodology

Adaptive methodology is a result of AI3, a group of researchers that developed a new method to create ontologies. The background of the development of a new methodology was described by Bergman [10].

The AI3 researchers had several goals they wanted to achieve and requirements to meet when they designed the adaptive methodology process:

- **Lightweight domain ontologies**
  The ontology created using the methodology they proposed must be lightweight and designed for a specific domain or subject area.
- **Adequate description of the context**
  The ontology must have a thorough description of the context of the domain. It can take the form of a comprehensive list of concepts definitions. Such a description will solve ambiguous meaning of concepts. One of the risks of ontology engineering is constructing an ontology that is not meaningful for the people involved. The description of the context provides a clear meaning of the concepts but the structure of the concepts must be logical and meaningful. The concepts of the domain make up a network of connections that must make sense.
- **Incremental process**
  The construction of an ontology is an incremental process. The process starts of small and expands with each iteration. The first two phases are devoted to scoping and prototyping. After the working ontology is constructed, the remaining phases are repeated over a number of increments. These are to test, maintain, revisit and extend the working ontology. This method ensures that deployment of the ontology proceeds incrementally and lessons learned during each iteration can be applied in the next. Another advantage of development in incremental steps is that complex functions and extension of the scope are only implemented if there are benefits in doing so.
- **Use existing sources of documents**
  Most systems in CS are not developed from new information and the concepts and structures already present in the domain should not be ignored. They are not used directly, but must act as a source of information to the developers.
- **Separating concepts and instances**
  The discussion of descriptive logics in Section 2.7.2, explains how the concepts in an ontology are either TBox statements (terminological statements), or ABox statements (assertion statements). In CS, the knowledge base of a domain is

made up of these ABox statements that describe the specific instances of the knowledge base and the TBox statements that are a set of definitions from the knowledge base.

- **Tools for the ontology**
  The last but most important requirement that the researchers of AI3 had was the provision of tools that are specifically designed for a task or a function.

Using the above-mentioned requirements as a guideline, the adaptive methodology process was designed and the following steps identified:

1. **Determine the scope**
   This first step of the methodology is to determine the extent of the ontology. A specific goal or purpose must also be identified. This will meet up with the guideline that Hevner *et al.* [38] propose.
2. **Use existing sources**
   Adaptive methodology emphasises the use of existing resources of information that exist in an organisation. There are almost always some existing documents that can be a source of information that can assist in identifying concepts.
3. **The prototype ontology**
   The prototype step is separated into two different phases. The one phase concentrates on creating the instances and the other one on the conceptual relationships between these instances. The prototype step produces the first operating instance of the ontology. The prototype structure is important since it communicates to the project sponsors the scope and basic operation of the starting structure. This stage often represents a decision point for proceeding or it may also trigger the next budgeting phase.
4. **The working ontology**
   The prototype is expanded to create a working ontology. The working ontology has to be complete enough to provide a full set of concepts and be usable in a working environment. If the ontology was created in an ontology editor, these functionalities of the editor will be available for use.
5. **Testing the working ontology**
   The purpose of testing the ontology is to ensure an artefact of good quality. Gomez-Perez [32] describes the evaluation process by defining a set of questions:
   - What can be tested? Evaluation can be done for any definition or a set of definitions in the ontology.
   - Why test and evaluate the ontology? To guarantee to users that the on-

tology is correct and complete. The purpose of testing the ontology is to ensure the following:

- There are no inconsistencies.
- The structure of the ontology is correctly presenting the concepts in the target domain.
- The concepts in the ontology are consistent.

- What to test against? Evaluation of the ontologies should be performed against a frame of reference. It can be a set of competency questions, requirements or testing it against the real world. Evaluation of the software environment should be performed against its requirements.
- When to test? Evaluation is an iterative process that should preferably be performed during each phase. If performed at every iteration, wrong, incomplete or missing information is detected as soon as possible.
- How to test the ontology?
- Who performs the tests? The development team, other developers and end users evaluate different features of the ontology definitions. While the development teams evaluate technical properties of the definitions, end users evaluate their actual utility within a given organisation or by other software agents.

6. **Use and maintain**
   The ontology is used in the environment it was developed for. The ontology is updated by performing necessary maintenance.
7. **Revise and extend the working ontology**
   The building of an ontology is an incremental process and not a once-off development effort. Bergman [10] stresses that the domain ontology must grow as the understanding of the domain grows and matures. The last step in the adaptive methodology is then to extend the ontology to keep up with the growth in domain growth. This step also includes documenting the ontology.

Figure 3.3 illustrates the design research process, extended with the adaptive development methodology as the third stage of the process.

The steps required to build an ontology with the Adaptive Methodology differ from traditional software development. Because of the familiarity of traditional software development, the stages in traditional software development are compared to those in the adaptive methodology process. Table 3.1 contains the comparison between the different stages of design research and traditional software methodology.

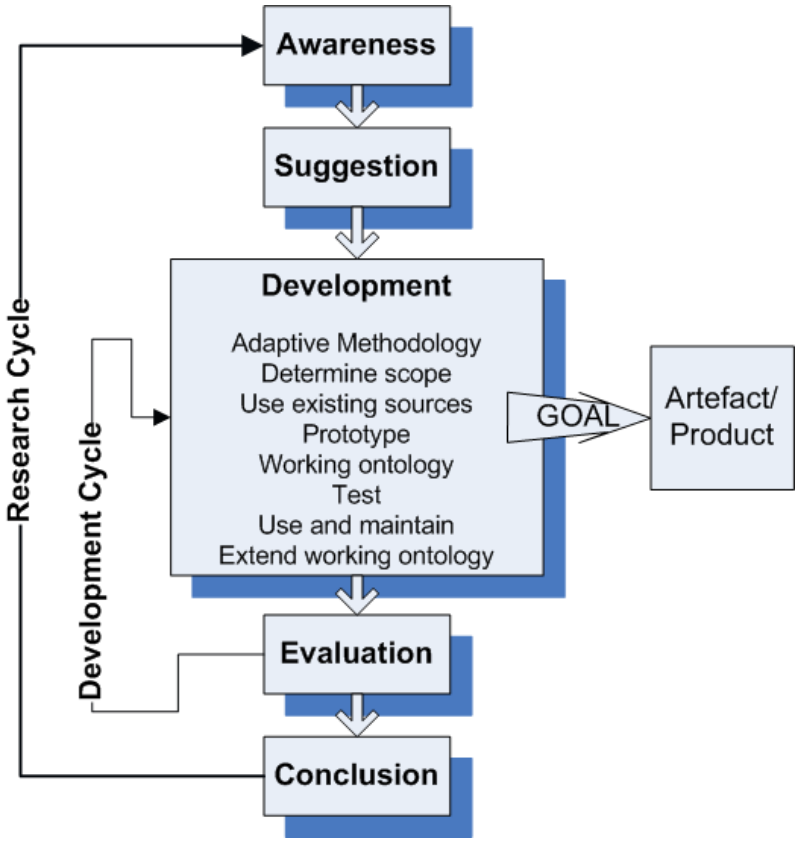

Figure 3.3: Overview of Design Research Process and the Development Stage

Adaptive Methodology	**Traditional Software Development**
Determine domain and scope of ontology Acquire domain knowledge from experts	Requirement engineering
Examine existing sources Develop prototype	Analysis
Expand the prototype	Design
Develop a working ontology	Implementation
Test the ontology	Testing and validation
Use and maintain Extend working ontology	Maintenance

Table 3.1: Adaptive Methodology versus Traditional Software Development

The development stage of the adaptive methodology process can continue to the next stage, evaluation, if the process reach a point where the artefact is deemed to be in a state to be evaluated.

3.3.4 Stage Four: Evaluation

After development and implementation of the artefact, the next step is to evaluate it. Evaluation of an artefact in the design research methodology context was described

by Hevner [40] as the systematic process to find out if something is useful, has some merit or add significant value. The evaluation must provide evidence that the artefact brings about an improvement in some part of the domain. The outcome of the evaluation stage can give possible input to a new iteration of development or suggestions for expanding the artefact.

According to Hevner [40], evaluation of the artefact is performed for three reasons. Firstly, to ensure that the artefact is applicable to the domain and that it adds value, secondly to give the research credibility and, thirdly, to determine the practical value of the research.

To evaluate the role of the artefact in the organisation, it is necessary to look at the issues raised in the awareness stage. The issues identified during awareness will guide the evaluation of the artefact and must be evaluated according to the criteria set.

An important issue to consider when undertaking design research, is that the researchers must prove that what was done was indeed research and not normal product development.

The artefact must be practically evaluated. Oates [63] lists several criteria for evaluation namely functionality, completeness, consistency, accuracy, performance, reliability, usability, accessibility and aesthetics. Oates [63] recommends that the methods chosen to do the evaluation must be aligned with the research objectives.

Hevner [40] suggests another process of evaluation that can be summed up in three stages:

- Determine beforehand what part of the artefact to evaluate. Different users would want to evaluate differently so having a contract gives the evaluators a clear goal. Hevner calls this negotiating a contract.
- Analyse the artefact by collecting data according to the contract. The person performing the analysis must follow a systematic approach and understand the problem well.
- Report on the outcome of the analysis or investigation.

According to Oates [63], the evaluation process can result in suggestions to modify the design process or the designed artefact. The evaluation stage is an iterative one and it is done after each new development cycle has been finalised. A decision has to be made when the last cycle of development/evaluation is reached and a conclusion is required. If the evaluation of the outcome of the development stage are satisfactory and no more iterations are going to be done, the design process reach the final stage, the conclusion.

3.3.5 Stage Five: Research Outcomes and Contribution

The final step in design research is to conclude and present the research. The contributions that the research made to the body of knowledge must be listed in a clear way. The purpose is to publish the research and the results of the research. It is important to write up the knowledge gained and the contribution of the knowledge. It is also important to note the outcomes that will lead to further research.

The previous paragraphs discussed the five stages of design research. After the completion of these five stages, an evaluated artefact is produced.

3.4 Design Research Outputs

Design research is about creating an artefact but the fields of CS are varied and therefore many different kinds of artefacts can possibly be created. These artefacts are the outputs of the design research.

Oates [63] describes the possible types of artefacts that can be created by design research as constructs, models, methods or instantiations.

- Constructs - The conceptual vocabulary of a domain that is constructed in the initial stages of design. It is refined throughout the development cycle.
- Models - Models are a presentation of the relationships between constructs.
- Methods - A set of steps followed in order to perform some function.
- Instantiations - The practical implementation of constructs, models and methods in an operational environment.
- Theory development, described by Vaishnavi *et al.* [92] to be the development of new and better theories in the domain.

3.5 Advantages and Disadvantages of Design Research

Design research has many benefits but there are also pitfalls to be taken into account. One of the main advantages of doing design research is that there is a product, or at least part of a product, to show for the research effort. Apart from having the product, it is also possible that some knowledge in the domain might not be accessible without having that product. The researcher has more control over the time spent and the conditions of the research. The designed artefact might have additional or unexpected value, in addition to being a tool for conducting research.

Design research is not always the correct research methodology to use and has disadvantages. One of the main reasons for the process to fail is the difficulties encountered in the real world. This might result in the development of the artefact not being

concluded up to a point where an evaluation or conclusion can be made. Since the research objects are of an artificial nature, they are imperfect or inaccurate. Mistakes and misleading conclusions might be drawn by using the artefact. The necessity to build artefacts might be costly, dangerous, or disadvantageous in some other sense. It might require a lot of resources, just to create the artefact, before research can be done. By trying to answer a research question by means of an artefact, one might (consciously or subconsciously) be tempted to develop the artefact biased towards an answer one would expect or like to find out. Design research imposes the danger of subjectivity towards the research.

Whereas the previous sections discussed design research in general, the next sections are a brief summary of how the design research stages are going to be applied in this research project.

3.6 The Stages of Design Research Process Applied to this Study

Every research project is unique and the stages of design research described by Vaishnavi *et al.* [92] in the previous section are a general guideline in undertaking design research. This general process will be applied to fit this research project.

The following section is a brief discussion of how the five stages of design research are going to be applied to the research process of determining if ontology plays a supporting role in the simulation environment. Detail of each stage is further discussed in Chapter 5.

3.6.1 Awareness

Military aircraft need to be protected against enemy attacks. These attacks are carried out in a number of ways, one of them being shoulder-launched missiles. A simulation system exists and is used by several different people and groups to do evaluation studies of countermeasures implemented on these aircraft. The simulation system is complex to use and several issues prevent optimum use of all the functionalities in the system. The simulation system operates in an environment which consist not only of the system itself but several tools, it is therefore referred to as the simulation environment.

There are developers, model builders and other users involved in the system. Inconsistencies in the terminology used between different users often led to frustration and the incorrect use of concepts. There is lack of a common vocabulary that is shared by everyone involved in building and using the system.

One of the main characteristics is the ability of the system to execute models at

different levels of detail. This poses a problem to users of how to know at which level of detail a model is implemented at. Users only interact with the system at a certain level and need a more technical insight into model detail to know what is available in the system.

The system and model repository are constantly being updated by improving existing models and adding new models. Minimum requirements exist for a model to be operational in the system. However, the developers need a mechanism to identify those requirements in a meaningful way.

A simulation consists of a scenario that is built up from models interacting. The models interact by a set of rules but there are currently no rules that verify model behaviour when a scenario is constructed.

The scenario input files of previously run simulations are stored in files and the results written in reports. There are a need to have access to those previously run simulations without referring back to technical reports or having to investigate simulation files.

In the awareness stage, the evaluation criteria must be defined as a set of guidelines to be followed during evaluation.

The previous paragraphs gave a brief overview of some of the issues of the simulation environment. A detailed discussion of the awareness stage is given in Chapter 4. The following step in the design research process is to put forward suggestions as to what will possibly solve the issues identified.

3.6.2 Suggestion

The simulation environment provides a platform to countermeasure evaluation studies but there are shortcomings as identified in the previous section. These shortcomings have been identified and the next step is to find a solution that can possibly address them and therefore improve the system.

A previous research project was undertaken by Lombard [51] in the domain of the simulation environment to address the lack of proper graphical views of system layout and the need for a standard notation for documentation of the simulation models. A study was done to investigate the use of a combination of IDEF and UML diagrams. The implementation of the suggested diagrams partly solved the problem of communicating the simulation concepts to different users. It also provided a graphical view of system components and process flow. Examples of IDEF diagrams are illustrated in Figure 3.4 and Figure 3.5.

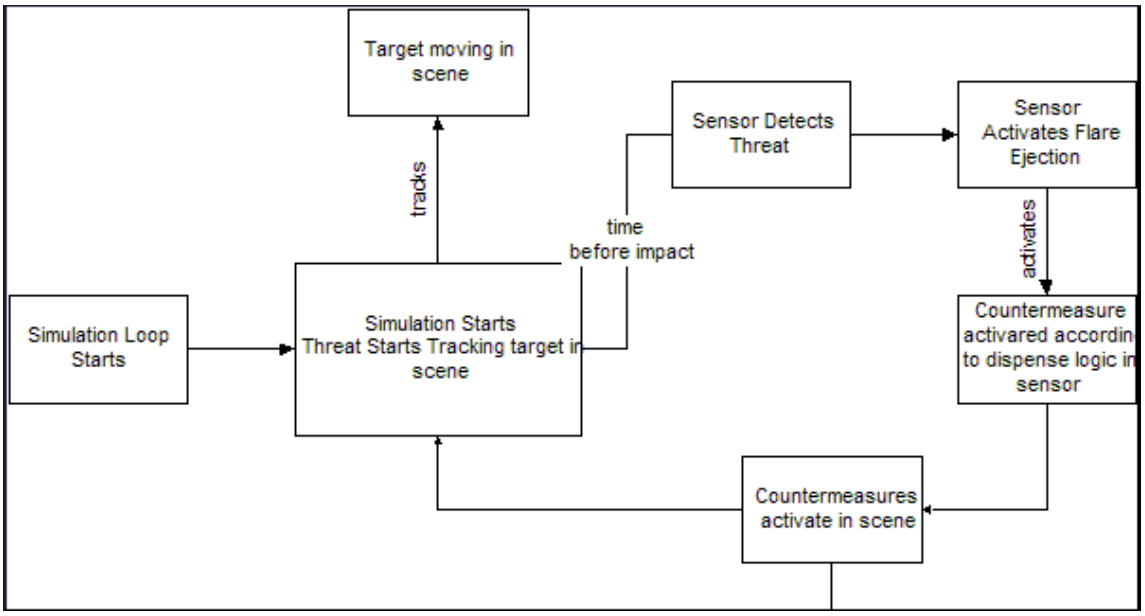

Figure 3.4: IDEF3 Process Diagram

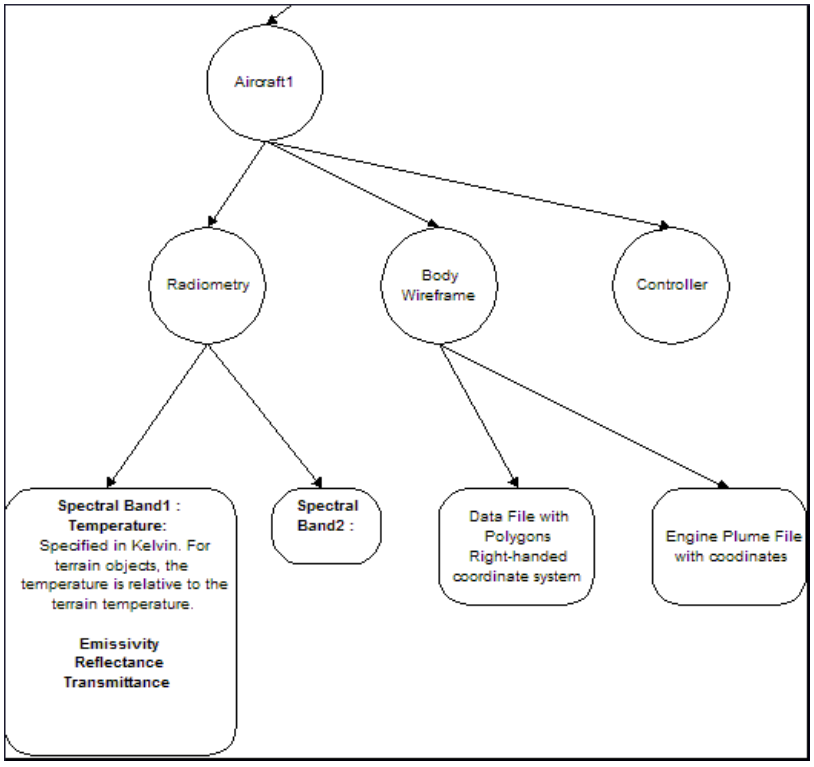

Figure 3.5: IDEF5 Ontology Diagram

During the research to find suitable notation, several possibilities were mentioned for further investigation. One of those suggestions was to use ontologies in the simulation environment and it was identified as a topic for future research.

The theory of ontologies has been discussed in Chapter 2. According to Noy *et al.*

[62], an ontology provides the following advantages:

- It provide a shared, common understanding of the structure of information among people or software agents.
- Enables the reuse of domain knowledge.
- Makes domain assumptions explicit.
- Separates the domain knowledge from the operational knowledge.
- Analyses domain knowledge.

The first research sub-question stated the task to determine the requirements or concerns of a countermeasure simulation system environment that ontology technologies support. By listing the requirements and issues, a solution can be formulated. The suggested solution is that by creating an ontology for the simulation environment, several issues can be addressed and the advantages of using an ontology, as mentioned by Noy *et al.* [62], can be achieved for the simulation environment.

The awareness of the issues stated and the proposed solution are summed by the following:

There exists a simulation system that
performs evaluation studies to determine countermeasure effectiveness but
it has shortcomings such as
too much of the success depends on the skills of the users, there is no common language between all the users and in-depth knowledge of models is required to use them in a correct way.
This can possibly be solved by implementing a tool, such as an ontology, that will support the system by
providing a shared, common language that is less dependent on user skills.

Section 4.3.2 provides a detailed discussion on how and why an ontology is suggested as a proper solution.

3.6.3 Development

The suggested solution is to construct an ontology. The purpose of the development stage is to provide a method to construct an ontology that will capture the knowledge of a countermeasure simulation system environment. To develop an ontology for the simulation environment, the steps as described in Section 3.3.3, the Adaptive Methodology process, are followed.

1. **Scope and Purpose**
   The first step is to scope the purpose and the extent of the ontology. In the case of a domain ontology, the concepts of the domain must be included. However,

it is not necessary to include all the concepts of the domain. The level of detail will be determined by the purpose of the ontology.

2. **Use Existing Sources**
   The following step is to check if there are any existing sources of information that can be used. In the case of creating an ontology for the simulation environment, there are several different documents available for example modelling guides and user manuals. In Chapter 5, how the documentation of the models was used to gather information to create the ontology is discussed.

3. **The Prototype**
   The prototype structure is the first version of the ontology that is operational. The purpose of the prototype is to provide an example of the ontology and the editor that is used. The prototype for the simulation environment contains only a selected set of components from the domain. The concepts are on a high level and the nested structures of complex concepts were not included in the prototype.

   The prototype was developed by following the steps recommended in Ontology Development 101, written by Noy *et al.* [62], adapted to use the approach followed by Gerber, Kotzé, and van der Merwe [30], and resulting in the steps as illustrated in Figure 3.6.

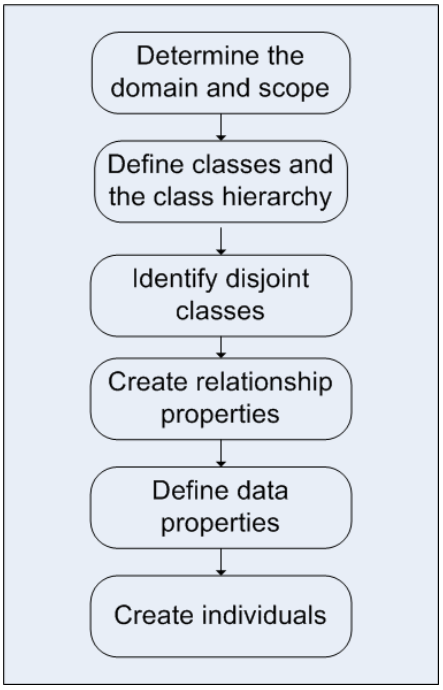

Figure 3.6: Steps to Develop Prototype Ontology

(a) The scope and domain of the prototype ontology is determined to contain

only a selected set of models in the countermeasure simulation scenario.

(b) Identify the classes and the class hierarchy.

(c) Identify disjoint classes.

(d) Create relationships between classes.

