Situation Assessment in the Ground Target Domain

Jenny Lagerlöf

September 2008
Master's Thesis in Computing Science, 30 ECTS credits
Supervisor at CS-UmU: Michael Minock
Supervisors at Saab Systems: Klas Wallenius and Johan Edlund
Commissioners at Saab Systems: Andreas Lingvall and Klas Wallenius
Examiner at CS-UmU: Per Lindström

Abstract

Tracking in the ground target domain is not as developed as the tracking of aircrafts and vessels. One of the main difficulties with tracking land objects is the irregularity of the incoming data; an operator must rely on reconnaissance reports together with snippets of sensor information. This puts high demands on the operator to associate the right report with the right tracked object. The Data Fusion team at Saab Systems is interested in investigating how ontological engineering can support decision-making in a Battle Management System. This thesis looks into how ontological reasoning can be used to find associations between an incoming report and an existing object represented in a knowledge base. The approach is to only present objects that are consistent with the incoming report. A method to rank the possible matches by their similarity using methods from the area of Information Retrieval is also presented. To get a broader picture of the benefits of methods within ontological engineering, a study of how it can be used within the field of situation detection is also included in this thesis. Results from the prototype application developed shows that it is achievable to retrieve only objects that are consistent for an association, thus preventing an operator to make an incorrect matching of an incoming report to a given tracked object.

Contents

1. Introduction

1.1. Background

At Saab Systems, there has been an ongoing development during several years to track and identify moving objects, aircrafts and ships for example. This technology gives the operator a picture of the situation containing the object's position and velocity, as well as its motion history since it was first detected. To increase the understanding of a situation to a higher extent than knowing what objects exist and how they are moving, Saab Systems has lately been researching the area of *situation assessment*.

The goal with situation assessment is to increase an operator's awareness of the situation by finding connections between objects and their surroundings. The foundation of the analysis is an *ontology*: a formal description of a domain that specifies what objects can exist and what the relationships between them may be.

The field of target tracking in the ground domain is not as developed as the tracking of aircrafts and vessels is. One of the main difficulties with tracking land objects is that the situation picture consists mostly of intermittent information, i.e. the objects cannot be tracked continuously. The operator, in this case an intelligence analyst, must therefore rely on reconnaissance reports together with snippets of sensor information from areas where sensors do exist. This generates a very complex situation picture that puts high demands on the analyst in order to match the right object with the right report and the right track.

1.2. Purpose and Goals

The data fusion team at Saab Systems is interested in investigating how ontological engineering can be used to support decision-making in a battle management system. The purpose and major goal of this thesis is to present methods of ontological reasoning and how it can be used to find associations between existing objects in a knowledge base and an incoming reconnaissance report. One of the difficulties lies in the irregularity of the incoming reports from humans as well as from technical sensors. An additional goal with this thesis is to obtain a broader picture of the possible usage areas of ontological reasoning, why a study of how this technology can be used to identify and recognise situations is included. The ambition is to develop an application that can test the ideas presented in this project.

1.3. Outline

Chapter 2 describes how an ontological reasoning module can fit in a Battle Management System. A scenario is also presented to get further understanding of the problem domain. The Theoretical Background in Chapter 3 provides further knowledge of ontologies and related topics. A study on how ontological reasoning can be used within the field of Situation Detection is also included in this chapter. An approach that addresses the *association* problem is presented in Chapter 4. Chapter 5 describes the implementation of the application. The results of the test

runs can be reviewed in Chapter 6, Results, followed by Conclusions in Chapter 7, which also includes a discussion and future work.

2. Problem Description

To get a full understanding of how an operator can benefit from the Ontological Reasoning Module, it is important to know how it is intended to be used. This section describes how an ontological reasoning module works in its context. A simple scenario is also presented, to further illustrate how an ontological reasoning tool can support an operator.

2.1. An Ontological Reasoning Module in a Battle Management System

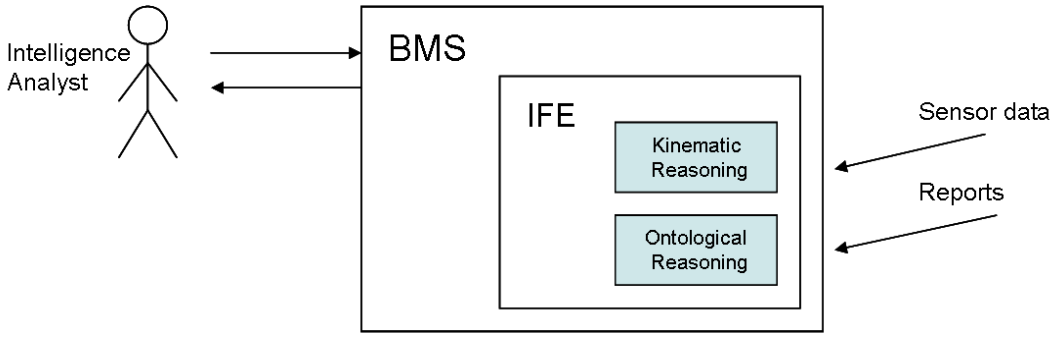


Figure 1: Overview of the Architecture in a BMS-system

The purpose with a Battle Management System (BMS) is to support decision-making in an army battalion by giving the operator a correct situation picture. This is achieved by gathering, merging and presenting information from different kinds of sources, such as radar, ground-sensors, UAV-pictures and reconnaissance reports. Figure 1 shows how an ontological and kinematical module constitutes a vital part of an Intelligence Fusion Engine (IFE), which purpose is to improve the situation picture presented to the operator. The task for the ontological reasoning module is to associate incoming data with existing objects in the system regarding structural properties. The kinematical reasoning module on the other hand, is only concerned with the objects physical properties. Output from these two modules are then merged and presented to the operator.

2.2. Scenario

To exemplify the problem, a simple scenario has been developed and is described in Figure 2. The purpose of the scenario is to show how ontological tools and reasoning can be used to associate reconnaissance information. The scenario describes how two hostile platoons (red) approach an area protected by friendly platoons (blue). A network of ground sensors is placed in such a way that it can detect vehicles passing the main road leading into the protected area. Additionally, a reconnaissance-troop patrols the south area around the river. The left column represents the real world, and the right column represents the things we know.

1) In the first part of the scenario, the operator receives reports from an UAV that four vehicles with cannons and heavy machine-guns and three vehicles with machine-guns have been spotted some distance from the area.

2) The scenario continues with a report from the ground sensor network with information of a light, wheeled vehicle. We do not know that the mechanised infantry platoon has deviated from the main road.

3) In step 3, a reconnaissance platoon reports that they have observed three vehicles, APCs. A tank commander reports a Red Cross Volvo car, and the ground sensor network reports four heavy, tracked vehicles.

Desired output from the implemented module for this scenario would be:

1) Classification results of the reported vehicles. Since the knowledge base is initially empty, no association will be done.

2) Classification results and indication of possible associations to existing objects.

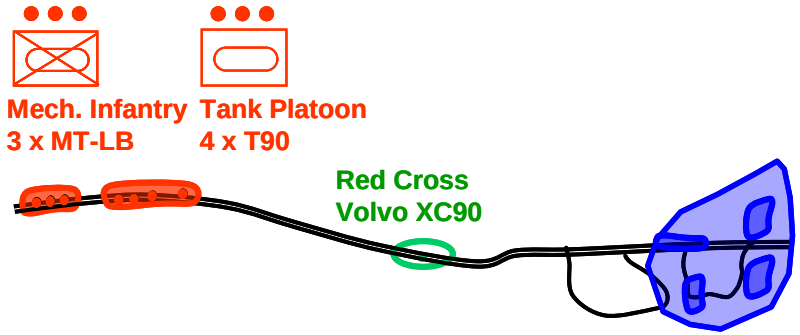
3) Classification results of the incoming reports and suggestions of associations to existing objects in the knowledge base.