(e) Define data properties.

(f) Create individuals.

The prototype was developed in the Protégé editor that provides several built-in functionalities. OntoGraf, a Protégé plug-in provides a graphical view of the ontology. The Protégé functions used with the prototype will be discussed in Section 5.4.2.

Another ontology function introduced in the prototype, is the ability to reason about the contents of the ontology. A reasoner will typically be used to validate the ontology. Reasoners are discussed in the section on ontology engineering in Chapter 2.

The prototype has a very important role in demonstrating that the suggested solution is the right choice. With the prototype ontology it was possible to demonstrate the supporting role of an ontology in the simulation system environment, such as providing a shared, common vocabulary and providing graphical views of the concepts defined in the ontology. The development and functionality of the prototype is discussed in detail in Section 3.6.3.

4. **The Working Ontology**

The working ontology will be constructed by adding concepts from the domain to the prototype ontology and developing functionality to use the ontology. The working ontology contains a full set of domain concepts that describe the simulation models and model properties of all the concepts used in the self-protection application of OSSIM.

The following functionality will be available for use in the working ontology:

- The functionality provided by the Protégé editor.
- Interaction with a graphical user interface. The ontology will provide information to populate the elements in the interface.
- A high-level description of scenarios and their components.

5. **Testing the Ontology**

Testing the ontology is an important step in the construction of a usable ontology. In a small ontology, with a limited number of classes, it is relatively easy to identify problem areas but when there are a large number of concepts

with complex relationships, it becomes more important to test the ontology on a regular basis. Although evaluation and testing are used interactively, it is important to distinguish between testing the ontology during development and evaluating the ontology as part of the process of design research.

In the discussion on testing ontologies in Section 2.7.5, several methods to test ontologies are discussed. For this research project, the goal of testing is to validate the contents and structure of the ontology. The OWL 2 ontology language enable the use of reasoners to ensure ontologies without inconsistencies or unsatisfiable conditions (see Section 2.7.4).

The questions that Gomez-Perez [32] raises have been used as guidelines for the test process:

What will be tested? The contents and structure of the ontology are tested.

Why test and evaluate the ontology? Proper testing will ensure an ontology of good quality.

What to test against? The testing must be done against the objects in the simulation environment.

When to test? The working ontology is tested. After every iteration of extending the ontology, it will be tested.

How to test the ontology? To test the validity of the structure of the ontology, reasoners in the ontology editor are used to validate the ontology for the following:

- Completeness.
- Consistency checking: Consistency checking is used to find circulatory errors in the TBox component of the ontology.
- Unsatisfiable conditions.
- Classification.

Who performs the tests? The developers and users of the functionalities will test the ontology.

The ontology is developed and expanded in an incremental way therefore the ontology must be tested every time new concepts or new functionality gets added.

The test phase of the ontology is part of the adaptive methodology process and must not be confused with the evaluation phase of the design research process where the research is evaluated. The testing process for the ontology is described in Section 5.5.

6. **Use and Maintain**
   This is an iterative phase to test already developed functionality and maintain it by making changes where necessary. Examples of maintenance that take place in this step of the development process include the finalisation of incomplete individuals or the addition of data properties. Section 5.6 contains detailed descriptions of how the ontology is used and maintained in the simulation environment.
7. **Expand the Working Ontology**
   When the ontology is working, implemented and in use, the ontology can be extended by adding concepts or by the development of new functionality. The functionality in this project was added in order of importance as indicated by the users of the system.

Table 3.2 gives a brief overview of the functionalities that will be developed in each step of the development process.

Step in Adaptive Methodology	**Functionality Added**
Prototype ontology	Protégé editor and plug-ins Selected set of concepts to classes Graphical views of the ontology classes
Working ontology	Expand set of concepts GUI integration Validate scenarios
Use and maintain	Add properties to individuals
Extend working ontology	Reverse engineering scenarios

Table 3.2: Progress of Development in Each Step of Adaptive Methodology Process

A full description of the development of the different stages of the ontology is described in Chapter 5. An artefact, such as the ontology, is never fully complete and it is always possible to add new functionality but at a certain stage the decision must be made to complete the research process and continue with evaluating the ontology.

3.6.4 Evaluation

The development process of the ontology is constructed and expanded until a working ontology is in place. The evaluation of the created ontology can commence if the ontology is completed up to a stage where it is usable in the environment. The evaluation will follow the process of determining if the ontology does indeed play a supporting role in the simulation system. This evaluation stage will provide answers to research sub-question three: Does the use of ontology technologies support and

enhance the functions of the simulation system in the domain?

In the awareness stage, certain issues are raised and suggestions made. The ontology must therefore be evaluated by checking if the expected advantages of building an ontology for the simulation system, the suggestion, were indeed met and if the ontology does indeed play a supporting role for the countermeasure system. Does the ontology deliver as promised? Was new knowledge generated by developing an ontology to support the simulation system? In order to evaluate the contribution of the research outcomes, the list of requirements identified in the awareness stage will be followed.

To give credibility to the research and emphasises that research was done and not product development, the requirements listed by Hevner *et al.* [39] will be applied. The following need to be identified or satisfied:

1. A clear research question or questions must be stated.
2. An artefact was created and represented in some way.
3. Formal design processes were followed in building the artefact.
4. It must be explained how the artefact and the design processes are grounded by the knowledge base. Were there theories that support the artefact design and the design process?
5. The evaluations performed during the design of the artefact must be listed as well as the improvements made in each design cycle.
6. How was the artefact introduced into the environment? Is it any better than previous solutions?
7. Was there new knowledge created that contributes to the body of knowledge?
8. Proof that the research questions are satisfactorily addressed.

If the conclusion is made that the above-listed requirements are met, the research process can safely claim that research was done, and that product development was only one phase in the research process.

The final purpose of evaluation is to evaluate the practical impact of the design of the ontology. What worked and what did not work? This is very important for future work and add to the body of knowledge.

A discussion of the evaluation and the outcome of the evaluation of the ontology are continued in Chapter 6. The evaluation of the ontology is closely related to the research contribution.

3.6.5 Research Outcome and Contribution

The final stage in design research is the conclusion of the research process. The purpose is to, as described by Vaishnavi *et al.* [92], conclude the findings of the research and publish the outcome. The outcome of the development phase, the ontology, must be properly documented. The ontology engineering process and the results of the evaluation phase need to be documented. The research must prove the value of using an ontology in the simulation system, and the conclusion stage must go beyond reporting the constructed ontology. The focus must be on the reporting of the knowledge gained.

The knowledge gained can be categorised according to the standard set by Vaishnavi *et al.* [92]. Solid knowledge is knowledge gained that can be applied again. 'Loose' knowledge is knowledge that is not solved, there are outstanding explanations. This knowledge forms the foundation for the formulation of further research.

Following the research process as described, the research sub-questions will be answered as follows:

- The awareness and suggestion stage of the research process will *determine the requirements or concerns of a countermeasure simulation system environment that ontology technologies support.*
- During the development stage, the construction of an ontology will show *how an ontology can be constructed that will capture the knowledge of a countermeasure simulation system environment.*
- Evaluating the role of the ontology and the research outcomes and contribution will provide evidence to justify the suggestion that *the use of ontology technologies support and enhance the functions of the simulation system in the domain.*

Adequately answering the three sub-questions will provide the answer to the main research question: *How can an ontology technologies be used to support a countermeasure simulation system environment?*

The research outcomes and contribution of the research in creating the ontology will be discussed in full in Chapter 7.

3.7 Summary

This chapter describes how the methodology of design research is going to be applied to implement an ontology for a countermeasure simulation system. According to Piirainen, Gonzalez, and Kolfschoten [66] design research must solve a relevant problem and add to the body of knowledge, through the design of an appropriate

artefact. The design research method and the reasons for choosing the design research methodology were described. There are certain steps to follow when doing design research and those steps were described as five different stages. Every stage must be applied to the research project.

A software development methodology must be applied in the development stage of the research process. Adaptive methodology was selected and a description of how adaptive methodology is going to be applied to construct an ontology given. The output of the process will be a specific artefact, the ontology, which will support the functions of the simulation system, and will add to the body of knowledge by displaying how ontologies can be applied to support the simulation system environment.

The following chapter describes the countermeasure simulation system environment. The purpose is to provide information for the awareness stage of design research. The background information on the simulation system and the effort involved in undertaking a countermeasure evaluation study will give insight as to where the outcome of this research project will be applied in the simulation environment.

CHAPTER 4

Awareness and Suggestion

To provide insight into the problem statement it is necessary to discuss the simulation system environment. This chapter provides background information, proceeds to describe the awareness stage of the research process and finally discusses a suggested solution to be implemented.

As described in Section 3.3, awareness of a problem and suggestion of a solution is the first stage in the design research process. Vaishnavi *et al.* [92] describe this stage as the process occurring when researchers became aware of a specific problem and subsequently suggest a research process to solve the problem. This chapter describes the background of the system and the environment wherein the system operates. It explains the road that leads to the decision to construct an ontology and the reasons behind the decision to suggest an ontology as a possible solution to address the issues in the simulation environment.

The first part of the chapter describes the simulation environment to provide background and understanding of the system and its components. All the steps involved to perform a complete evaluation study are described. This will provide the necessary insight into the issues and concerns that are experienced by the people involved in the system. By highlighting these issues it is possible to formulate requirements for a possible solution. The list of requirements is compared to that which ontologies can offer. This led to the suggestion that a development process be followed to investigate the possibility that an ontology can indeed provide a solution to these requirements.

After providing an overview of the simulation environment, the remainder of the first part of this chapter covers the self-protection application of the Optronic Scene Simulator (OSSIM). The environment is described according to the steps taken to perform a countermeasure simulation study, the models in the system and how the results of a simulation is presented to the client.

4.1 Overview of the Simulation System Environment

Computer-based simulation systems are used to support military information and developments. Infra-red scene simulations are used in the development and evaluation of electro-optical systems such as infrared missiles and the countermeasures against these missiles.

The Optronics Scene Simulator (OSSIM) is an engineering tool that utilises computer-based simulation and models to create synthetic images of scenes in the visual and infra-red bands [97]. These images are used for radiometry research and image processing algorithm development. The systems create models of possible real-world scenes, complete with target and threat objects, terrain data and information about the atmosphere.

OSSIM covers a wide spectrum of applications. Some examples of these are scenario analysis and theoretical concept studies, infra-red measurements and signature modelling, hardware development and support for flight tests. The most important functions are to:

- Assist in the design and development of optronic sensor systems that are part of countermeasure systems or thermal imaging cameras by providing performance evaluation functions.
- Assist in preparation of military flight missions by running prepared flights in simulation. Running the flight tests in simulation provides a way to identify problem areas prior to real world testing, thus saving time.
- Gain experience with building models and testing them against complexities in the simulation to ensure a better understanding of the complexities of the real world domain.
- Eliminate the need for extensive field tests in certain cases by providing flexibility in what can be simulated.

This research project involves only one application of OSSIM that supports evaluation of countermeasures for aircraft protection, called self-protection. This application assists in studies to determine the effectiveness of countermeasures on military aircraft. It runs as an application within a bigger environment, linking different models that interact. Figure 4.1 illustrates how the application fits inside OSSIM.

The self-protection application is used to perform the following functions:

- Assist in field trials by providing input to the planning of the field trials. The use of models and simulation minimise the necessity of field tests for every possible scenario. Field tests are very expensive and missiles cannot be fired at own

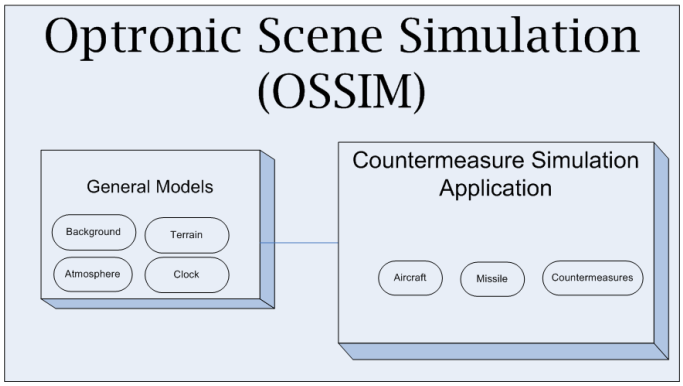

Figure 4.1: The Simulation Environment

aircraft.

- Influence military knowledge and strategy for specific scenarios.
- Act as a knowledge repository.
- Evaluate countermeasure effectiveness for aircraft protection.

The main function of the self-protection application is to perform an evaluation of countermeasures implements on military aircraft. Military aircraft implements flares as countermeasures but, to ensure effective protection for the aircraft, the performance of these flares must be evaluated. The evaluation studies can point out necessary improvements and assist further development of the countermeasures. Doing the evaluation studies by simulation has many advantages. Field trails are expensive and time-consuming to do and it is not possible to perform certain tests in the field. Simulations make it possible to predict what is going to happen in the field trial. Field trials that are not going to add any value to the countermeasure study can be eliminated.

The next section describes a simulation scenario in an evaluation study. This is the most important part of the simulation and if not set up correctly, leads to inaccurate or erroneous results.

4.1.1 A Simulation Scenario

A single evaluation study consists of specific aircraft and countermeasures, depending on the need of the client. OSSIM simulates such a real-world scenario by using software models of all the objects that are part of the simulation scene. Models of every component are built and implemented in the system. The aircraft, missiles and countermeasures are presented by different models in the system. Model parameters describe the characteristics of each model. By simulating their behaviour and interaction in a specific scenario, a prediction can be made as to the effectiveness of the

countermeasure against the missile attack.

A simulation study is performed by following these steps:

1. Determine the user requirements and the purpose of the study.
2. Set up simulation files and model parameters.
3. Run the simulations.
4. Analyse and present results in a report.

From the initial requirements, a simulation is set up by specifying which models are going to be used. A Graphical User Interface (GUI) guides the user to set up a simulation by providing a list of possible models (See Section 4.1.2) for the type or aircraft and missile. The parameters for every model must be set if default values are not going to be used. The user selects the models and sets the necessary parameters. The GUI creates the simulation input files from the specified models. These files describe the simulation scene. A scene in the simulation is a virtual world containing all the elements of a military scenario as specified.

4.1.2 The Simulation Models

The purpose of the simulation system is to model a scenario in the real world and in the case of a countermeasure evaluation study, several models are used to achieve that. It is accepted that a model will not perform in exactly the same manner as the actual real-world object but the purpose of models is to assess how a real-world object will most likely react to different engagement scenarios [11]. Thus a model will never be an exact presentation of the real world, but in order to closely achieve this, models of a countermeasure scenario are built by using models for the different components taking part in a scenario. This section describes the models that are used in the self-protection application.

The models are implemented in different ways, depending on the type of model and its purpose in the simulation. Aircraft models are developed by measuring their spectral properties and building a model from their physical attributes, the spectral properties and behaviour. The threat models, or missiles, are built by exploiting the missiles and building models to simulate the way they work. The countermeasure models, such as flares, are built by measuring their physical values and modelling their behaviour determined by these measurements. The missiles operates by identifying targets in different spectral bands, and therefore the atmosphere can possibly play a role in what the missiles observe. The terrain models present the physical terrain where the scenario takes place and interacts with objects on the ground.

The following is a list of models in a simulation scenario:

- Aircraft: Geometric and behaviour properties are modelled. A wireframe of the aircraft (Figure 4.2) is used to present the geometric properties of the aircraft.
- Missiles: An accurate model of the way the missile behaves and track targets. An image of the view from the missile can be seen in Figure 4.3.
- Countermeasures: Flares are modelled with the correct temperature and physical properties.
- Atmospheric conditions. Smoke, fog or pollution are modelled to improve simulation fidelity and to create a more realistic scene.
- Terrain: Model of the physical terrain where-in the scene is set.
- Clock: The clock controls the simulation time.

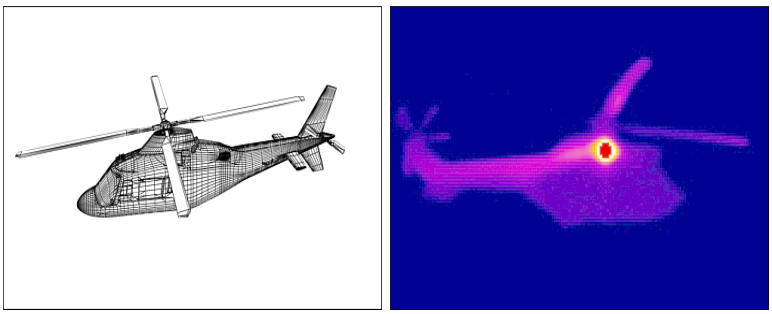

Figure 4.2: Example of Aircraft Models

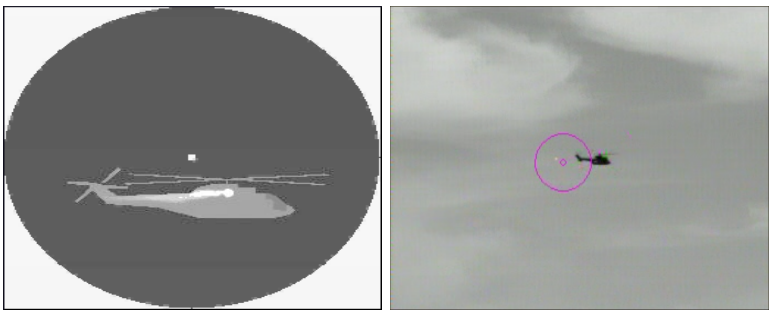

Figure 4.3: The Missile View

The models have different levels of detail. The atmospheric models are for example modelled with an external program controlled by a set of input parameters but the aircraft models are modelled by internal code. One of the main characteristics is the ability of the system to execute models at different levels of detail. The implementation detail of the models is not known to users and this can pose the a problem of knowing the level of detail a model is implemented at. Those users only interacting with the system at a higher level do not have technical insight into the models but need to know what detail are available in the system.

All these models described above are used to create a scene, called the simulation scenario.

4.1.3 The Simulation Scenario

The main concept in the simulation is the *simulation scenario*. The scenario is therefore the root of a simulation run. A scenario describes what is in the simulation run, and specifies the models that are part of the simulation run.

A scenario is described by a set of XML files that forms a hierarchical tree. Figure 4.4 illustrates the configuration of the files, starting from the root file. A scenario starts with a scenario root file with parameters linking in additional files to describe certain parts of the model. A model in the scenario can possibly consist of only one file, for example the Clock model. In other cases, a model is described by several files. These files contain all the parameters describing the use of the model in the specific scenario.

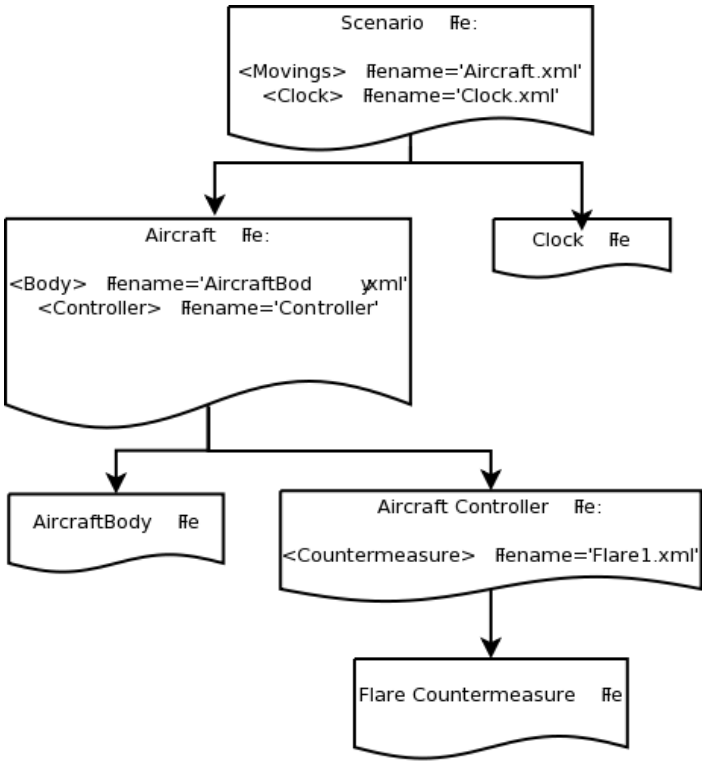

Figure 4.4: Relations Among Files in a Scenario

To get answers to specific evaluation studies, the scenario must be set up with the appropriate models and parameters. The simulation is a complex environment and a certain amount of domain knowledge is necessary to be able to set up a meaningful simulation. After the simulation is set up, it is run in the simulation engine. After the simulation run is complete, results can be obtained from the output of the run.

The presentation of the results depends on the type of evaluation study that was performed.

4.1.4 The Simulation Results

The results of a simulation is saved in XML files that can be manipulated in various ways to present the results. One such output is to play the scenario in a 3D viewer, using the information of how the models interact during the simulation run. The user can see exactly what happened during the simulation and why models behave in a certain way. An example of an image taken while playing the viewer is displayed in Figure 4.5.

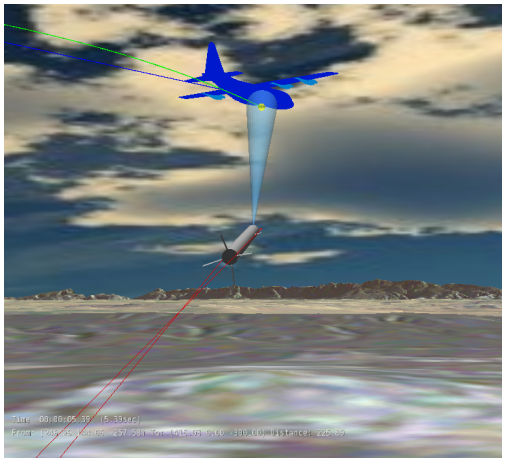

Figure 4.5: Example of Visual Output Created from Simulation Results

The previous sections describe what happens during a simulation run and how a simulation is set up. Although the simulation system is successfully used to assist in evaluation studies, there are issues and shortcomings that, when addressed, will improve the use of the simulation system. The following section discusses the issues and concerns and the requirements for improvement.

4.2 The Concerns and Issues in the Simulation Environment

There are several aspects of the simulation environment that can be improved to make it easier to use and to enhance the value of the evaluation studies. The process to identify and list those issues will answer sub-question two, namely to determine the requirements or concerns of a countermeasure simulation system environment that ontology technologies support.

There are many different people involved in the system and the simulation environment. For example, the client does not always know what models already exist, to what level the models were constructed, and the scenarios that might be possible

in the simulation. Figure 4.6 shows how many different people could possibly be involved in the simulation environment.

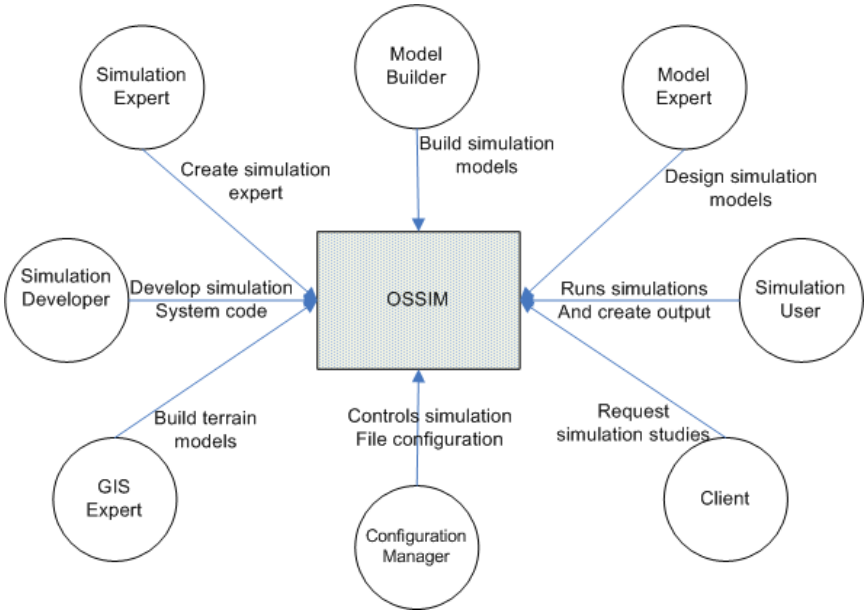

Figure 4.6: People Involved in Simulation System Environment (adopted from Willers [97])

The **Simulation Expert** has knowledge that must ideally be captured in a knowledge base so that it can be shared across the domain.

The **Model Builder**'s task is to build new models and expand existing models according to input from the experts.

The **Model Expert** is a technical person with detail knowledge on how to build simulation models.

The **Simulation Users** build scenarios, run simulations, create reports and interact with the clients.

The **Clients** request specific simulation studies according to their need.

The **Configuration Manager** controls the system configuration and model versions.

The **GIS Expert** builds terrain models. Terrain models describe objects on the ground and the interaction of objects in the scene.

The **Simulation Developer** develops simulation code to perform certain functions.

One person can fulfil more than one role and coincidentally might have the appropriate knowledge to perform the task at hand. The more likely case is that different role players must work together to perform a simulation study. As illustrated in Figure 4.6, system experts, model builders, developers, clients and users are all involved in the system in a different way. There are inconsistencies in the terminology used

between the different people involved. A common vocabulary is unlikely and this often leads to frustration and the incorrect use of concepts. The first issue in the simulation environment is therefore that of these different terms used for different concepts. There is no agreement between developers, users and clients as to which term is to be used for which concepts. This can lead to poor communication between different users. Missile models can for example be called missile, threats or seekers. This problem of sharing the same language and understanding is deemed to be necessary in order to create a common, shared understanding of terms among all possible role players. The first concern is thus to have a **common, shared vocabulary** to be used by everyone involved in a simulation study.

The first step to set up a simulation study is to create scenarios by choosing instances of models and setting the properties of those models. Although captured in documentation, the information for each model must be available during the process to define the models. The information of the models also needs to be available in such a format that it can be used to validate the simulation files by supporting the users that compile scenario files for a specific simulation as well as the software modules that verify the scenario files. There is therefore a **need to know what models are available** in the simulation environment.

The simulation environment contains information on a number of scenarios and previous simulation studies. The ability to describe a simulation scenario on a high-level, using consistent terminology, will greatly improve the content of the simulation environment. There is therefore **a need for a high-level description of a scenario** that provides a human-friendly way to view scenario detail.

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. The system and model repository are consistently being updated by improving existing models and by adding new models. Minimum requirements exist for a model to be operational in the system and it is important that these models adhere to existing standards and rules. A client, wanting to develop its own model for integration into the simulation system, needs a set of rules to adhere to and a way to have access to those requirements in a meaningful way in the form of **guidelines for the specification of new models**

Models behave and interact in a certain way and although it might be possible to set up a simulation that produces no runtime errors while the simulation is executing, it might not be correct according to the rules of how the models behave. The specified behaviour might be unrealistic or even impossible. Consequently there is a need for a function to **verify and validate model interaction** in a scenario.

The scenario input files of previously run simulations are stored files and the results are written up in reports. There is not an existing solution that shows, on a high level, what previous scenarios were run. A method is needed to construct descriptions of previously run simulations by **reverse engineering** the previously run simulations.

Table 4.1 is a summary of the issues and concerns in the simulation environment and suggestions of what the necessary functions are that will offer a solution to a specific issues.

Concern	**Requirement**
Different users use different terms	A common. shared language
Model parameters not available Guidelines to add new models	Having a knowledge base for the domain
Difficult to understand scenario	High-level description of scenarios
Scenario not correctly set up	Validation of scenario

Table 4.1: Concerns and Suggestions in the Simulation Environment

The second part of the awareness phase of design research is to suggest a possible solution that can address these issues and a solution that will provide the functions as listed in Table 4.1. The next section discusses the requirements of a possible solution.

4.3 Suggesting a Solution

Suggesting a solution is not a mere question of 'How to solve the problem?', but entails a process of looking at the issues that describe the problem, and finding something that promise a possible solution. This can be an application, a product or a new way to perform certain tasks. The first step to find such a solution is to list the requirements that it must adhere to.

4.3.1 Requirements of the Solutions

The list of requirements were drawn up to reflect the issues and concerns. The first requirement is to have a common vocabulary to share amongst everyone involved in the system environment in some way, developer or user. There is a lack of such a common technical vocabulary to describe specific concepts and often different terminology are used to describe the same concept. A single source of definitions describing each concept and a shared language to be used by the members in the team will improve communication.

The success of a simulation depends firstly on how good the models are. If the

behaviour of models is not implemented or used correctly, the simulation results will not be accurate and the simulation results not usable. New models are also constantly being added to the simulation system by different clients. It is important that these models adhere to existing standards and rules. If a repository of models and parameters is available it can be used by clients who want to develop their own models. The integration of new models into the simulation system will be easier if the developers of these new models have a guideline to adhere to. Furthermore, while setting up simulations, all the model parameters may not be readily available. The simulation practitioner performing the simulation set-up might not have specialist knowledge of how the models interact, and may set up scenarios that are syntactically correct but do not make sense in the real world.

A repository of descriptions of previously run simulations is needed. Such a repository will provide a high-level description of a specific scenario in a consistent terminology. A method to validate how the models are set up in a scenario will ensure that simulations run in the system have the correct syntax and meaning.

The solution has to be expandable to cater for any future developments. It must be able to add for example new models of aircraft and missiles or expand the parameters of a current model.

In the previous chapter, it was discussed how possible solutions were investigated in the past. Several possibilities were investigated such as IDEF [57]. From the research it emerged that technology such as ontologies should be investigated further.

4.3.2 Ontology as Suggested Solution

In Section 2.3 and Section 2.5, ontologies are described and examples given of ontologies used in computer science and military systems. Ontologies in computer science are defined by Zúñiga [100] as an ontology specifically designed for a domain to present the objects and the meaning of the objects in that domain in a formal vocabulary.

According to Noy *et al.* [62] and combined with the statements of McGguinness [56], ontologies can possibly offer the following for this domain:

- A shared, common vocabulary describing the concepts in the simulation environment. Domain users can use the same language to describe concepts and a list of models and description of the models in the domain are available.
- The information captured and contained in the ontology can be re-used by defining previous simulation scenarios and used them in subsequent simulations.

- The documents describing a specific concept can be made available as part of the definition of that concept in the ontology. This provides easy access to, for example, user manuals or design descriptions.
- Browsing support is offered by providing a list of scenarios or specific models, navigation around the concepts gives access to those concepts.
- Search functions can be customised by setting specific queries to search for specific concepts, for example, finding all scenarios using a specific aircraft model.
- The ability to discover ambiguities in the information. Rules can be set to define interaction between models.
- The ability to check for inconsistency between concepts. In this domain, inconsistent use of specific types of countermeasures protecting aircraft can be checked.
- Restrictions can be applied. Aircraft have specific behaviour that can be specified in the ontology.
- Concepts only need to be partially defined. If a new aircraft model is added to the ontology, and it is defined as an aircraft, the ontology will classify it correctly.
- Interoperability support. In this application, the ontology can interact with a graphical user interface to provide information stored in the ontology.
- Integration of information or processes. The ontology will be an integral part of the process to build a scenario for a specific simulation.
- Validation and verification are supported. The correct use of models in a scenario can be validated.
- Configuration support by controlling the addition of new models and scenarios and effectively organise domain information.

Section 4.2 listed shortcomings of the simulation environment as well as defining some requirements that exist, summing it all up in the requirements for a solution. Comparing the offerings of ontology and what it provides, there is enough evidence to suggest ontology as a possible solution. Constructing an ontology is thus the proposed solution to the issues and shortcomings of the simulation environment. Consequently the following research questions were formulated:

- Sub-question one: What are the requirements or concerns of a countermeasure simulation system environment that ontology technologies support?
- Sub-question two: How can an ontology be constructed that will capture the

knowledge of a countermeasure simulation system environment?

- Sub-question three: Does the use of ontology technologies support and enhance the functions of the simulation system in the domain?

These research questions need to be answered in order to make conclusive statements about the main research question: ***How can ontology technologies be used to support a countermeasure simulation system environment?***. The following step is to design, develop, use and evaluate an ontology to decide if it indeed provides the promised solution.

4.4 Summary

This chapter provides an overview of the simulation system environment. The goal was to describe how the awareness of the problem, the first stage in the design research process as described by Vaishnavi *et al.* [92], came about.

The simulation system is an application used in an environment to perform countermeasure simulation studies. Scenarios are prepared with aircraft, missiles and countermeasure models in specific conditions. The scenario is set up by specifying models and model parameters which are then run in the simulation engine.

This awareness sets the background for making a list of requirements that a possible solution must adhere to. The list of requirements and the suggestions on what will satisfy those requirements are summed up in Table 4.1. The claim was made that an ontology will provide a solution to some of the issues in the simulation system environment. This chapter describes the requirements and concerns in the countermeasure simulation environment. A suggestion was made that ontology technologies will support the simulation environment. To be able to validate this claim and to determine if it indeed provides a solution, an ontology must be constructed. The next chapter describes the development process that was followed in order to construct a working solution and to achieve a point where an evaluation of the implemented solution becomes possible.

CHAPTER 5

Development

This chapter focusses on the construction and use of the ontology in the simulation environment. Every step in the development process of the ontology is discussed.

The design research process consists of several stages as described in Section 3.3. This chapter describes stage four of the process, the development stage. The development stage answers the second research sub-question namely to determine how an ontology be constructed that will capture the knowledge of a countermeasure simulation system environment. According to Bergman [10], an artefact is created or constructed during the development stage of the design research process. A proper system development methodology is applied that entails a series of actions or steps to follow, the aim being to reach a point where the artefact can be evaluated, as described by Oates [63]. The different outputs of design research are summed up in Section 3.4. Figure 5.1 shows where the development stage of the artefact fits into the design research process.

In Chapter 3, it was stated that the choice of development methodology is influenced mainly by the type of artefact that is going to be created [92]. The literature study that was conducted together with the first phase of the design research process, awareness and suggestions, concluded that an ontology be created as the chosen artefact. In Section 3.3.3, the adaptive methodology process, as described by Bergman [10], was discussed. Adaptive methodology focusses on the development of domain ontologies to suit the nature of the environment.

The design research process is divided into stages and the development stage of the process is further divided into steps. The steps proposed by Bergman [10] in applying the adaptive methodology process are as follows:

1. Determine the scope by analysing the domain and deciding if the entire domain is going to be covered and if not, which part of the domain is going to be covered and what is going to be excluded.
2. Use, study and investigate existing sources such as manuals and technical documents.

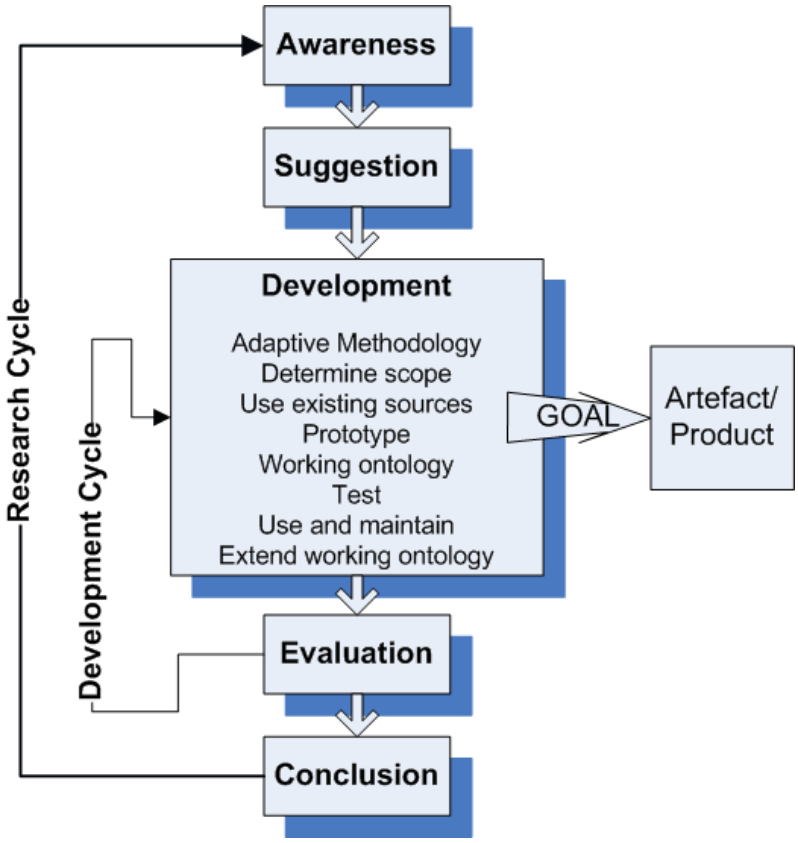

Figure 5.1: Development Process inside Design Research Process

3. Construct a prototype ontology from a selective part of the domain.
4. Construct a working ontology by expanding the prototype ontology.
5. Test the structure of the ontology.
6. Use, maintain and document the ontology.
7. Revise and extend the working ontology and repeat the process by testing and using the new additions.

After constructing a prototype, there is an ontology, although not complete, that contains a subset of the concepts in the domain. Constructing the working ontology produces an ontology that contains all information as determined by the scope. Extending the working ontology includes testing any new additions at every iteration.

5.1 Scope of the Ontology

The first step in the development process, as shown in Figure 5.2, is to determine the scope of the ontology, ensuring that the development effort stays focused [62] towards the end goal.

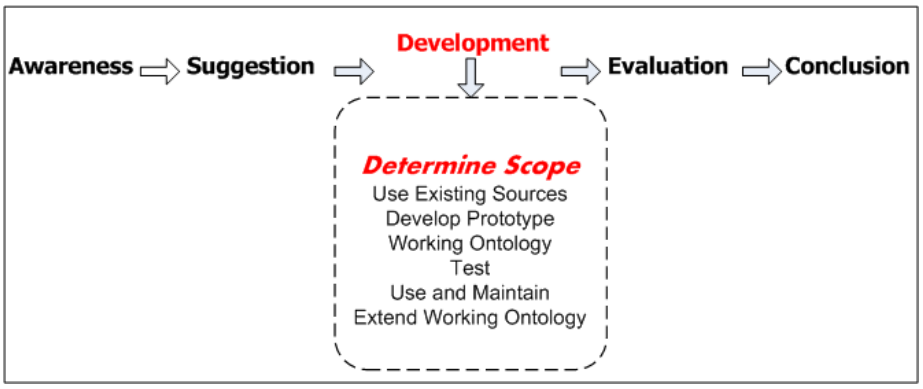

Figure 5.2: Development Process: Determine the Scope of the Ontology

To determine the scope of the ontology, the domain is investigated. The self-protection application runs in a broader simulation environment. It incorporates libraries from this environment and shares models with other applications inside this environment, as described in Chapter 4. The scope and extent of the ontology that are going to be constructed are determined by two factors:

- The ontology only needs to support the self-protection application of OSSIM and will thus be restricted to containing **concepts used in the self-protection application only**. For this research process, the remaining concepts inside OSSIM are excluded.
- The main purpose of the ontology is to **improve the understanding of the concepts in the countermeasure simulations**, therefore the behaviour and properties of all models in the self-protection application that performs the countermeasure simulations must be captured in the ontology.

In order to support the application, the following concepts must be included in the ontology:

- Scenarios - A scenario contain all the models and parameters for a specific simulation run.
- Aircraft - All aircraft that can possibly be used in simulations run in this application. Classes for the AircraftBody and the Aircraft controller are also part of the scope.
- Observers - That is the group of missiles that are used in the evaluation of the countermeasures.
- Countermeasures - The specific countermeasures that are evaluated by this application as well as the classes controlling the countermeasures, for example the missile approach warner.
- Environment - That is the terrain, atmosphere and background models included

in the simulations.

- Clock - The clock concepts determine the time and duration of the simulation run.

The scope of the ontology must guide the developers of the ontology throughout the whole process. After determining the scope of the ontology, the following step is to investigate the documents and other sources that are available in the environment that will assist in identifying the concepts and relations to be contained in the ontology.

5.2 Use Existing Sources

The developers of the adaptive methodology emphasises the use of existing information, referred to as structures or sources, in the domain [10]. Figure 5.3 illustrates where in the development process the step of using these existing sources occurs.

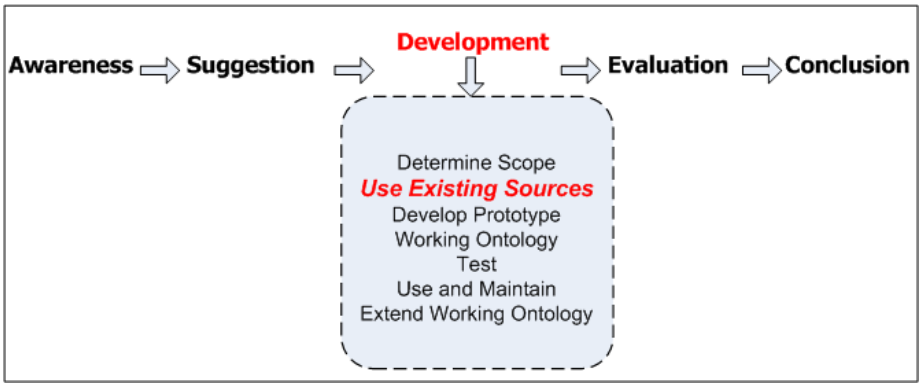

Figure 5.3: Development Process: Use of Existing Sources

These documents, reports and other sources of information contain existing and current domain knowledge. Information can be obtained from these sources to assist in defining concepts and properties for the construction of the ontology. There are several documents, structures and sources available in the OSSIM simulation environment that can be used to assist in making a list of the concepts in the simulation. Table 5.1 is a summary of the existing documents used in the design and development process.

Once these documents are identified, they are analysed and investigated. From this investigation a list of concepts is defined. This is not the final list of concepts but it does provide a starting point to build upon. These documents are investigated not only for concepts of the ontology but also for the description of the concepts as well as the way the concepts interact with each other. The following example illustrates

Name of Source	Description
Installation Guide	This document contains instructions on setting up and building the simulation environment on a new computer.
Modelling Report	A description of the models in the self-protection application. It includes equations, pictures, tables, graphs and in some cases scripts and data file formats describing the models.
OSSIM White Paper	A technical document that describes the simulation environment and its uses.
Software Manual	The software manual contains information on building properties of concepts and indicates how to build models.
Test Scenarios and Procedures	There are a set of test points in the system and all of the test points are documented. Each test point is described in detail, with a summary of the expected outcome of the test point.
User Guide	The user guide provides the user with information on how to use the system. This file contains a description of all the file formats for the different models and guidelines on how to execute the simulation test points.
Technical Reports	Simulation studies are documented in technical reports delivered to clients. These reports describe the simulation scenarios and how the models were used.
Configuration Files	Configure simulation and models.
Source Code	The source code contains valid property and parameter values.

Table 5.1: Summary of Supporting Documents Used as Sources

how one of these documents is used to extract information. This is an extract from the User Guide describing the Scenario concept:

> *The (flat ASCII) scenario file describes the simulation setup, moving object files and observer files. From the files specified here, all other files are loaded. Depending on the settings in the scenario file, a number of supplementary files are also created.*

The typical scenario file, as described above, is illustrated in Figure 5.4.

By analysing the descriptions from the user guide and the file that describe the contents of a scenario, detail about a scenario is obtained. The information indicates a

ScenarioFile.xml

```
<?xml version="1.0" ?>
<Scenario Name="Task05">
  <Clock Filename="Clock.xml" />
  <Observers>
    <Observer Type="Missile" Filename="Missile_A.xml"/>
  </Observers>
  <Movings>
    <Moving Type="Heli" Filename="Puma.xml"/>
  </Movings>
  <GroundAltASL Height="0.000000e+000" />
  <LatLongLocation Latitude="-2.544000e+001" Longitude="-2.811000e+001" COMMENT="
  <Atmosphere Filename="Atmo.xml" />
  <ErrorHandler ExitOnFatal="true" />
</Scenario>
```

Figure 5.4: Scenario File in Simulation

concept named **Scenario**, that is the root of a simulation run. It describes information on what a scenario file can contain and how, from the scenario file, the other data files can be accessed, almost like a tree structure. By studying the scenario file description, the following detail, describing mandatory and optional objects in a scenario, are obtained:

- **Scenario**: There is a root element, called Scenario. This element has attributes `Name`, `ID` and `Version`.
- **Spectral**: The Spectral element has a single attribute called `MaxColours` to define the number of spectral colours in the simulation run.
- **Clock**: The Clock element defines the clock data input file with an attribute `Filename`.
- **Observer**: The Observer element describes all the observer elements in the scenario, each of which is described in an Observer element. It is a mandatory object. Each Observer element has two attributes: Type, which defines the C++ object name and a Filename attribute that defines the observer data file.
- **Moving** The Moving element encapsulates a block of moving object elements. Each Moving element has two attributes - Type, which defines the C++ object name and a Filename attribute that defines the moving data file.
- **Atmo** The Atmo element describes the file containing the information for the atmospheric conditions in the scenario.

From the above, certain information can be obtained. There is a Scenario class that

contains properties called `Name` and `ID`. The mandatory elements are also obtained. This information can be used to make a list of concepts, starting from a Scenario class and building up the concepts of the domain.

The outcome of this step is a list of concepts, containing properties of the concepts as well as individuals that are instances of the concepts. Table 5.2 shows an extraction of the list of concepts created after analysing the sources.

Concept	**Parameters**	**Individuals**
Scenario	Name Date Clock Filename Moving Observer Atmo Filename	ScenarioPumaFly50ft ScenarioNoFlares ScenarioPuma60ftNoFlares
Clock	Date SimulationStopTime TimeOfDay	GeneralClock
Atmosphere	Model	MidLatitudeSummer
Moving	Type	Puma
Observer	Type	MissileA MissileB MissileC

Table 5.2: Extract from List of Concepts

After constructing a list of concepts in the domain, the next step is to construct a prototype ontology using the existing documents and sources and the list of concepts constructed from these sources.

5.3 The Prototype Ontology

As described in Chapter 3, the next step in the adaptive methodology development process is to build a prototype ontology. The main purpose of the prototype ontology in this research project is to demonstrate the concept of an ontology in the simulation environment. The construction of a prototype ontology is the third step in the development process, as seen in Figure 5.5.

The advantage of constructing a prototype before constructing a full working ontology is that the theory and concepts of ontologies and how they can be used inside the simulation environment can be demonstrated at an early stage. A practical example provides a better explanation to potential users than mere theory. The prototype was built inside the Protégé editor [70] and defined by using the OWL 2 ontology language [41] provided by the Protégé editor, as described in Section 2.7.2. The pro-

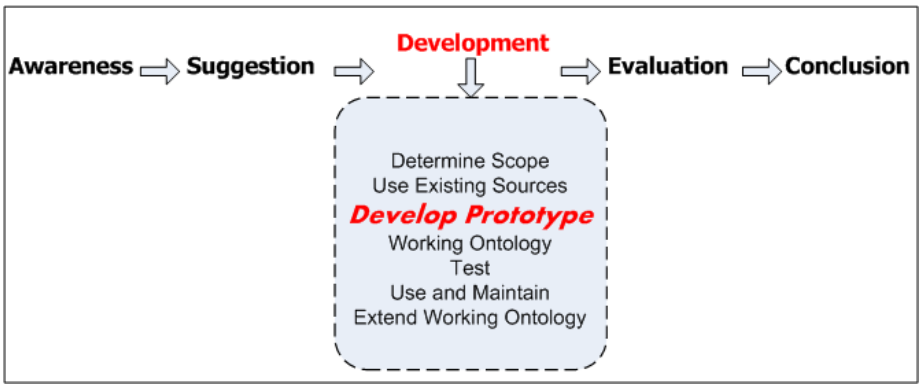

Figure 5.5: Development Process: The Prototype

totype ontology contains a subset of the concepts in the simulation system environment. The aim of the prototype is to get to a small, working example and it is therefore unnecessary to include all the concepts at this stage of the process. Although only a prototype, a methodology was followed to construct the prototype.

The methodology followed to construct the prototype, was built upon the methodology described by Noy *et al.* [62], and adapted to use the methodology described in Gerber *et al.* [30]. Figure 5.6 illustrates the steps involved in creating the prototype that are described in more detail in the following sections.

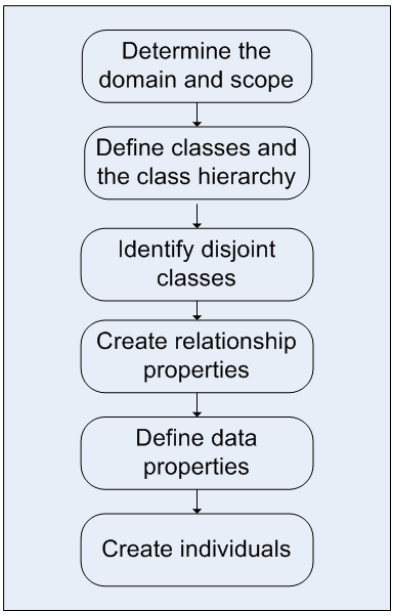

Figure 5.6: Steps in Creating the Prototype Ontology

5.3.1 Determine the Scope of the Prototype

According to Noy *et al.* [62], the important issues to consider when determining the scope of the ontology are the purpose of the ontology and the domain in which it is

going to be applied and used. The scope will also be determined by the goal of the ontology: what purpose is served by constructing the ontology and who is going to use it.

The scope of the prototype ontology covers only a subset of the concepts in the self-protection application of OSSIM. The purpose of the prototype ontology is to demonstrate that an ontology can be constructed for the simulation system and that the basic functionality available in Protégé can be applied and used. The prototype will only be used by developers to test the fitness of use and to demonstrate the theory of ontologies. It was decided to limit the scope of the prototype to a set of concepts describing one type of simulation study. The prototype must at least present a scenario in the simulation; therefore, the mandatory objects in a scenario had to be included. The limit in scope was managed by not including all the type of models. For example, only one aircraft model was included.

5.3.2 Define the Classes and the Class Hierarchy for the Prototype Ontology

The first step in the development of a prototype is to identify the concepts that are going to be used in the prototype. A class in the ontology is a concept that is used to define a group of individuals. When defining classes, the application of the ontology must be kept in mind; in this case, the application of the prototype must be kept in mind.

To identify classes for the prototype, the list of concepts drawn up in the previous step of development was used. Apart from using the list, an additional technique mentioned by Noy *et al.* [62] was used to identify concepts used in the prototype. This is to list the important concepts in the domain and use that as a starting point to define classes. The important concepts in this case are only those concepts that will be part of the prototype ontology. The classes defined for the prototype ontology are illustrated in Figure 5.7 in the section discussing the prototype ontology.

5.3.3 Identify Disjoint Classes

One of the advantages of having an ontology for a domain is to have explicit knowledge made available. This is done by making classes disjoint from each other so that two classes cannot share an individual [62]. A concept cannot be two different things at the same time. This is an example of explicit knowledge of the domain that is published by the ontology.

Disjoint classes have to be explicitly defined in the ontology because, as explained by Horridge [43], OWL classes overlap by default. An individual is by default a member of a particular class unless it is explicitly excluded from that class. An individual

of class **Terrain**, for example *Forest*, cannot be a subclass of class **Atmosphere** because a *Forest* cannot be a member of class **Terrain** and of class **Atmosphere**. If the fact that two classes are disjoint from each other is not stated in the ontology, incorrect assumptions can be made about individuals of those classes.

5.3.4 Create Relationship Properties

Class properties are defined to provide information to what is known about each class. In Section 2.7.3, the components of an OWL 2 ontology are discussed and it is explained how object properties define a link or relationship between two individuals in the ontology. Object properties were defined for the classes defined in the previous step. Examples of relations between classes in the prototype ontology is:

Moving is a set of classes describing objects that can act as targets in the scenario. **Aircraft** is defined as a member of class **Moving**, therefore **Aircraft** `is-a` **Moving**.

Domain-specific relationships were defined for the ontology. For every model that is part of a scenario, an object property was defined to create the relationship between the scenario and the model, for example between a scenario and a moving.

Scenario therefore has `hasMoving` some **Moving**, `hasObserver` some **Observer**, `hasTerrain` some **Terrain**, `hasAtmosphere` some **Atmosphere**, `hasClock` some **Clock**.

AircraftBody is a set of classes describing the features of the body of an aircraft. **Aircraft**, therefore `hasBody` some **AircrafBody**.

FlareProgram is a set of classes describing the flare dispensing from the aircraft. **FlareProgram**, therefore `hasFlare` some **Flare**.

A total of 32 object properties are defined in the ontology.

5.3.5 Define Data Properties

Data properties are data values linked to classes, giving them internal structure. Every class needs to have enough properties to define that class in adequate detail to be able to use the class successfully in a simulation. Examples of data properties defined in the prototype ontology are:

Class **Clock** has data properties `Date` (date of simulation) and `SimulationTime` (maximum time a simulation can run). Class **Scenario** has data properties `Name` (unique name of the scenario) and `Description` (description for the scenario).

Class **Aircraft** has data properties `AircraftName`, `Velocity`, `Attitude` and

`EngineSetting`.

Class **`Flare`** has data properties `FlareName`, `FlareVelocity`, `FlareEjectAngle` and `FlareEjectTime`.

In the ontology, 52 different data properties are defined to decribe the different parameters for each concept in the simulation.

5.3.6 Create Individuals

The final step in the development of the prototype ontology is to create individuals or instances of classes. One of the challenges when defining classes and instances is the decision between classifying a concept as an instance or classifying the concept as a class. Again, as in the case of determining the scope of the ontology, the purpose of the ontology as well as the application using it, act as the guideline for making such decisions. An example of this is the **`Flare`** class. A flare is a countermeasure mounted on the aircraft by attaching it to a flare pod. The flare pods are mounted at fixed positions on the aircraft, on the left, centre or right of the aircraft. There is a choice of having one class, **`Flarepod`**, and having instances of the class, *FlarepodLeft* and *FlarepodRight*. Another way to model this is to have three classes: **`Flarepod`**, **`FlarepodLeft`** and **`FlarepodRight`**. Both of these techniques are correct but in the end the way the ontology is going to be used will determine which one of these techniques must be applied.

5.3.7 Discussion of the Prototype

The prototype was constructed and used as a model to demonstrate the capabilities of an ontology for the simulation environment.

No additional applications were developed for the prototype ontology other than the functionalities provided by the Protégé plug-ins. The Protégé editor provides a rich set of functionalities in the form of plug-ins of different types for different applications. These plug-ins can be viewed on the Protégé plug-in website [70].

The following functionalities, provided by Protégé plug-ins, were used to graphically demonstrate the prototype ontology:

- OWLViz
- Ontograf

Figure 5.7 illustrates the graphical display of the classes created in the prototype, as displayed by the Protégé functionality OWLViz.

The prototype also assist in identifying challenges that may arise when constructing the working ontology. The choice of modelling an event in the ontology is such a

Figure 5.7: Ontology Classes in the Prototype

challenge. An example of an event in the simulation domain is when a missile locks onto a target. There is thus a relation between an aircraft and a missile, called lockonTarget, which models an event. According to Noy *et al.* [62], 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 poses 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 implies that a missile must lock on to a target and thus enforces a rule that prevents having missile and targets not related in a meaningful way.

Concepts in the simulation have certain roles they perform. A class can lose its role over time. A missile looking at a specific target at a specific height, has a property to indicate that it is locked onto the target. If that property changes, that role is lost. If

however, an instance of class **`Missile`**, was used in that role, it can be stored and retrieved later. This was identified in the prototype as an aspect that must be attend to during the construction of the working ontology.

In some cases, although there exists a relation between two classes, instances of those classes are not allowed to have that relationship 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.

It was possible to show that an ontology can indeed provide support to the simulation environment by developing a prototype ontology as a first step in the research project. Issues and challenges in modelling could be identified early in the research project. After successfully demonstrating the capabilities of the prototype, it was expanded into a more comprehensive ontology, as described in the next section.

5.4 The Working Ontology

The following section describes the development of the ontology from prototype to full, working ontology. Using the prototype ontology as a starting point, the same activities that were followed to create the prototype are applied when the working ontology is developed that is create classes, identify disjoint classes, define object and data properties and create instances.

Figure 5.8 shows where in the development process the working ontology is constructed.

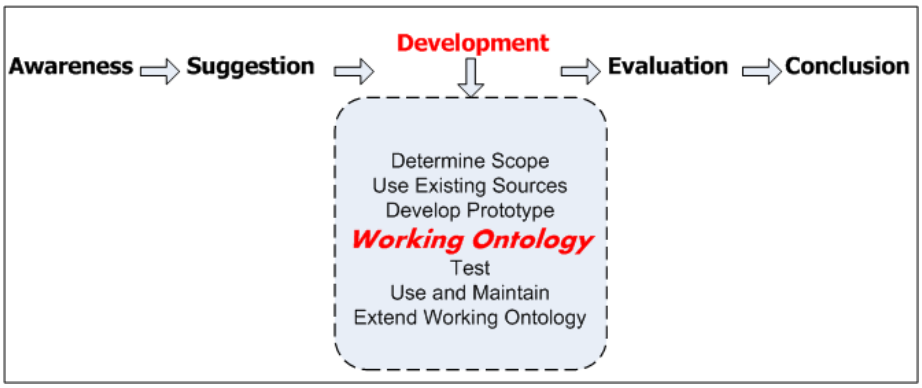

Figure 5.8: Development Process: The Working Ontology

The ontology was expanded by investigating the list of concepts defined in the second step of the development process. The concepts not included in the prototype ontology

had to be defined as a class in the working ontology. Every concept defined from the sources needs to be expanded to include all the properties necessary to create the full working ontology. The following is an extraction of the full list of classes added:

- **`Scenario`** class: The prototype ontology had only a limited set of classes describing a Scenario. In the working ontology, classes were defined for a scenario with a specific type of threat. For example, class ScenarioPumaFlying800ft, a class describing a set of scenarios all containing a specific aircraft (the Puma) flying at 800 feet.
- **`Atmosphere`** class: Includes all atmospheric models that can possibly be used in a scenario.
- **`Terrain`** class: This class was expanded to include all terrain types that can possibly be included in a scenario.
- **`Moving`** class: This class was expanded to include all objects of type moving in the simulation.
- **`Observer`** class: The observer class was expanded to include all the missiles that can possibly be used in the simulation.

The working ontology includes all the classes, properties and individuals necessary to present all the models contained in the self-protection application of the simulation environment. The classes and relations were constructed to match the classes and relations in a scenario of the simulation. The working ontology is called Simtology.

5.4.1 The Flare Class in Simtology

This section demonstrates an example of how a class was expanded from the prototype to the working ontology, Simtology. The **`Flare`** class is a class created to describe one type of countermeasure to be used against missiel threats. There are different kinds of flares and each flare can only be used in a specific context, interacting with specific models, or else the simulation will be incorrect.

In the ontology, a flare is a type of class **`Countermeasure`**, which is of class **`Moving`**. The class **`Flare`** also has several subclasses that denote the type of flares. Figure 5.9 is a graphical view of the **`Flare`** class hierarchy.

By having the class presented in the ontology as described above, it is possible to use one of the reasoners (see Section 2.7.4) installed in Protégé to validate instances of the class.

Similar to the implementation of the **`Flare`** class, all other concepts in the simulation were implemented. All the models in the simulation environment are presented in

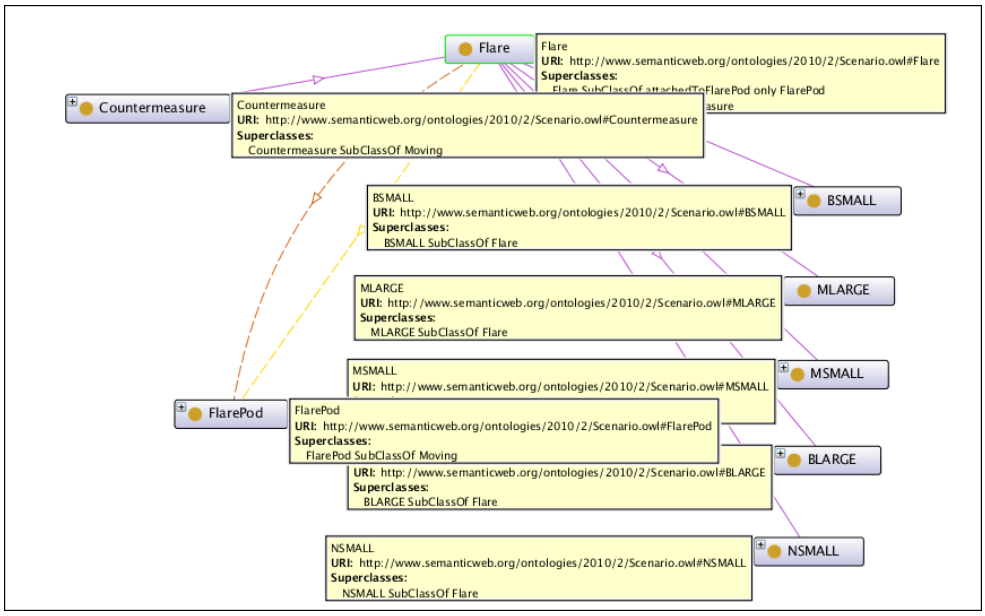

Figure 5.9: Flare Class in Simtology

the working ontology. Having all the models and their behaviour defined in the working ontology, it is possible to use it in several ways which will be discussed in the next section.

5.4.2 Functionalities in Simtology

All the concepts and properties are defined in the working ontology. Functionalities provided by the Protégé editor are immediately available. There are a number of these as listed on the Protégé website [70]. In addition to these, extra functionality were developed as part of the development phase to give the simulation environment even more support.

Applications from the Protégé editor are used in the working ontology by implementing them through plug-ins. This list only indicates current functionalities in use but will grow as new functionalities are introduced:

1. DL Query, as displayed in Figure 5.10, is the result of running a query on the ontology.

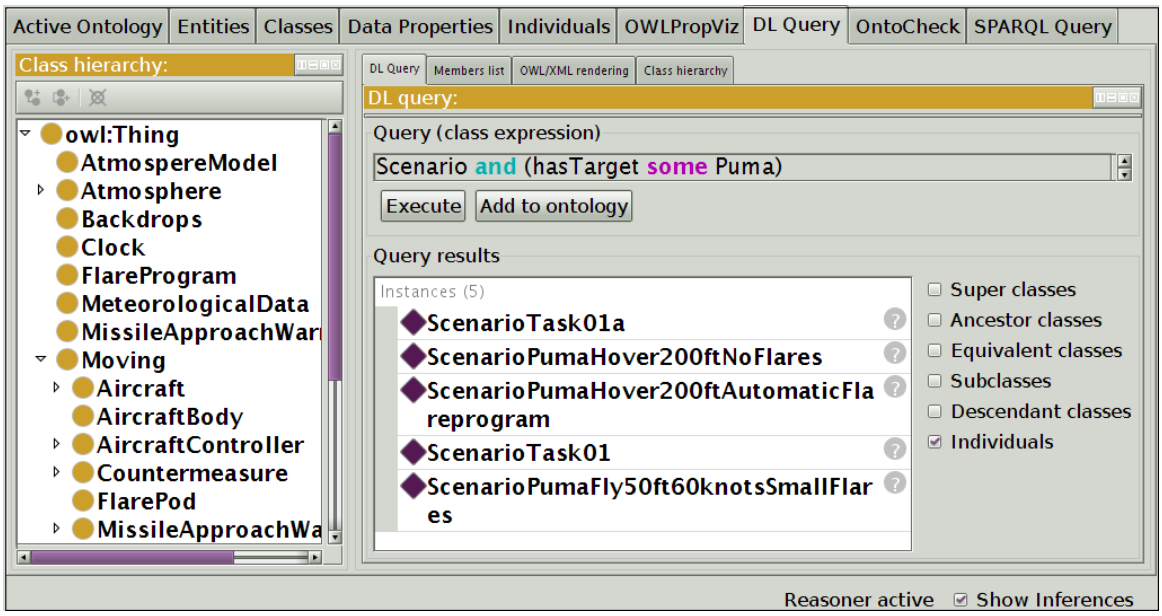

Figure 5.10: Example of DL Query

2. OWLViz is a viewer plug-in in Protégé that displays class hierarchies in the ontology. Figure 5.11 shows the class **`Moving`**.

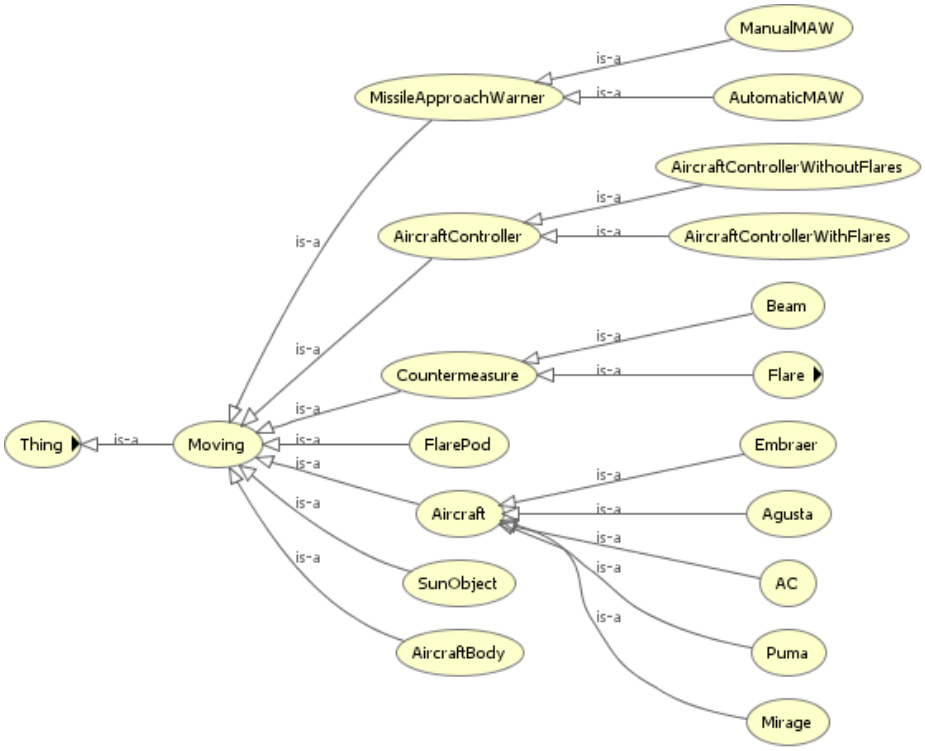

Figure 5.11: Example of Ontology Classes with OWLViz

3. OntoGraph is a plug-in that interactively navigates the relationships in the ontology. Relationships and types can be filtered to create the desired view, as in

Figure 5.12 which shows the **`Moving`** class.

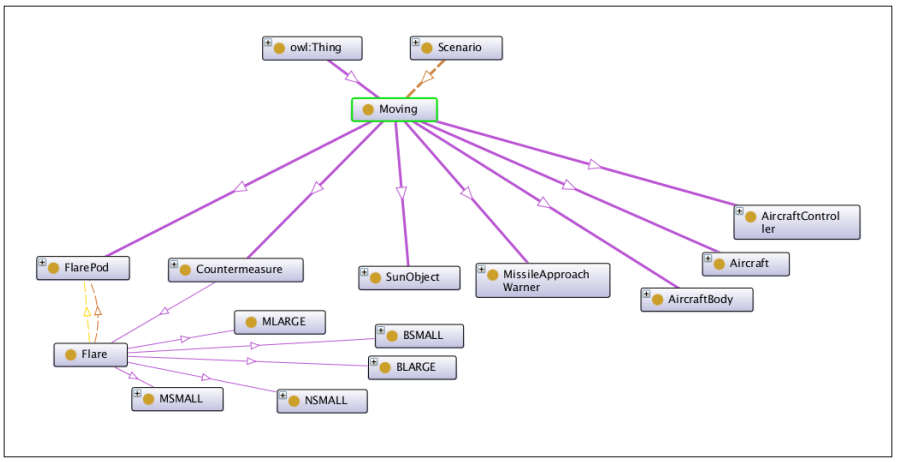

Figure 5.12: Example of OntoGraf View

4. Manchester Syntax Entity Rendering as shown in Figure 5.13 is a syntax designed for writing OWL 2 class expressions. The syntax is easy to read and write and output in this format shows a clear view of a specific individual.

```
Individual: BaselineHeli

    Types:
        owl:Thing,
        Puma

    Facts:
     hasMissileBody  BaselineMissileBody,
     hasMissileSensor  BaselineMissileSensor,
     hasMissileFuse  BaselineMissileFuse,
     AccZ  "0.0"^^xsd:string,
     AccX  "0.0"^^xsd:string,
     LookAtY  "0.0"^^xsd:string,
     LookAtZ  "-2000.0"^^xsd:string,
     LookAtX  "0.0"^^xsd:string,
     GimbalYaw  "0.0"^^xsd:string,
     YPosition  "0.0"^^xsd:string,
     XPosition  "-2000.0"^^xsd:string,
     ZPosition  "0.0"^^xsd:string,
     GimbalPitch  "0.0"^^xsd:string,
     AccY  "0.0"^^xsd:string,
     GimbalRoll  "0.0"^^xsd:string
```

Figure 5.13: Example of Manchester Syntax Entity Rendering

The following section describes the second type of functionalities, those specifically developed for Simtology and are not available through the standard Protégé application.

5.4.3 Interface with Graphical User Interface

In order to use the ontology to its full potential, additional functionalities were developed. The first of these was a function to use the information in the ontology as input to an existing graphical user interface (GUI).

In OSSIM, a scenario can be set up using different menus in a GUI. When building the scenario, models are chosen for inclusion in the scenario. The user is presented with a choice of different models implemented in the system. These options are stored in files and must be updated by the system administrator when a new model is added or an existing model is modified. The ontology is a knowledge base that already store the information. By using the ontology as input to the GUI, the information from the ontology can be exported and used to populate the list of options in the GUI. This provides the following benefits:

- No need to change the code of the GUI program when models are added as options.
- What is presented in the ontology is exactly the same as what is offered to the user of the GUI.
- In future, different languages can be used in the GUI and the ontology can provide different languages.

A script was developed that exports a selected class from the ontology to an XML file. The XML file has the following format:

```
 <?xml version="1.0" ?><object>
<terrain name="BeachFynbos"/>
<terrain name="Desert"/>
<terrain name="Forest"/>
<terrain name="Grass"/>
<terrain name="Urban"/></object>
```

The XML file is read by functions that set up the options in the GUI. In the menu of the GUI that offers the user of the GUI a choice of **Terrain** types to choose from, those options will be read from the XML file that was exported from the ontology. Figure 5.14 illustrates the process of integration with the GUI. The left of the figure shows the ontology, Simtology, as a source of information extracted by a program in a format ready for the GUI to import.

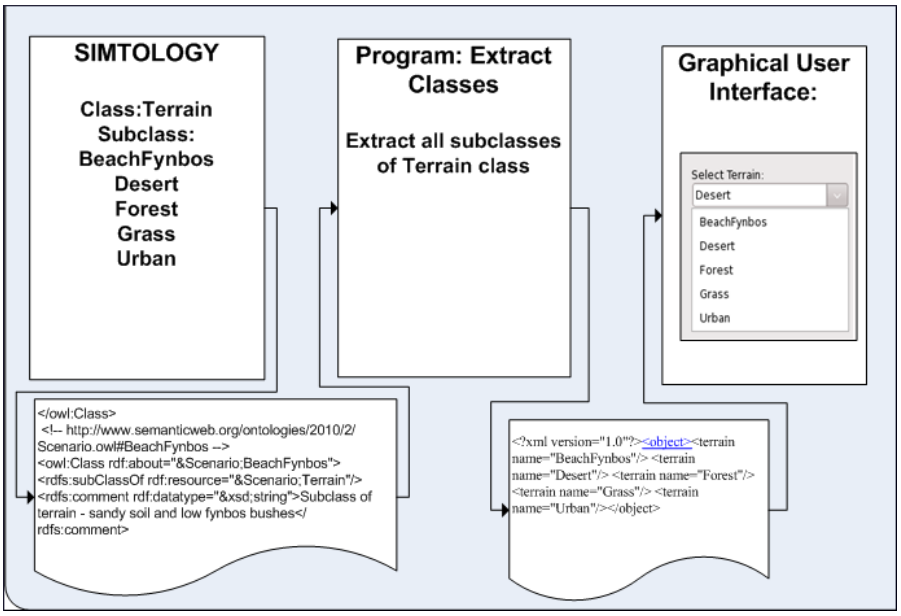

Figure 5.14: Integration of Simtology with Graphical User Interface

5.4.4 Export Descriptions of Scenarios

This section discusses the management of scenarios in Simtology. A scenario is a single simulation in the system and consists of a hierarchical set of XML files. This hierarchical set of files is difficult to comprehend and get the full picture of what the scenario contains.

A scenario is defined in the ontology by defining object and data properties. To get a description of the complete scenario, a function was developed to export the description to a user-friendly format that can be used in documentation. The function is a script that is run from inside the Protégé editor. Any individual of the class Scenario can be exported to a user-friendly view of the scenario. Figure 5.15 illustrates the screen that the user is presented with from within the editor.

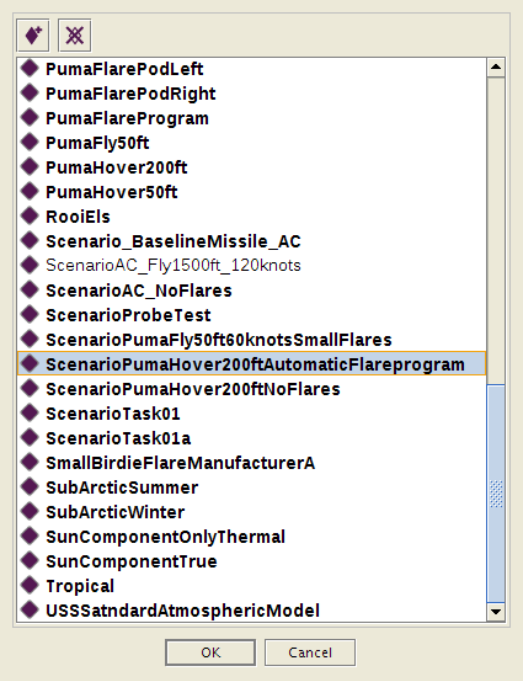

Figure 5.15: Select an Individual To Export

The user selects an individual and the complete scenario will be exported to a user-friendly format, as shown in Figure 5.16.

Scenario:
Name = TestPoint01, FileName = TestPoint01a.xml

Clock:
Name = GenClock Date: 12 February 2013, Time Of Day = 15h00
Start Time = 0ms, Stop Time = 10ms, Step Time = 1ms

Missile:
Type: MissileA
Position x=0 y=0 z=-2000
Body FileName = MissileBody.xml
Controller Filename = MissileController.xml
Fuse FileName = MissileCFuse.xml, Sensor FileName = MissileCSensor.xml, Gimbal FileName = MissileCGimbal.xml

Target:
Name = Puma
Flying at height 200 ft, speed 60 knots
Body FileName = PumaBody.xml, Controller FileName = PumaControllerwithFlares.xml

Countermeasures:
FlareSmallA on FlarePod Name: left
Ejection Azimuth = -90 Elevation = -3

FlareSmallA on FlarePod Name: right
Ejection Azimuth = 90 Elevation = -3

FlareProgram
Name = F01
FileName = TestPoint01/FlareProg01a.xml
Flare Name: f1, 0ms delay, from FlarePod=left, Velocity=35.0, Flare01.xmls
Flare NameL f2, 0.1ms delay, from FlarePod=left, Velocity 35.0, FileName: Flare01.xml

Atmosphere:
Name = Atmo1
Profiles FileName = LowtranAtmoProfiles.dat
AtmoModel FileName = MidlatitudeSummer.inp

Figure 5.16: Example of Exported Scenario Information

The previous sections describe the current functionality in Simtology. The development of the ontology is an incremental process and new functionalities will be added as needed. The different users of the simulation environment have different needs and these different needs will determine future developments.

5.5 Testing the Ontology

This section describes the testing of the ontology, Simtology, which is step number five in the development process. There are several methods of testing an ontology but they all have one goal in common: to ensure an artefact of good quality. The testing of Simtology consist of two main areas: validating the structure and contents of the ontology and testing the applications developed.

In the following figure, Figure 5.17, it is explained where in the development process the testing of the ontology takes place.

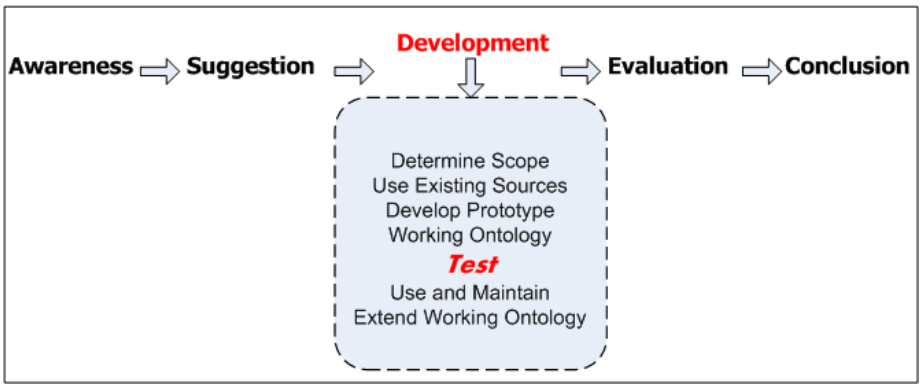

Figure 5.17: Development Process: Testing the Ontology

Gomez-Perez [32] describes the evaluation process by defining a set of questions. In testing Simtology, the questions can be answered as follows:

- Simtology tests against the definitions of the real world scenarios.
- Simtology is tested after every incremental change.
- Simtology is tested by using the reasoners in the Protégé editor to validate the structure.

The following two sections describe the two different tests that were performed: firstly to validate the structure and contents of Simtology and secondly, to test that the developed applications are working correctly.

5.5.1 Validating the Structure and Contents of the Ontology

The structure as well as the contents of the ontology must be tested. The concepts in Simtology must present objects in the simulation environment.

To test the structure and the contents, tests were performed for completeness, consistency, unsatisfiable concepts and classification results.

Completeness

Completeness refers to the extension, degree, amount or coverage to which the information in a user-independent ontology covers the information of the real world. The completeness of a definition depends on the level of detail agreed to in the whole ontology; in Simtology, completeness was reached when all the concepts in the self-protection application are included in full in the ontology.

Consistency checking

Wang, Horridge, Rector, Drummond, and Seidenberg [95] describe how ontologies can be tested for consistency by using a reasoner. The FACT++ reasoner was used to check Simtology. A given definition in the ontology is consistent if and only if the individual definition is consistent, the meanings of the formal as well as the informal definitions are consistent with the real world, as well as being consistent with each other.

Test for unsatisfiable concepts

This determines whether it is possible for a class to have any individuals. If a class is unsatisfiable, then defining an individual of that class will cause the whole ontology to be inconsistent.

Classification

The classification of Simtology was tested using the FACT++ reasoner. Figure 5.18 illustrates the output of the classification results.

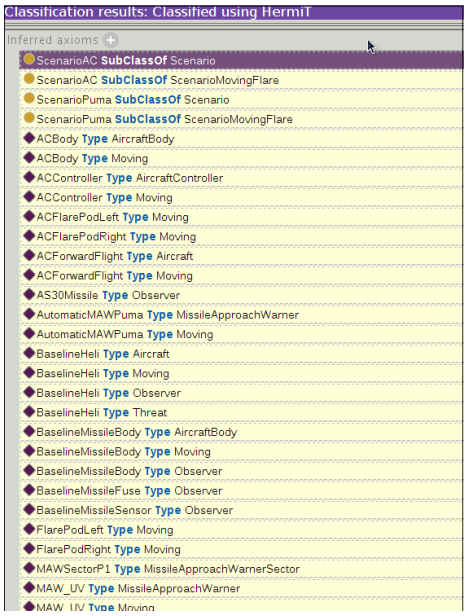

Figure 5.18: Output of Classification

The reasoner computes the subclass relations between every named class to create the complete class hierarchy. The class hierarchy can be used to answer queries such as getting all or only the direct subclasses of a class. The reasoner is also used to get a specific class that an individual belongs to or all the types for each individual. Testing the structure and the contents ensures an ontology that is validated and a true representation of the domain.

5.5.2 Testing the Applications

Two applications were developed for use in Simtology: one which is an interface to the GUI and another a tool to manage the simulation scenarios. The testing proves that the applications runs without errors and that the functionality delivers what was planned.

5.6 The Use and Maintenance of the Ontology

In order to determine if the ontology is successfully applied to the simulation environment, the ontology are used within the simulation environment as part of the countermeasure simulation studies. Figure 5.19 illustrates this step in the development process.

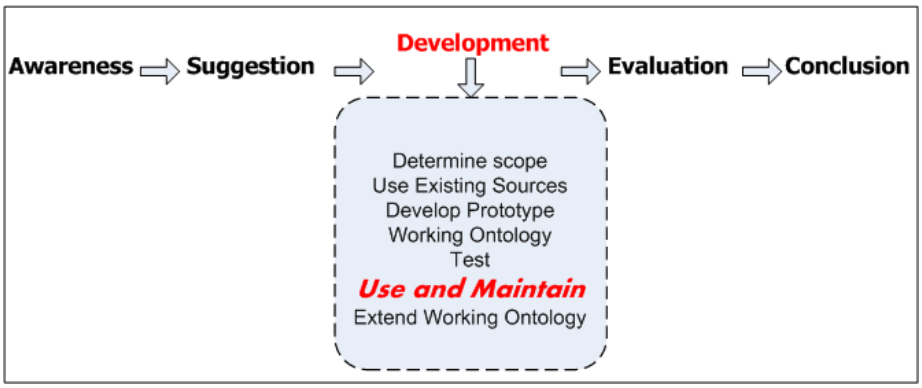

Figure 5.19: Development Process: Use and Maintain the Ontology

The ontology for the simulation environment is mainly used to communicate the concepts of a scenario and the models in the simulation. The countermeasure class is again a good example of how the information about models, which is now contained in the ontology, could be presented and discussed without going into documentation or code.

Maintenance of the ontology takes place as the result of changes arising from the following:

- Changes and additions in the simulation system.

- Changes in the ontology structure.

5.7 Extend the Working Ontology

The purpose of extending the working ontology is to have a more complete ontology that contains all the information relating to the simulation environment. As shown in Figure 5.20 , this step is the last step in the adaptive methodology development process.

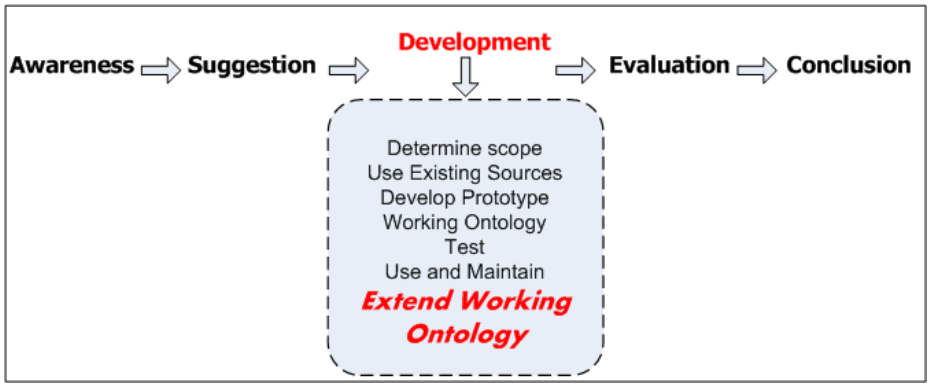

Figure 5.20: Development Process: Extend the Working Ontology

The ontology was extended by adding object and data properties to incomplete classes.

5.7.1 Future Developments

Future developments will include the automatic import of scenarios into Simtology. The intention of this is to have all previously run scenarios loaded into the ontology to form a repository (history) of simulation runs. Having all this information in a knowledge base will make it easy to determine if a specific scenario was used before and to access the related files.

5.7.2 Documentation

The ontology was documented using a Protégé plug-in called OntoDoc. Figure 5.21 shows an extract of the documentation, viewed through a standard Web browser.

The outcome of the development process as described in the previous sections is Simtology, an ontology for the simulation environment. Simtology is a complete presentation of the concepts in a countermeasure simulation study.

Figure 5.21: Output of Simtology Documentation

5.8 Summary

After ontology was put forward as a possible solution to the requirements for enhancing the simulation environment, an ontology was constructed and this chapter describes the process of constructing that ontology. A development process called adaptive methodology was applied, following the steps as described in Section 3.3.3. A prototype ontology was first constructed to act as a demonstrator. Concepts that are representative of a single simulation were chosen for presentation in the prototype ontology. The prototype ontology successfully demonstrated the use of ontology in the simulation environment.

After determining the requirements in the awareness stage, the next phase was to answer sub-question two: How can an ontology be constructed that will capture the knowledge of a countermeasure simulation system environment? This chapter describes the development process in constructing the ontology, thus addressing sub-question two. The working ontology was constructed by expanding the prototype ontology. The construction and use of the ontology is an ongoing process and for each iteration, as new functionality is added, it must be tested, used and evaluated. Existing functionality is maintained by making changes where necessary. Proper version control is also necessary to keep track of changes.

An ontology was successfully constructed and thus stage five of the design research process was concluded. In the next chapter, the evaluation of the research artefact and the construction process of the artefact are discussed.

CHAPTER 6

Evaluation of the Research Artefact and Approach

Evaluation in design research is performed when the development of the artefact reaches an acceptable level of maturity, making evaluation of the artefact viable. In this chapter, the evaluation stage is discussed in view of what the construction and use of the ontology brings to the simulation environment.

Evaluation of the research is done to determine and possibly prove the success of the research undertaken. In the case of this research project, the construction and use of the ontology in the simulation environment must be evaluated to determine the success of the process. Evaluation is the fourth stage in the design research process and follow the successful development of an artefact. In the awareness stage, certain issues are raised and suggestions made and it is during the evaluation stage that it is determined how the artefact behaves in relation to what was specified in the awareness and suggestion phases.

The ontology must therefore be evaluated by checking if the expected advantages of constructing an ontology for the simulation system, the suggestion, were indeed met and if the ontology does indeed play a supporting role for the countermeasure system. Does the ontology deliver as promised? Was new knowledge generated by designing and constructing an ontology to support the simulation system?

The reason for performing this research was to address the issues in the simulation environment by finding a possible solution to the identified issues. A research question was formulated, asking how can ontology technologies be used to support a countermeasure simulation system environment. This was further broken down into three sub questions: What are the requirements or concerns of a countermeasure simulation system environment that ontology technologies support? How can an ontology be constructed that will capture the knowledge of a countermeasure simulation system environment? Does the use of ontology technologies support and enhance the functions of the simulation system in the domain? To determine if the research process was undertaken satisfactory to answer the research questions, the development and use of the artefact must be evaluated.

Design research produces different types of output, as summed up by Vaishnavi *et al.* [92]:

- Constructs - the vocabulary of a domain.
- Models - a set of statements expressing relationships among constructs.
- Methods - an algorithm used to perform a task.
- Instantiations - the artefact in its environment.
- Better theories - improved ways to perform tasks in the domain.

The output of this research project falls into two of these categories. Firstly, an ontology was constructed for the simulation system environment, thus an instantiation of the artefact in its environment. Secondly, whilst analysing the scenarios and detail of scenario file structures, an improvement in the structure of scenario files was suggested, thus an improved way of performing tasks in the domain.

The first role of evaluation is to evaluate the artefact, the ontology, and its contribution to the simulation environment. The second role of evaluation is to evaluate the research approach to constructing the artefact. This is done according to the requirements set by Hevner *et al.* [39] as described in Section 3.6.4. The following section discusses the first part of the evaluation: to evaluate the supporting role of the ontology in the simulation environment.

6.1 Evaluating the Role of the Ontology in the Simulation Environment

The aim of the study was thus to determine, through a rigorous research process, the way ontology technologies can be used to support a countermeasure simulation system environment. The main function of this supportive role should be to assist in the understanding of the simulation system. This function was identified in the awareness stage of the design research process together with a list of requirements for a solution to be used to determine the success of the ontology. With reference to Chapter 4 a list of issues and shortcomings was identified.

- Does the ontology provide a **common shared language** to use across the user spectrum and enable consistent use of terminology?
- Does the ontology provide a way to have a **high-level description** of a scenario and a graphical view of the concepts in the domain?
- Does the ontology provide a way of **storing knowledge about the models** that are available in the system?
- Does the ontology provide **guidelines for the specification of new models**?
- Can the functionality of the ontology provide **verification and validation** of model interaction?

The following sections discuss the evaluation of the issues in the above-mentioned list.

6.1.1 A Common Shared Language

The ontology provides a common shared language to be used in the simulation environment. All the concepts in the simulation environment are presented in the ontology as a single structure. This allows the various people involved in the system to share the same terminology. A greater understanding of the terms in the simulation environment leads to better communication.

Two examples of this are the use of the term **threat** and the concept of a **flare**. A threat had a different meaning for different users of the system. Having consensus over the use of the term **threat** clarifies the meaning and use thereof. The concept of a flare was vague and many different types of flares exist. The flare class in the ontology was created in such a way that it can be used as an explanatory tool to illustrate the different flares available in the simulation as well as the use of each different flare.

Having a shared, common language provides the possibility that, in future development, software agents can be implemented.

6.1.2 A High-level Description of Scenarios

There was a need to have a high-level description of scenarios in the system. This is now provided by the ontology by having the definitions of scenarios specified in the ontology. Any user of Simtology can look at the description of any class, which present concepts in the simulation. The annotation property gives a thorough description of the concept as well as the name of the documentation file that provides more detail, for example the User Guide. By exploring the class, the properties of the class can be viewed and explained. This proved to be a better method of explaining a simulation run rather than using input files or simulation code.

6.1.3 Knowing What is Available

The simulation system environment contains information on a number of models and parameters. By providing this information in the ontology, it can be conveniently accessed. A picture can convey the same meaning as a thousand words and a visual display of the scenario immediately display the components and the relations between them. The information in the ontology is also integrated with other applications. The ontology act as a repository of models and model parameters. A function has been developed to export parameters to files that a graphical user interface can use.

6.1.4 Guidelines for Specifying New Models

When building new models for the simulation system, the model builder needs to know what the minimum requirements are that the intended models should adhere to. Certain parameters might be mandatory and without specifying them in the scenario, the simulation will not run correctly. The ontology can provide these guidelines in a meaningful way.

6.1.5 Validate and Verify Simulation Scenarios

The behaviour between the different models is implemented in the ontology. If a scenario is defined with incorrect parameters, it will not be classified by the reasoner because of the incorrect behaviour having been specified.

6.1.6 Evaluation by Practical Example

The following evaluation by example illustrates that the ontology do indeed provide the advantages mentioned in the previous sections. Firstly, Simtology provides all the simulation concepts and their interactions in the domain of the countermeasure application. The classes in Simtology are listed and can be viewed. Every class has an annotation, explaining the class and the use of the class. For example, in Figure 6.1, the scenario class annotation can be viewed.

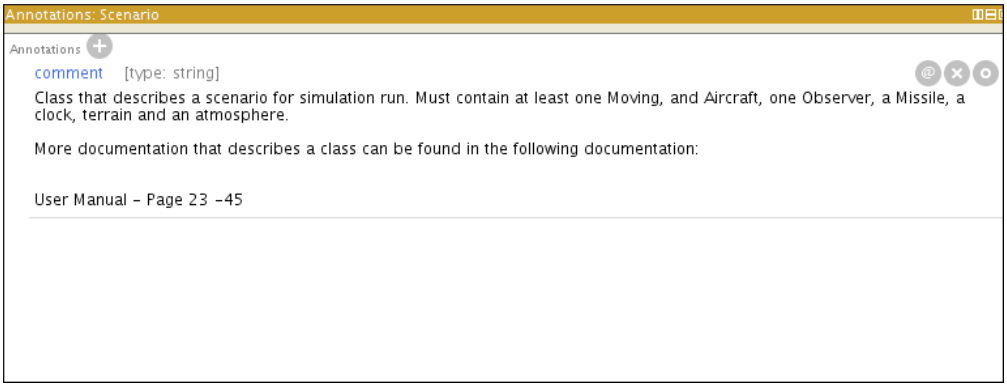

Figure 6.1: Annotation for Class Scenario

The construction a scenario can be explained and by utilising the tools available in the ontology it is possible to view the scenarios for a specific aircraft. Figure 6.2 shows the scenarios for the Puma aircraft.

Furthermore, models in the simulation can be explored in the ontology. Figure 6.3 shows the parameters necessary for a flare model.

A flare in the simulation must be modelled in a specific way. For example, a flare is attached to a flare pod and be part of a flare program. The data properties for a flare are described in the ontology. By viewing the flare model in the ontology, developers

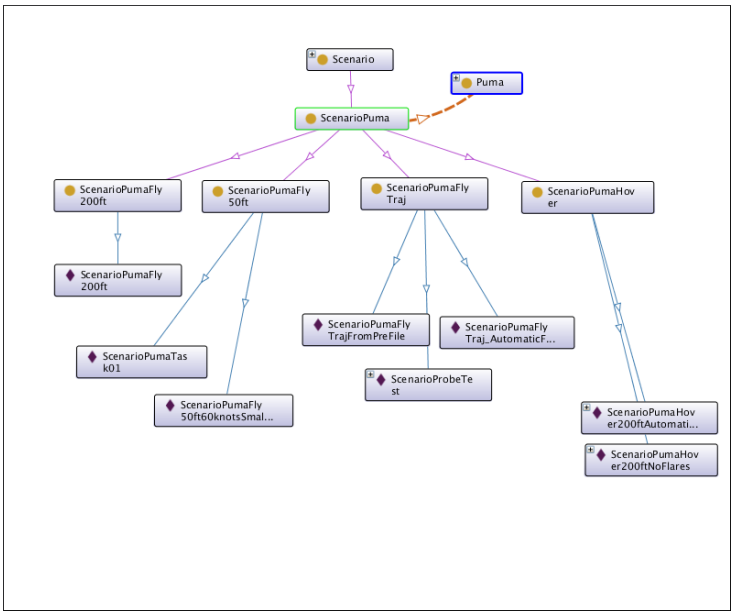

Figure 6.2: Scenarios for Puma Aircraft

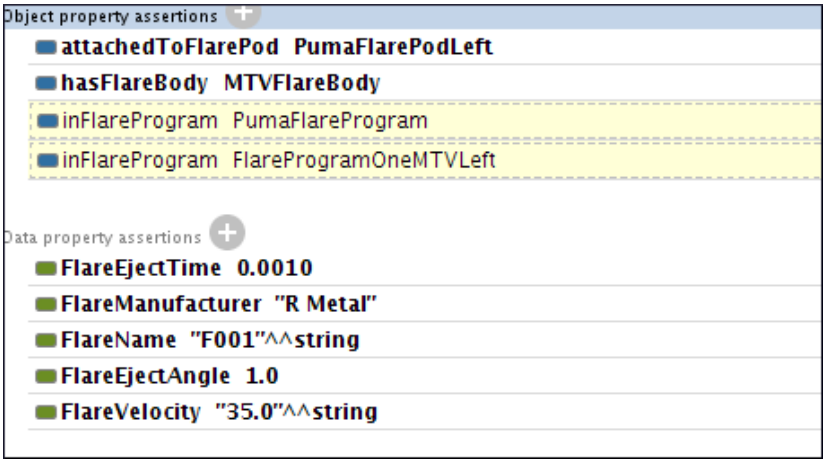

Figure 6.3: Description of a Flare

of new models can immediately see what is necessary to define a new flare in the simulation.

Once a scenario is constructed it can be validated by using the reasoner. The reasoner is active during creation and by synchronising the reasoner, the ontology can be checked. A class Puma and a class Augusta is both defined as Aircraft. There are individuals defined for them. If the individual AugustaHover is included in the ScenarioPuma, the reasoner will give a warning. Figure 6.4 shows the output from the reasoner.

The previous sections discuss the supporting role of the ontology according to the issues raised in the awareness stage of the research process. From these discussions

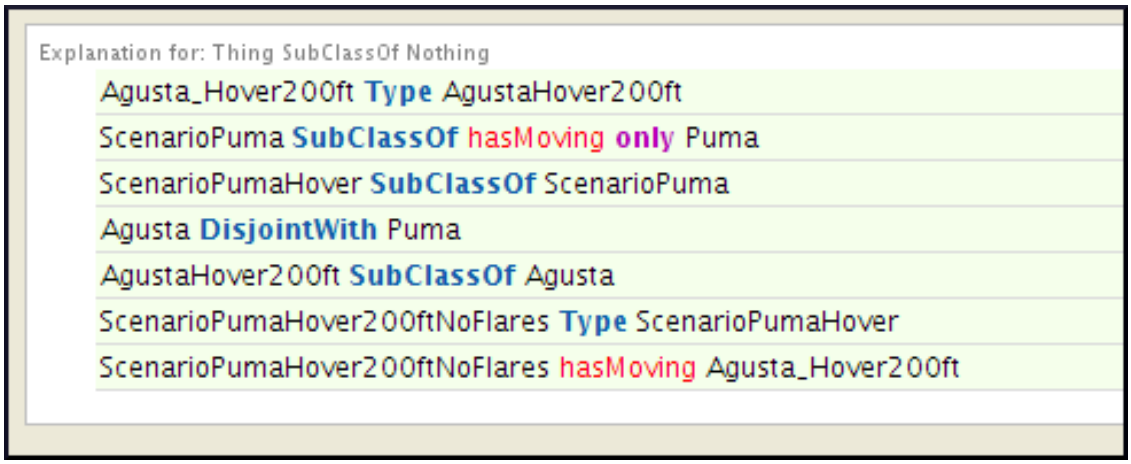

Figure 6.4: Inconsistancy in Reasoner

one can conclude that the ontology does indeed play a supporting role and that the output of the research performed produced an artefact that is purposeful. The second part of evaluation is to evaluate the research process.

6.2 Evaluation of the Research Process

To give credibility to the research and emphasise that research was done and not product development, the requirements listed by Hevner *et al.* [39] and discussed in Section 3.6.4 were applied to this research. According to the requirements, successful design research was undertaken as the following aspects were adhered to:

1. The main research question was to investigate how an ontology can be constructed to capture the knowledge of the countermeasure simulation system environment and will the use of it support the functions of the simulation system in the domain?
2. The artefact is an ontology and is presented in the OWL 2 language inside the Protégé editor.
3. Adaptive methodology was applied as a design process to build the artefact.
4. Design research and the theory of design research were applied to support the design of the ontology.
5. The ontology was evaluated after development of the working ontology. The artefact was evaluated according to the criteria set in the awareness stage.
6. The artefact was introduced into the simulation environment by firstly constructing a prototype that was expanded into a full ontology.
7. New knowledge was added to the knowledge base by contributing to the way in which ontologies are used in a military simulation environment.

8. The research successfully addressed the research question by following a design process to construct an ontology that supports the simulation environment.

If the above-listed requirements were followed, the research process can safely claim that research was done, and product development was only one phase in the research process.

6.3 Summary

In Chapter 4, the simulation system environment was discussed and background information given. In this chapter, the ontology and the construction thereof were evaluated to determine if the issues mentioned in Chapter 4 were dealt with during the development stage.

Evaluating the ontology follows after construction of the ontology for the simulation environment and is an important step in the research process. The outcome of the evaluation process indicated that the ontology does play a supporting role in the simulation environment. The initial stage of the ontology, the prototype, already demonstrated some advantages and the working ontology proved to be a solution to issues raised in the awareness of shortcomings in the use of the simulation environment.

CHAPTER 7

Research Approach Contribution

This chapter discusses the research outcomes and contributions following the construction, adoption and use of the ontology in the simulation environment.

The goal of research is to improve or to make a contribution to the body of knowledge [63]. The goal of design research in Computer Science (CS) is to solve a specific problem in a domain [92] by creating a CS product. The ontology, Simtology, was constructed by following the approach proposed by the design research process. The previous chapters discussed the development process followed in constructing the ontology and the evaluation of the role of the ontology in the simulation environment. A decision had to be reached whether the ontology was constructed in such a way that it indeed captures the knowledge of the countermeasure simulation system environment and if the use of it supports the functions of the simulation system in the domain.

This chapter discusses the research contributions made during the design and construction of the ontology and the method followed to present the concepts of a simulation in the ontology. The techniques followed to identify concepts in the domain for example, led to an investigation into what exactly are the concepts and their roles as contained in a simulation scenario. Presenting these concepts in the ontology was in some cases more successful than in other cases. This is important knowledge for future work in the simulation environment.

The research process followed a unique method of combining design research and the adaptive methodology development process. The research shown that the stages in design research can successfully applied to construct an ontology.

In Section 2.5 ontologies in military systems are discussed. In several works described in this section (for example those of Schlenoff *et al.* [77], Valente *et al.* [93], Winklerova [98] and Smart *et al.* [84]), the use and application of ontologies in military information systems were discussed. In the domain of military simulations systems, the following main uses of ontologies were identified:

- Ontologies are used to simulate trajectory information [23].
- Ontologies are used in distributed simulation environments [9].

- In a mobile route planning application, as described by Nagle *et al.* [60].
- Agents use ontologies to communicate [42].

During the evaluation phase, it was founded that Simtology provides supporting functions to the simulation environment. In the military domain, Simtology made a contribution in the categories of sharing knowledge and the integration of sources and made a unique contribution towards the military systems domain. Examples of using ontologies are scares in the military domain. This chapter concludes the research process by exploring the research outcomes and the contribution of the research to the body of knowledge.

7.1 Research Outcomes

The main research outcome for this research project was an ontology named Simtology, for the simulation system environment. The ontology was constructed, implemented and used in the simulation environment and according to the evaluation of the ontology, provides the following:

- A common, shared language and terminology to be used by all people involved in a simulation study.
- Integration with other tools in the simulation environment.
- A knowledge base for the simulation environment.
- Visual displays of all concepts in the simulation environment.
- Validation for scenarios.

The outcome of the research is the ontology but other observations made during the construction of the ontology are also important. With regards to modelling, it is important to distinguish part-of from subclass-of. An aircraft body is part of an aircraft, not part of a specific type of aircraft or subclass.

There were additional outcomes with regards to the structure of simulation files. While construction of the ontology took place, the analysis of the domain knowledge gave insight into the structure of the simulation files and how the information is linked. The process of constructing the ontology for the simulation system forces the developers of the system to evaluate the design of the simulation configuration files. The research process highlighted the importance of correctly configuring the simulation setup files.

It is important to correctly model the roles of the different classes. Modelling a missile as an *observer* in the simulation means that it can never be used in the simulation as an *object of type moving*. In Simtology, a missile can therefore never be used in a

different role.

An important modelling decision involves the modelling of individuals versus concepts. This decision has an impact on how the ontology can ultimately be used. The choice between concept and individual is often contextual and application-dependent but it needs to be evaluated in one of the development cycles.

The outcomes discussed in the previous paragraphs provide contributions to the body of knowledge.

7.2 Research Contribution

The contribution of the research in constructing the ontology resides in the domain of military information systems and more specifically in simulation system of the domain. The outcomes discussed in the previous section lead to research contributions as described in the following paragraphs.

The first contribution is better communication between the people involved because of the shared vocabulary and visual display of concepts in the ontology. By having consensus as to the meaning of all terms, communication problems are lessened.

Simtology validates model interaction of the different models in the scenario. Simtology was constructed in such a way that scenarios can be set up by creating new instances of the class **`Scenario`**. While specifying the models in the scenario, the rules that was built into Simtology makes it possible to validate model interaction.

The possibility exists that Simtology can be used as part of training material for military personnel. It is possible to explain to personnel, for example pilots, what exactly is in the system and how the real world can be tested in a simulation environment. By housing the simulation scenarios inside the ontology, it can be precisely explained how a scenario is created in the simulations as well as illustrating the effect of changing the parameters of the countermeasures and the effects thereof.

During the awareness stage of the design research process, the requirements of the countermeasure simulation system environment were identified. After taking these requirements into consideration, the ontology was constructed in the development stage. The third sub-question, that deals with whether the use of ontology technologies supports and enhances the functions of the simulation system in the domain, was discussed in the previous sections. The research outcomes and contributions indicate that the use of ontology technologies do indeed support the simulation system environment. This resolved the sub-questions and therefore successfully concluded the main research question of whether there is enough evidence to show how ontology

technologies can be used to support a countermeasure simulation system environment.

This research project also has a unique contribution in the way that the stages of design research was applied. In a recent article on design research Johannesson, Perjons, and Bider [46] explain how design research has not found maturity in research. They identify different branches of design research. One of these branches is to apply design research as a practical research methodology, as in this research project.

The awareness and suggestion stages of design research was followed as suggested in Vaishnavi *et al.* [92]. The nature of the development stage that follows the first stage depends on the artefact to be constructed. In the case of this research, the artefact, an ontology had to be constructed. After investigations it was decided to use the adaptive development methodology to construct the ontology.

The adaptive methodology for development of the ontology suited the research project in various ways. A prototype ontology was constructed first at prove to be very successful to convay the technologies of ontologies to the people involved in the system. The next step was to expand the prototype ontology to a full working ontology.

This research demonstrates in a practical way that design research and the adaptive methodology development process can successfully be applied to perform this type of research.

7.3 Summary

This chapter discussed the research outcomes and the contributions made by this research. The outcomes of the research were firstly, an ontology for the simulation system environment (as described in Chapter 4) and, secondly, several functionalities that the use of Simtology provides in the simulation environment. The ontology captures the knowledge in the domain of the simulation system and all the concepts making up a scenario are captured in the ontology. In addition, the properties and relationships between the concepts that are defined in Simtology add to the knowledge base that is provided by Simtology.

Design research was applied as research methodology. The development stage of design research was performed by applying the adaptive methodology process. The methodology followed is an unique contribution to the body of knowledge and shows that design research can be applied in this manner.

The contributions of this research project are the support that the research outcome, the ontology, provides to enhance the functions of the simulation system. Simtology provides a knowledge base for the simulation environment that enables better com-

munication among all the people involved and gives meaning to the concepts in the simulation system. The ontology was constructed in such a way that it captures the knowledge of the countermeasure simulation system environment and the use of it supports the functions of the simulation system in the domain. This statement can be made because it has been proven by the research contributions that the outcomes of the research do indeed support the simulation environment and its users. This research is in the domain of military simulation systems and makes a contribution to military ontologies.

CHAPTER 8

Conclusion

The preceding chapters reported on the design research process from the awareness stage to the final evaluation and research contributions. This chapter concludes the document by summarising the process followed and providing a concise description of what was described in detail in each chapter.

8.1 Background, Problem Statement and Research Question

Military aircraft are important and expensive assets. Huge investments are made to obtain, equip and maintain the aircraft, consequently, protection of the aircraft is a priority. Protection is needed from various threats and one of these threats is possible attacks by enemy missiles such as man-portable air defence systems (MANPADS). These missiles are widely available and easy to operate and protection against attacks from these missiles is vital for the survival of military aircraft operating in war zones. To increase survival rates against such attacks, countermeasures are implemented on the aircraft. In order to develop and deploy countermeasures successfully, it is essential that the behaviour and operation of these MANPADS are fully understood. The military utilises computer-based simulation to assist in understanding the behaviour of the missiles and in developing countermeasures against these missiles.

Simulation systems model real-world objects and simulate them in an artificial world. The development of computer models that simulate possible encounters between aircraft and missiles are important aspects in assessing the capabilities of defence systems and are fundamental in the development of aircraft countermeasures. Optronic Scene Simulator (OSSIM) is an image simulation tool that creates scenes of missile engagement in the visual and infrared bands. One specific application of OSSIM is that of assisting in the self-protection of military aircraft. Such an application utilises computer-based simulation of military scenes in order to perform countermeasure evaluation studies. Models of real world objects such as aircraft and missiles are implemented and used to simulate different scenarios to evaluate these models under different circumstances and to predict what will happen in the real world. The goal of the self-protection application is to assist, by simulation, the effective design and development of countermeasures deployed on military aircraft.

Although a successful application, there are issues and shortcomings in the system that led to an investigation being carried out to possibly better the studies already being performed in the simulation environment. The following list points out the issues raised by people involved in the application:

- Different people use different terms for objects and the meaning of words and terms are not always clear.
- The models in the system can be used in a number of ways and several parameters can be set for each model. For a person setting up the simulation, it is not always obvious to know what is available in the system and how models can be used together.
- Models behave in a certain way and although it might be possible to set up a simulation that is pro-grammatically correct and which produces 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. There is therefore a need to validate how the models are specified in the scenario.

An investigation was launched to examine if these issues can be solved and, if a solution can be implemented, will it indeed support the studies performed in the simulation environment. There was therefore a need to find a tool that could provide a solution to address these issues.

Ontologies and ontology engineering featured extensively when performing investigations into possible technologies that support modelling within Computer Science (CS) and simulation systems. Having a background knowledge of ontologies and ontology engineering was vital in order to reach the goal of the research project. In Chapter 2, the background study of ontology and the use of ontologies in CS systems are described. A study was undertaken to firstly look at the theory of ontologies, especially ontology engineering, and secondly to investigate the use of ontologies in computer simulations and military applications.

There are several terms to describe ontologies but they generally relate to the description and concepts of a domain. Traditional ontology paved the way for the use of ontologies in CS today. The need to represent and share knowledge in a formal way, encouraged by the requirements of the AI engineers, software developers and database modellers, played a role in the transition of ontologies to CS fields. Guarino [35] formulates a formal definition of ontologies in CS from the original definition of Gruber [34]. Guarino [35] aims to emphasise the difference between ontology and conceptualisation because he believes that an ontology is dependent on the language used, whilst a conceptualisation is not dependent on the language used. Guarino

[35] then defines ontology as follows:

'An ontology is a logical theory accounting for the intended meaning of a formal vocabulary.'

The literature study concluded that ontologies offer several advantages such as providing a structured way to analyse domain knowledge. Knowledge can be shared across the domain and thus provides a way to share the knowledge between software modules and users. It further provides reasoning mechanisms to the check consistency of classes and therefore ensures that an ontology is meaningful and correct with minimum redundancy [44]. Several applications in which ontologies were used in military applications using simulation were found and are as follows:

- Trajectory simulations (Section 2.6.1).
- Distributed simulation environment (Section 2.6.2).
- Mobile route planning (Section 2.6.3).
- Software agents (Section 2.6.4).

Having looked at the claims made by using ontologies, it was decided to investigate the use of ontology to address the requirements in the simulation system. The following research questions were formulated:

1. What are the requirements or concerns of a countermeasure simulation system environment that ontology technologies support?
2. How can an ontology be constructed that will capture the knowledge of a countermeasure simulation system environment?
3. Does the use of ontology technologies support and enhance the functions of the simulation system in the domain?

To answer the research questions, a research process was initiated by using design research as the research methodology.

8.2 Design Research Process

Figure 8.1, shows an overview of the research process followed. As seen in the left side of the diagram in the figure, the background study was completed prior to commencing the design research process.

Piirainen *et al.* [66] state that design research solves the relevant problems by adding to the body of knowledge, through the design of the artefact. This research was aimed at solving a known problem by developing an artefact, through design research, to solve the problem. The steps in design research, as described by Vaishnavi *et al.* [92] (illustrated in Figure 8.1) were followed. The guidelines proposed by

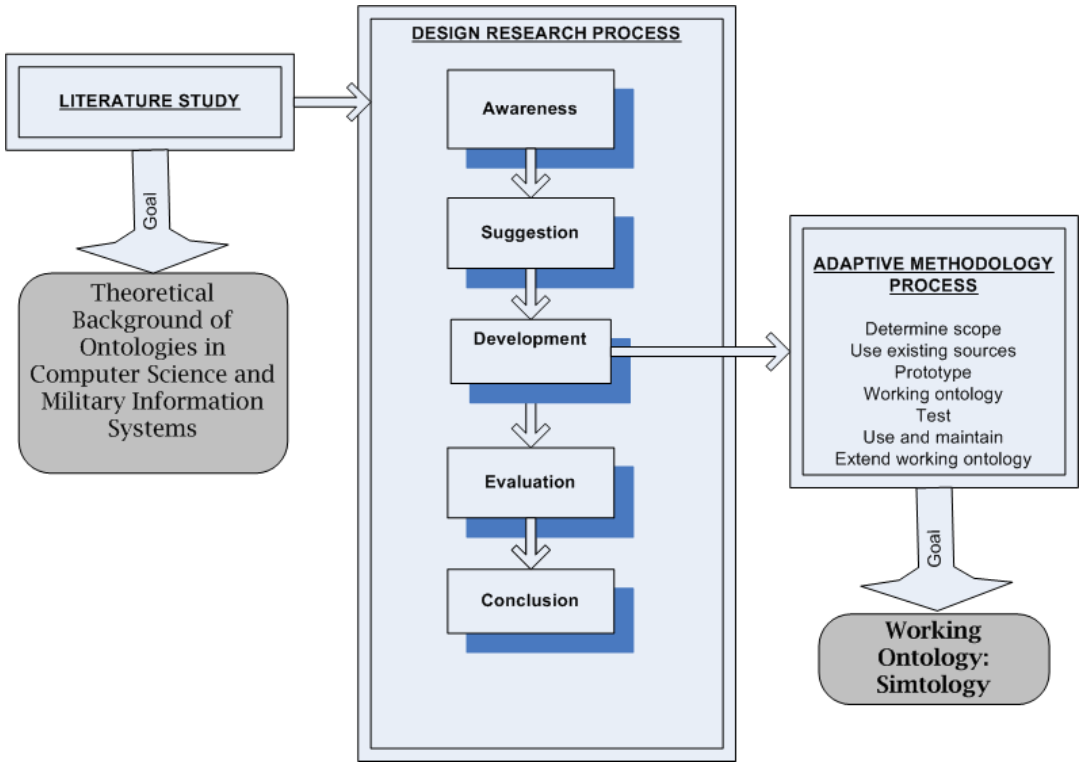

Figure 8.1: Overview of the Research Process

Hevner *et al.* [38], discussed in Section 3.2, were implemented to ensure successful design research as follows:

1. The output of the research is an IT artefact - Simtology.
2. There was a specific goal: to determine if an ontology can provide a solution for the issues in the simulation system.
3. The ontology was evaluated by using reasoners and validation techniques.
4. The research added to the body of knowledge by obtaining experience of the construction and the use of an ontology in the simulation environment
5. A proper research methodology was followed.
6. Alternative solutions were investigated and the most suitable solution was implemented.
7. The outcome of the research is the ontology and the ontology was put to use in the simulation environment.

The first stage in design research applied to the simulation environment was awareness that a problem or problems exist. The following issues were raised during the awareness stage:

- In the simulation system environment there are people with different skills and expertise. Different people are involved and therefore different functions are

performed. There is little common technical language to describe concepts in the simulation environment. A need therefore exists to have one source of terminology definitions and a shared language to be used by the members in the team.

- The application contains a number of models, each with a number of parameters that describe the models. There are also many scenarios where-in the models are defined. A tool is necessary to explain a specific scenario and the result of the simulation executed by running that scenario.
- New models are constantly being added to the simulation system by different clients. It is important that these models adhere to existing standards and rules. These standards and rules must be available not only in code and documentation, but also in a knowledge base.
- Models behave in certain ways and although it might be possible to set up a simulation that is pro-grammatically correct and produces 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. The simulation practitioner performing the simulation set up might not have specialist knowledge of how the models interact, and could thus set up scenarios that are syntactically correct but that do not make sense in the real world. This results in the need for a tool to assist in setting up valid scenarios.

Solving the above-mentioned requirements will not only improve the use of the simulation system but also provide meaning to the contents of the different scenarios. The problem statement therefore was to find a tool to provide a solution to these issues and requirements.

The adaptive methodology process, designed by Bergman [10], is an agile development methodology suitable for the construction of ontologies. With reference to Figure 8.1, the steps can be seen as part of the development stage of the design research process. The adaptive modelling process was applied to the simulation environment. The scope of the ontology covered the concepts in the application of self-protection only; not all the concepts in OSSIM. Existing sources of information were used to obtain a list of concepts in the domain. A prototype ontology was constructed and successfully used to illustrate the concepts and principles of an ontology in the simulation environment.

The prototype ontology was expanded to a working ontology by adding concepts from the domain to the ontology to include all the concepts in the self-protection application. The functionality provided by the Protégé editor provided viewing and

query capabilities. Apart from the functionality that Protégé provided, extra functionality was developed. The information in the ontology was integrated into the graphical user interface used to construct a simulation scenario. Functionality was developed to write out scenarios created in the ontology to files that can act as input to the simulation. This makes it possible for a scenario to be checked for logical correctness before it is run in the simulation. Modelling errors not handled by the simulation software are dealt with early in the simulation process by using the reasoning technology in the ontology. By having a scenario defined in the ontology, it is possible to export a high-level description of a scenario and its components to be used for reporting and documentation of simulation studies.

The working ontology was tested for completeness and consistency. Using the ontology in the simulation environment ensures that the ontology has value to the domain and supports the functions of the simulation systems. The use of the ontology provide input on what must be maintain. The final step is to extend the working ontology. The extension of the ontology can be done in several iterations and testing and evaluation at every iteration are very important.

Evaluation of the ontology followed the successful construction of the working ontology. The purpose of evaluation was to prove the claim that an ontology can play a supporting role in the simulation system. This was done by comparing the issues brought up in the awareness stage with the contribution of the construction and use of the ontology to the simulation environment. According to the initial requirements, the ontology should provide a shared, common language to address the problem of communication. It was found that the prototype ontology already provided this functionality. By analysing the domain and supporting documents in the domain, it was found that the concepts in the ontology not only resembles the models in the simulation but also provide the members of the Simulation Team with a shared language. The ontology provides a platform to share the information of the models across applications. The ontology furthermore provides knowledge about models and model behaviour. The ontology delivers a way to obtain a high-level description of a scenario; leading to better communication between different users. In the event of new models being added to the simulation system, the ontology provides additional information that is not contained in the documentation. By setting up scenarios in the ontology, reasoners (Section 2.7.4) can be used to validate the scenario against the rules and restrictions specified in the ontology.

In order to evaluate the research process, the check-list in Chapter 3 (as proposed by Hevner *et al.* [39]), was used and the following actions taken and conclusions made:

1. The main research question put forward was the following: How can ontology technologies be used to support a countermeasure simulation system environment? This evolved to the following sub-questions: What are the requirements or concerns of a countermeasure simulation system environment that ontology technologies support? How can an ontology be constructed that will capture the knowledge of a countermeasure simulation system environment? Does the use of ontology technologies support and enhance the functions of the simulation system in the domain?
2. The artefact was an ontology - Simtology.
3. Adaptive methodology as development methodology was applied in the development stage of the design research process to construct the artefact.
4. The theory of design research and how design research is conducted supports the research project.
5. The ontology was tested after construction and evaluated after obtaining a working ontology.
6. The prototype ontology was used to demonstrate the principles of an ontology in the simulation environment. The working ontology is used in the simulation environment as a communication tool.
7. The artefact, an ontology, is the outcome of the research. The ontology captures the knowledge of the concepts of one application of OSSIM thereby adding to the body of knowledge.
8. Simtology captures part of the knowledge contained in OSSIM and its functions support the performance of a simulation study. The research question has been addressed by evaluating if Simtology and the construction thereof provide a solution to the issues and shortcomings in the application.

On evaluating the supporting role of the ontology and the research process followed to construct the ontology, it was found that the ontology does indeed play a supporting role in the simulation system and that design research was successfully used to construct an ontology for the simulation environment. Four cases are mentioned to illustrate the evaluation of the ontology.

8.3 Outcomes and Research Contributions

The research outcome was an ontology, called Simtology, constructed for use in the simulation environment to support the countermeasure effectiveness studies. Simtology is a domain ontology containing all the information that is contained in the self-protection application of OSSIM.

The contribution of this research is not only an ontology, but also the knowledge gained through the process of constructing the ontology. It was necessary to conduct a thorough investigation into what exits in the domain and how to describe it. This process, as well as the construction process of the ontology, provides a rich insight into the domain, the models and how the simulation files are configured in the simulation.

Simtology contains a full set of classes which present the concepts in the simulation system. Functionality was developed to integrate the information in Simtology with other applications in the simulation environment. The information contained in Simtology is used to populate the elements in a Graphical user Interface (GUI) resulting in several advantages such as that only one source of simulation information has to be maintained, as well as that the ontology can be used to change the language displayed in the GUI.

Functionalities were also developed to write out scenarios created in the ontology to files that can act as input to the simulation. This made it possible that a scenario can first be checked for logical correctness before it is run in the simulation. Modelling errors not handled by the simulation software are handled early in the simulation process by using the reasoning technology in the ontology. By having a scenario defined in the ontology, it is possible to export a high-level description of a scenario and its components to be used for reporting and documentation of simulation studies.

During the construction of Simtology, the following observations were made:

- With regards to modelling, it is important to distinguish part-of from subclass-of. An aircraft body is part of an aircraft, not part of a specific type of aircraft or subclass.
- It is important to correctly model roles. Modelling a missile as an *observer* in the simulation means that it can never be used in the simulation as an *object of type moving*. In Simtology, a missile can therefore never be used in a different role.
- Equally important in the defining of concepts is the issue of individuals versus concepts. This decision has an impact on how the ontology can ultimately be used. The choice between concept and individual is often contextual and application-dependent but it needs to be evaluated in one of the development cycles.

8.4 Future Work

Simtology is utilised as a communication tool, a knowledge base, and as a tool to integrate different applications. It is maintained and expanded as the application to perform countermeasure studies is expanded and maintained. One of the future uses of Simtology is the possibility to use it in the training of pilots to more easily explain the system and how the real world can be tested in a simulation environment.

Future research will be performed to determine if Simtology can be expanded to include all the concepts and relations in OSSIM. The aim is to support all applications of OSSIM. Additional functions will be added to reverse engineer previously performed simulations and add the detail of all scenario simulations to the ontology.

A final aspect to emphasise is that the development and use of the ontology is an iterative process. Not only will it be maintained but as new functionality is added, it will be tested, used and evaluated. The simulation system is not stagnant. Improvements are constantly made and the system is expanded by adding new models, adding properties to existing entities in the system or adding new functionality. Simtology needs to grow with the system; therefore there will always be future expansions that will ensure that the concepts and their meanings in Simtology are always a true representation of the knowledge in the simulation system environment.

Bibliography

[1] Alatrish, E. S. (2012). "Comparison of Ontology Editors". In: *e-RAF Journal on Computing* 4, pp. 23–38 (cit. on p. 37).

[2] Alhir, S. S. (1998). *The Object-Orientated Paradigm*. URL: http://www.cs.utep.edu/sroach/S10-5380/TheObjectOrientedParadigm.PDF (visited on 10/02/2011) (cit. on p. 52).

[3] Ashburner, M. (2000). "Gene Ontology: Tool for the unification of biology". In: *Nature Genetics* 25, pp. 25–29 (cit. on pp. 8, 21).

[4] Baader, F., Horrocks, I., and Sattler, U. (2005). "Description Logics as Ontology Languages for the Semantic Web". In: *Mechanizing Mathematical Reasoning: Essays in Honour of Jörg H. Siekmann on the Occasion of His 60th Birthday*. Ed. by D. Hutter and W. Stephan. Vol. 2605. Lecture Notes in Artificial Intelligence. Springer-Verlag, pp. 228–248 (cit. on pp. 2, 17).

[5] Bailey, I. and Partridge, C. (2009). "Working with Extensional Ontology for Defence Applications". In: *Ontology in Intelligence Conference*. Fairfax, VA (cit. on pp. viii, 31).

[6] Baskauf, S. and RDF/OWL Task Group (2012). *Beginner's Guide to RDF - TDWG RDF/OWL Task Force*. URL: http://code.google.com/p/tdwg-rdf/wiki/Beginners (visited on 07/02/2012) (cit. on p. 42).

[7] Bellinger, G. (2004). *Mental Model Musings*. URL: http://www.systems-thinking.org (visited on 10/23/2012) (cit. on p. 8).

[8] Benjamin, P., Patki, M., and Mayer, R. (2006). "Using ontologies for simulation modeling". In: *Proceedings of the 38th conference on Winter simulation*. WSC '06. Monterey, California: Winter Simulation Conference, pp. 1151–1159 (cit. on pp. 8, 16, 25–27).

[9] Benjamin, P. and Akella, K. V. (2009). "Towards Ontology-driven Interoperability for Simulation-based Applications". In: *Winter Simulation Conference*, pp. 1375–1386 (cit. on pp. 33, 34, 115).

[10] Bergman, M. (2010). *A new methodology for building lightweight, domain ontologies*. URL: http://www.mkbergman.com/908/a-new-methodology-for-building-lightweight-domain-ontologies/ (visited on 10/02/2011) (cit. on pp. 12, 52, 53, 55, 83, 86, 124).

[11] Birchenall, R. P. *et al.* (2010). "Modelling an Infrared Man Portable Air Defence System". In: *Infrared Physics & Technology* (cit. on pp. 1, 73).

[12] Blessing, L. and Chakrabarti, A. (2009). *DRM, a Design Research Methodology*. Springer (cit. on p. 50).

[13] Bock, J. *et al.* (2008). *Benchmarking OWL Reasoners*. CEUR Workshop Proceedings (cit. on pp. 42, 43).

[14] Bolkcom, C. and Elias, B. (2006). *Homeland Security:Protecting Airliners from Terrorist Missiles*. Tech. rep. The Library of Congress (cit. on p. 1).

[15] Bowman, M. and Lopez A. M.and Tecuci, G. (2001). "Ontology Development for Military Applications". In: *Proceedings of the Thirty-ninth Annual ACM Southeast Conference*. ACM Press (cit. on p. 9).

[16] Bray, T. *et al.* (2008). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Tech. rep. W3C. (Visited on 04/24/2012) (cit. on p. viii).

[17] Carnap, R. (1950). "Empiricism, Semantics, and Ontology". In: *Revue Internationale De Philosophie* 4.2, pp. 20–40 (cit. on p. 19).

[18] Carson II, J. S. (2004). "Introduction to modeling and simulation". In: *WSC '04: Proceedings of the 36th conference on Winter simulation*. Washington, D.C.: Winter Simulation Conference, pp. 9–16. ISBN: 0-7803-8786-4 (cit. on p. 5).

[19] Clausewitz, C. von (1989). *On War*. Ed. by P. P. Michael Eliot Howard Michael Howard. Trans. by P. P. Michael Howard. Princeton paperbacks. Princeton University Press. ISBN: 9780691018546. URL: http://books.google.co.za/books?id=DRoL0NmsrlwC (cit. on pp. 9, 29).

[20] Devedzic, V. (2002). "Understanding ontological engineering." In: *Communications ACM* 45.4, pp. 136–144. URL: http://dblp.uni-trier.de/db/journals/cacm/cacm45.html (cit. on p. 36).

[21] Dolbear, C. *et al.* (2005). *Semantic interoperability between topographic data and a flood defence ontology*. Tech. rep. Ordnance Survey Research & Innovation, Romsey Road Southampton SO16 4GU, UK (cit. on p. 21).

[22] Dragan, G., Djuric, D., and Devedíc, V. (2009). *Model Driven Engineering and Ontology Development*. Berlin: Springer (cit. on pp. 17, 36, 38, 44).

[23] Durak, U., H., O., and Ider S, K. (2006). "An ontology for trajectory simulation". In: *Proceedings of the 2006 Winter Simulation Conference*. Ed. by L. F. Perrone *et al.*, pp. 1160–1167 (cit. on pp. 33, 115).

[24] Durak, U., Güler, S., and Oguztuzun H. Ider, K. (2007). "An Exercise In Ontology Driven Trajectory Simulation with MATLAB Simulink". In: *21st EUROPEAN Conference on Modelling and Simulation*. Prague, Czech Republic (cit. on p. 33).

[25] Education, J. and Doctrine Division J-7, J. S. (2010). *Department of Defense Dictionary of Military and Associated Terms*. Ed. by J. D. Directorate for Joint Force Development J-7 and E. Division. URL: http://www.dtic.mil/

doctrine/dod/_dictionary (visited on 03/28/2012). 15 June 2013 (cit. on p. 31).

[26] Eriksson, H. *et al.* (2009). "Simulation modeling using Protégé". In: *Proceedings of the Eleventh International Protégé Conference* (cit. on pp. 28, 38).

[27] Fensel, D. (2003). *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. 2nd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc. (cit. on pp. 23, 44).

[28] Fishwick, P. A. and Miller, J. A. (2004). "Ontologies for modeling and simulation: issues and approaches". In: *Proceedings of the 36th conference on Winter simulation*. WSC '04. Washington, D.C.: Winter Simulation Conference, pp. 259–264 (cit. on pp. 16, 26–28).

[29] Friedman, K. (2003). "Theory construction in design research: criteria: approaches, and methods". In: *Design Studies* 24, pp. 507–522 (cit. on p. 47).

[30] Gerber, A., Kotzé, P., and van der Merwe, A. (2010). "Towards the Formalisation of the TOGAF Content Metamodel using Ontologies". In: *ICEIS (2)*, pp. 54–64 (cit. on pp. 63, 90).

[31] Goddard, W. and Melville, S. (2004). *Research Methodology: An Introduction*. Juta Academic (cit. on p. 46).

[32] Gomez-Perez, A. (1995). "Some ideas and examples to evaluate ontologies". In: *Artificial Intelligence for Applications, 1995. Proceedings., 11th Conference on*, pp. 299–305 (cit. on pp. 54, 65, 103).

[33] Gómez-Pérez, A., Fernández-López, M., and Corcho, O. (2005). *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*. Advanced Information and Knowledge Processing. Springer (cit. on pp. 36, 37).

[34] Gruber, T. R. (1993). "A Translation Approach to Portable Ontology Specifications". In: *Knowledge Acquisition* 5.2, pp. 199–220 (cit. on pp. 2, 6, 7, 22, 23, 44, 121).

[35] Guarino, N. (1998). "Formal Ontology and Information Systems". In: *Proceedings of the First International Conference on Formal Ontologies in Information Systems (FOIS-98), June 6-8, 1998, Trento, Italy*. Ed. by N. Guarino. IOS Press, Amsterdam, The Netherlands, pp. 3–15 (cit. on pp. 16, 22, 23, 121).

[36] Guizzardi, G. (2007). "On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models". In: *Proceeding of the 2007 conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DB&IS'2006*. Amsterdam, The Netherlands, The Netherlands: IOS Press, pp. 18–39 (cit. on pp. 18, 20).

[37] Guzzoni, D., Baur, C., and Cheyer, A. (2007). “Modeling Human-Agent Interaction with Active Ontologies”. In: *Interaction Challenges for Intelligent Assistants'07*, pp. 52–59 (cit. on p. 21).

[38] Hevner, A. R. *et al.* (2004). “Design Science in Information Systems Research”. In: *MIS Quarterly* 28.1, pp. 75–105 (cit. on pp. 46–50, 54, 123).

[39] Hevner, A. R. and Chatterjee, S. (2010). “Evaluation”. In: *Design Research in Information Systems*. Vol. 22. Integrated Series in Information Systems. Springer US, pp. 109–120 (cit. on pp. 12, 46–48, 67, 109, 113, 125).

[40] Hevner A. R.and Chatterjee, S. (2010). “Evaluation”. In: *Design Research in Information Systems*. Vol. 22. Integrated Series in Information Systems. Springer US, pp. 109–120. ISBN: 978-1-4419-5653-8 (cit. on p. 57).

[41] Hitzler, P. *et al.* (2009). *OWL 2 Web Ontology Language Primer*. W3C Recommendation. World Wide Web Consortium. URL: http://www.w3.org/TR/owl2-primer/ (visited on 05/27/2010) (cit. on pp. 17, 39, 89).

[42] Holmes, D. and Stocking, R. (2009). “Augmenting agent knowledge bases with OWL ontologies”. In: *Aerospace conference, 2009 IEEE*, pp. 1 –15 (cit. on pp. 31, 35, 36, 116).

[43] Horridge, M. (2009). *A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools: Edition 1.2*. URL: http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_2.pdf (visited on 05/27/2009) (cit. on p. 91).

[44] Horrocks, I. (2005). *Description Logic Reasoning*. 13th International Conference on Conceptual Structures. URL: http://www.kde.cs.uni-kassel.de/conf/iccs05/horrocks_iccs05.pdf (cit. on pp. 45, 122).

[45] Irene, P. (2003). *Ontology Tool Support (Ontology Development Lifecycle and Tools)*. Tech. rep. Alexandria. USA: TopQuadrant Technology Briefing (cit. on p. 37).

[46] Johannesson, P, Perjons, E., and Bider, I. (2013). “What are the siblings of design science research?” In: *SIG Prag Workshop on IT Artefact Design & Workpractice Improvement* (cit. on p. 118).

[47] Kapoor, B. and Sharma, S. (2010). “A Comparative Study Ontology Building Tools for Semantic Web Applications”. In: *International Journal of Web & Semantic Technology* 1 (3), pp. 1–13 (cit. on pp. 37, 38).

[48] Kung, J. (1986). “Aristotle on "Being Is Said in Many Ways"”. In: *History of Philosophy Quarterly* 3.1, pp. 3–18 (cit. on p. 6).

[49] Lacy, L. W. *et al.* (2005). "Experiences Using OWL in Military Applications". In: *OWLED*. Ed. by B. C. Grau *et al.* Vol. 188. CEUR Workshop Proceedings. CEUR-WS.org (cit. on p. 16).

[50] LLC, C. . P. *Pellet: OWL 2 Reasoner for Java*. URL: `http://clarkparsia.com/pellet` (cit. on p. 43).

[51] Lombard, N. (2009). "Investigating Suitable Notation for Documenting Infrared Countermeasure Simulations". Research Script. University of South-Africa (cit. on p. 60).

[52] Lombard, N. (2010). "Supporting role of ontology in a simulation system for countermeasure evaluation." In: *Proceedings of the Workshop on ICT Uses in Warfare and the Safeguarding of Peace*, pp. 22–30 (cit. on p. 14).

[53] Lombard, N., Gerber, A., and van der Merwe, A. (2012). "Using Formal Ontologies in the Development of Countermeasures for Military Aircraft". In: *Proceedings of the Eighth Australasian Ontology Workshop, Sydney, Australia*, pp. 98–110 (cit. on p. 14).

[54] Mandrick, B. (2012). "Military Ontology". In: URL: `http://militaryontology.com/` (visited on 02/12/2012) (cit. on pp. 9, 16, 29, 30).

[55] MATLAB (2010). *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc. (cit. on p. 33).

[56] McGguinness, D. L. (2003). "Ontologies Come of Age". In: *The Semantic Web: Why, What, and How*. Ed. by Dieter. MIT Press (cit. on pp. 20, 23, 80).

[57] Menzel, M. and Mayer, R. (1998). "The IDEF Family of Languages". In: *Handbook on Architectures of Information Systems – InternationalHandbooks on Information Systems, P. Bernus et al., Springer-Verlag,Heidelberg*, pp. 209–242 (cit. on p. 80).

[58] Miller, J. A. *et al.* (2004). "Investigating Ontologies for Simulation Modeling". In: *Proceedings of the 37th annual symposium on Simulation*. ANSS '04. Washington, DC, USA: IEEE Computer Society, pp. 55–63. ISBN: 0-7695-2110-X (cit. on p. 25).

[59] Murdock, J., Buckner, C., and Allen, C. (2010). "Two Methods for Evaluating Dynamic Ontologies". In: *KEOD*, pp. 110–122 (cit. on p. 43).

[60] Nagle, J. A. *et al.* (2008). "Using an Ontology for Entity Situational Awareness in a Simple Scenario". In: *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* 5.2, pp. 139–158 (cit. on pp. 17, 34, 35, 38, 116).

[61] Niles, I. and Pease, A. (2001). "Towards a standard upper ontology". In: *Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001*. FOIS '01. ACM, pp. 2–9 (cit. on p. 33).

[62] Noy, N. F. and McGuiness, D. L. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*. Tech. rep. Stanford Knowledge Systems Laboratory (cit. on pp. 8, 61–63, 80, 84, 90, 91, 94).

[63] Oates, B. (2006). *Researching information systems and computing*. SAGE (cit. on pp. 46, 47, 49–52, 57, 58, 83, 115).

[64] Obitko., M. (2007.). "Translations between Ontologies in Multi-Agent Systems." PhD thesis. Faculty of Electrical Engineering, Czech Technical University in Prague. (cit. on pp. 17, 42).

[65] OWL Working Group (2009). *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation. URL: http://www.w3.org/TR/owl2-overview/ (visited on 12/12/2012) (cit. on pp. 39, 40, 44).

[66] Piirainen, K., Gonzalez, R., and Kolfschoten, G. (2010). "Quo Vadis, Design Science? - A Survey of Literature". In: pp. 93–108 (cit. on pp. 68, 122).

[67] Poli, R. (2006). "The Ontology of What is Not There". In: *Poznan Studies in the Philosophy of the Sciences and the Humanities* 91.1, pp. 73–80 (cit. on pp. 18, 19).

[68] Porzel, R. and Warden, T. (2010). "Working Simulations with a Foundational Ontology". In: *European Conference on Artificial Intelligence. Workshop on Artificial Intelligence and Logistics (AILog)*. Ed. by K. Schill, B. Scholz-Reiter, and L. Frommberger. (workshop proceedings), pp. 67–72 (cit. on p. 8).

[69] Preece, A. *et al.* (2007). "An Ontology-Based Approach to Sensor-Mission Assignment". In: *1st Annual Conference of the International Technology Alliance (ACITA), Maryland, USA* (cit. on p. 30).

[70] Protege. *The Protege Ontology Editor*. URL: http://protege.stanford.edu (visited on 04/13/2011) (cit. on pp. 17, 38, 43, 89, 93, 97).

[71] Prototyping. *Prototyping Software Life Cycle Model*. URL: http://www.freetutes.com/systemanalysis/sa2-prototyping-model.html (visited on 09/20/2011) (cit. on p. 52).

[72] Quine, W. V. (1948). "On What There Is". In: *Review of Metaphysics* 2, pp. 21–38 (cit. on p. 18).

[73] Samuel-Ojo, O. *et al.* (2010). "Meta-analysis of Design Science Research within the IS Community: Trends, Patterns, and Outcomes". In: *Global Perspectives on Design Science Research*. Ed. by R. Winter, J. Zhao, and S. Aier. Vol. 6105.

Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 124–138 (cit. on p. 47).

[74] Sanchez, D., Cavero, J., and Martinez, E. (2007). "The Road Toward Ontology". In: *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*. Springer US, pp. 3–20 (cit. on pp. 7, 8, 18, 20, 44).

[75] Sandholm, T. (2005). *The Philosophy of the Grid: Ontology Theory - From Aristotle to Self-Managed IT Resources*. Tech. rep. TRITA-NA-0532. Stockholm, Sweden: Royal Institute of Technology (cit. on pp. 18–20).

[76] Santini, S. (2008). "Ontology: Use and Abuse". In: *Adaptive Multimedial Retrieval: Retrieval, User, and Semantics*. Ed. by N. Boujemaa, M. Detyniecki, and A. NÃČÂijrnberger. Vol. 4918. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 17–31 (cit. on p. 19).

[77] Schlenoff, C., Washington, R., and Barbera, T. (2005). "An Intelligent Ground Vehicle Ontology to Enable Multi-Agent System Integration". In: *Integration of Knowledge Intensive Multi-Agent Systems*, pp. 169–174 (cit. on pp. 16, 17, 29, 38, 115).

[78] Sharifloo, A. A. and Shamsfard, M. (2008). "Using Agility in Ontology Construction". In: *Proceedings of the 2008 conference on Formal Ontologies Meet Industry*. Amsterdam, The Netherlands, The Netherlands: IOS Press, pp. 109–119 (cit. on p. 52).

[79] Showalter, J. (2011). *U.S. Air Force AC-130H Spectre gunship jettisons decoy flares*. URL: `http://publicdomainpictures.wordpress.com/` (visited on 03/28/2012) (cit. on p. 4).

[80] Silver, G., Hassan, O., and Miller, J. (2007). "From domain ontologies to modeling ontologies to executable simulation models". In: *Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come*. WSC '07. IEEE Press, pp. 1108–1117. ISBN: 1-4244-1306-0 (cit. on pp. 16, 26).

[81] Simon, H. A. (1996). *The Sciences of the Artificial*. 3rd ed. The MIT Press (cit. on p. 48).

[82] Siricharoen, W. V. (2008). "A Software Engineering Approach to Comparing Ontology Modeling with Object Modeling". In: *Computer Science and its Applications, International Symposium on* 0, pp. 320–325 (cit. on p. 24).

[83] Siricharoen, W. V. (2006). "Ontologies and Object models in Object Oriented Software Engineering". In: *IMECS*, pp. 856–861 (cit. on p. 24).

[84] Smart, P. R. *et al.* (2007). "AKTiveSA". In: *Computational Journal* 50 (6), pp. 703–716 (cit. on pp. 29, 31, 32, 115).

[85] Smith, B. (2004). "Beyond Concepts: Ontology as Reality Representation". In: IOS Press, pp. 73–84 (cit. on p. 41).

[86] Smith, B. and Welty, C. (2001). "Ontology: Towards a New Synthesis". In: *Proceeding of the 2nd International Conference on Fortmal Ontology and Information Systems (FOIS 2001)*. Ogunquit, Maine, USA (cit. on p. 20).

[87] Smith, B. (2003). In: *The Blackwell Guide to the Philosophy of Computing and Information*. Ed. by L. Floridi. Oxford: Blackwell. Chap. Ontology, pp. 155–166 (cit. on pp. 7, 16, 18–20, 42).

[88] Smith, B., Köhler, J., and Kumar, A. (2004). "On the Application of Formal Principles to Life Science Data: a Case Study in the Gene Ontology". In: *Data Integration in the Life Sciences, First International Workshop*. Ed. by E. Rahm. Vol. 2994. Lecture Notes in Computer Science. Springer (cit. on p. 21).

[89] StrategyPage (2012). *The Strategy Page*. URL: http://www.strategypage.com/military_photos (visited on 03/29/2012) (cit. on p. 2).

[90] Tran, T., Lewen, H., and Haase, P. (2007). "On the Role and Application of Ontologies in Information Systems". In: *Proceedings of the 5th IEEE International Conference on Computer Science - Research, Innovation and Vision for the Future* (cit. on pp. 24, 25).

[91] Tsarkov, D. and Horrocks, I. (2006). "FaCT++ Description Logic Reasoner: System Description". In: *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*. Vol. 4130. Lecture Notes in Artificial Intelligence. Springer, pp. 292–297 (cit. on p. 43).

[92] Vaishnavi, V. and Kuechler, W. (2004). *Design Research in Information Systems*. URL: http://desrist.org/design-research-in-information-systems/ (visited on 03/28/2012) (cit. on pp. 12, 47, 49, 50, 52, 58, 59, 68, 70, 82, 83, 108, 115, 118, 122).

[93] Valente, A., Holmes, D., and Alvidrez, F. C. (2005). "Using a Military Information Ontology to Build Semantic Architecture Models for Airspace Systems". In: *Aerospace Conference, 2005 IEEE*, pp. 1–7 (cit. on pp. 16, 29, 30, 115).

[94] Viinikkala, M. (2004). *Ontology in Information Systems*. URL: http://www.cs.tut.fi/~kk/webstuff/Ontology.pdf (visited on 03/29/2011) (cit. on pp. 16, 21).

[95] Wang, H. *et al.* (2005). "Debugging OWL-DL Ontologies: A Heuristic Approach". In: ed. by 4th International Semantic Web Conference (ISWC 2005). Vol. LNCS 3729. Springer, pp. 745–757 (cit. on p. 104).

[96] White, S. (2005). *Better computational descriptions of science*. URL: http://www.scientific-computing.com/features/feature.php (visited on 04/14/2012) (cit. on p. 7).

[97] Willers C. J.and Willers, M. S. (2012). *OSSIM: Optronics System Simulator*. Tech. rep. Pretoria: DPSS, CSIR (cit. on pp. viii, 3, 71, 77).

[98] Winklerova, Z. (2003). *Ontological Approach to the Representation of Military Knowledge*. Tech. rep. Military Academy in Brno, Command and Staff Faculty, Czech Republic (cit. on pp. 29, 31, 32, 115).

[99] Yu, J., Thom, J. A., and Tam, A. (2009). "Requirements-oriented methodology for evaluating ontologies". In: *Information Systems* 34.8, pp. 686–711 (cit. on p. 43).

[100] Zúñiga, G. L. (2001). "Ontology: its transformation from philosophy to information systems". In: *Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001*. FOIS '01. ACM, pp. 187–197 (cit. on pp. 16, 22, 23, 80).

APPENDIX: Extract from Simtology

Simtology contains all the classes that are necessary to set up a scenario in the simulation environment. The following is a list of figures that visually show a subset of the information contained in Simtology:

- Figure A.1, the classes in Simtology
- Figure A.2, the individuals in Simtology
- Figure A.3, an instance of an Aircraft class
- Figure A.4, the data properties
- Figure A.5, the object properties

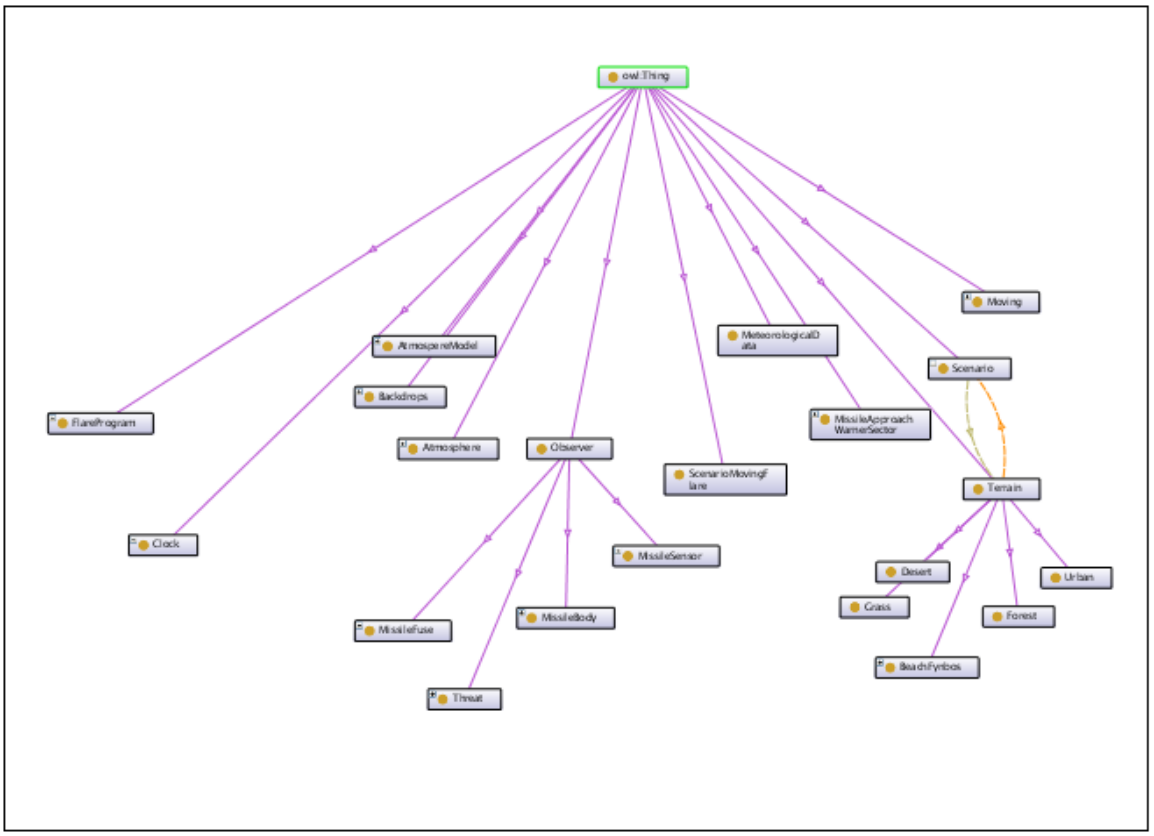

Figure A.1: Ontology Classes of Simtology

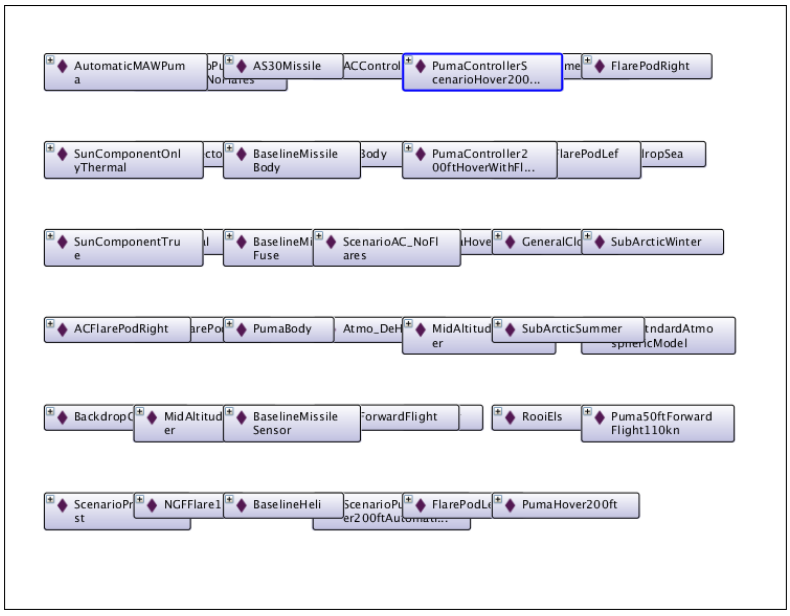

Figure A.2: Individuals in Simtology

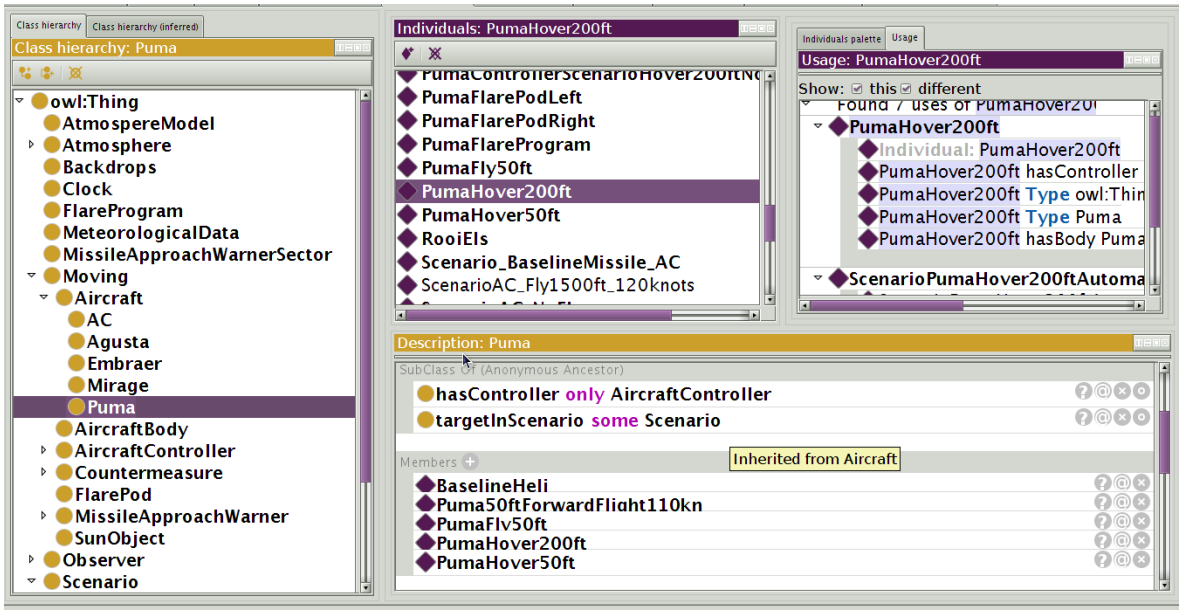

Figure A.3: Instance of an Aircraft Class

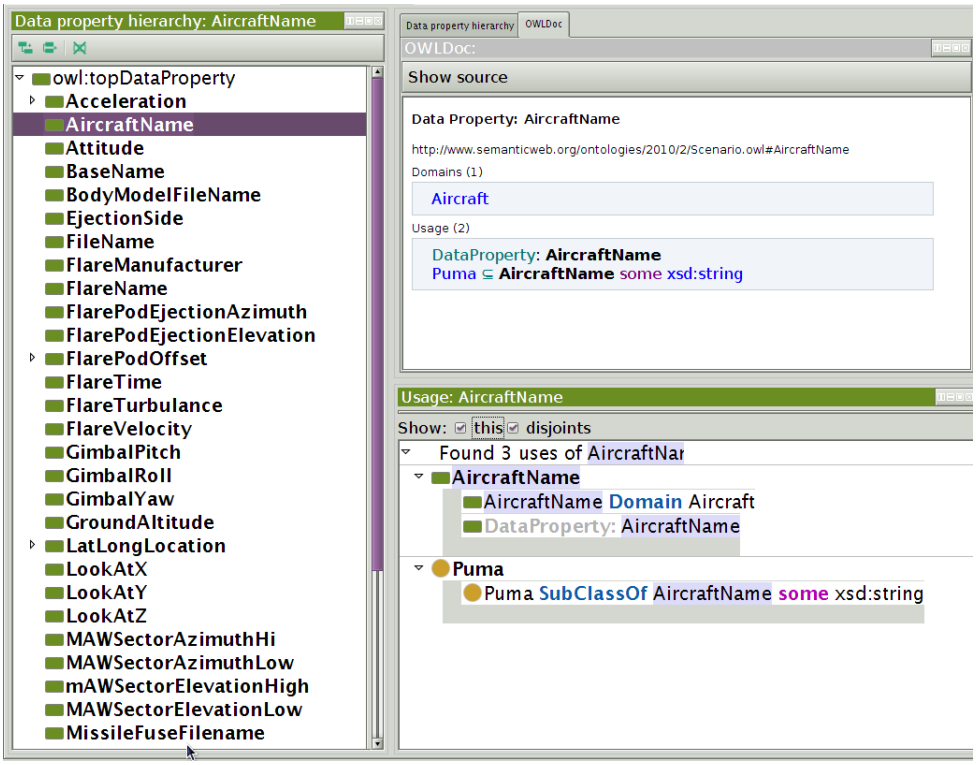

Figure A.4: Data Properties in Simtology

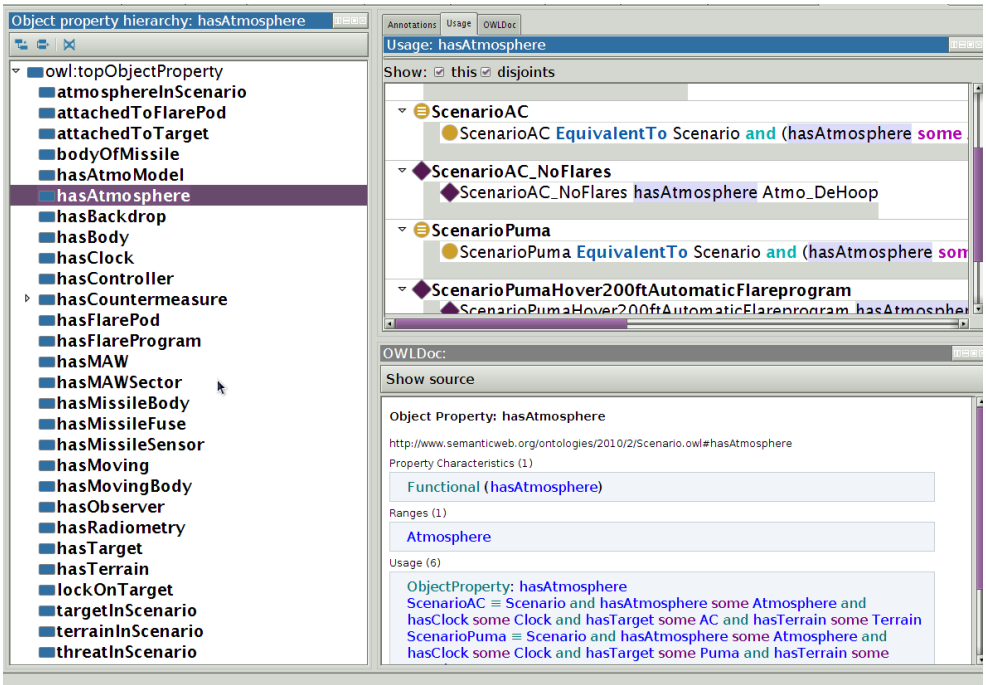

Figure A.5: Object Properties in Simtology