2.3. Situation Detection Study

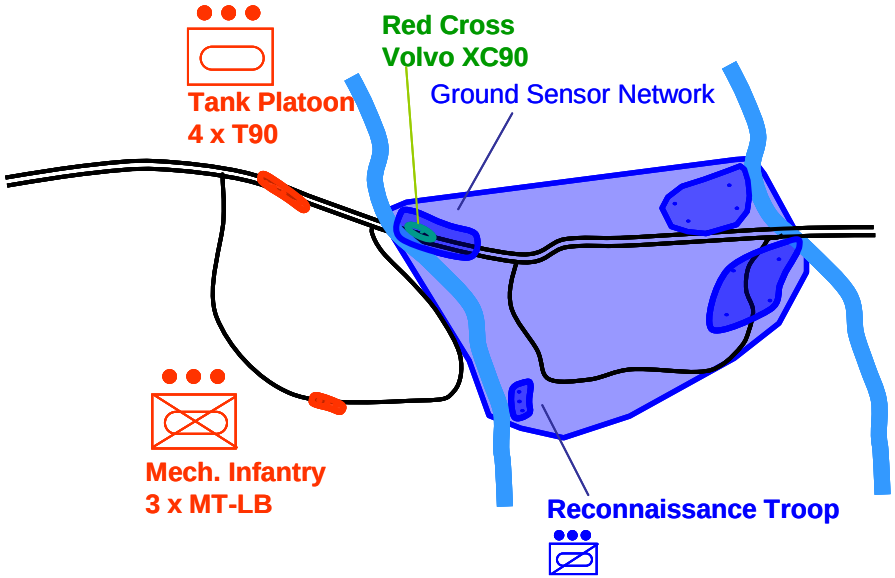
In addition to the ontological reasoning module, this project also includes a small study of how ontological reasoning can be used within the field of situation detection. This study will examine if the common ontology-language OWL can be used as a tool to define and recognise situations, and is presented section 4.3.

What is there

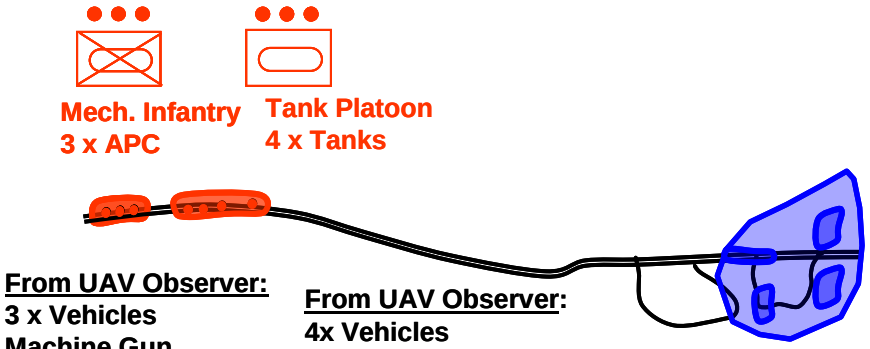
1.



2.

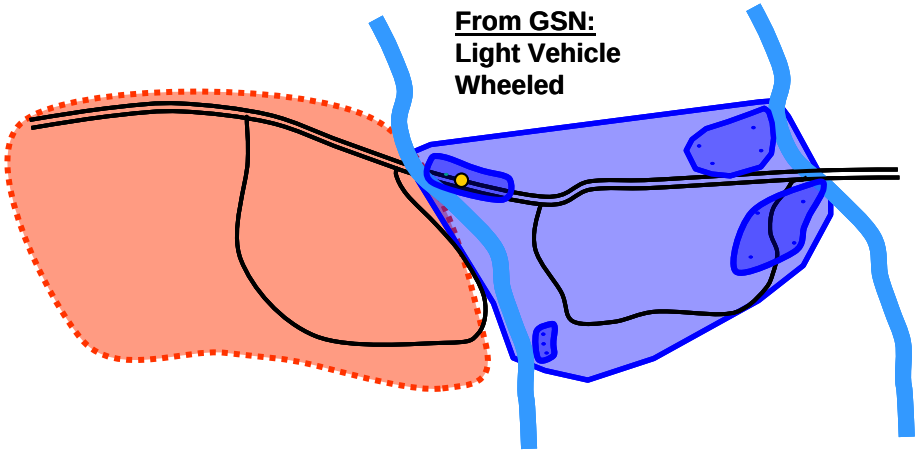


What we see



From UAV Observer:
3 x Vehicles
Machine Gun

From UAV Observer:
4x Vehicles
Machine Gun
Cannon



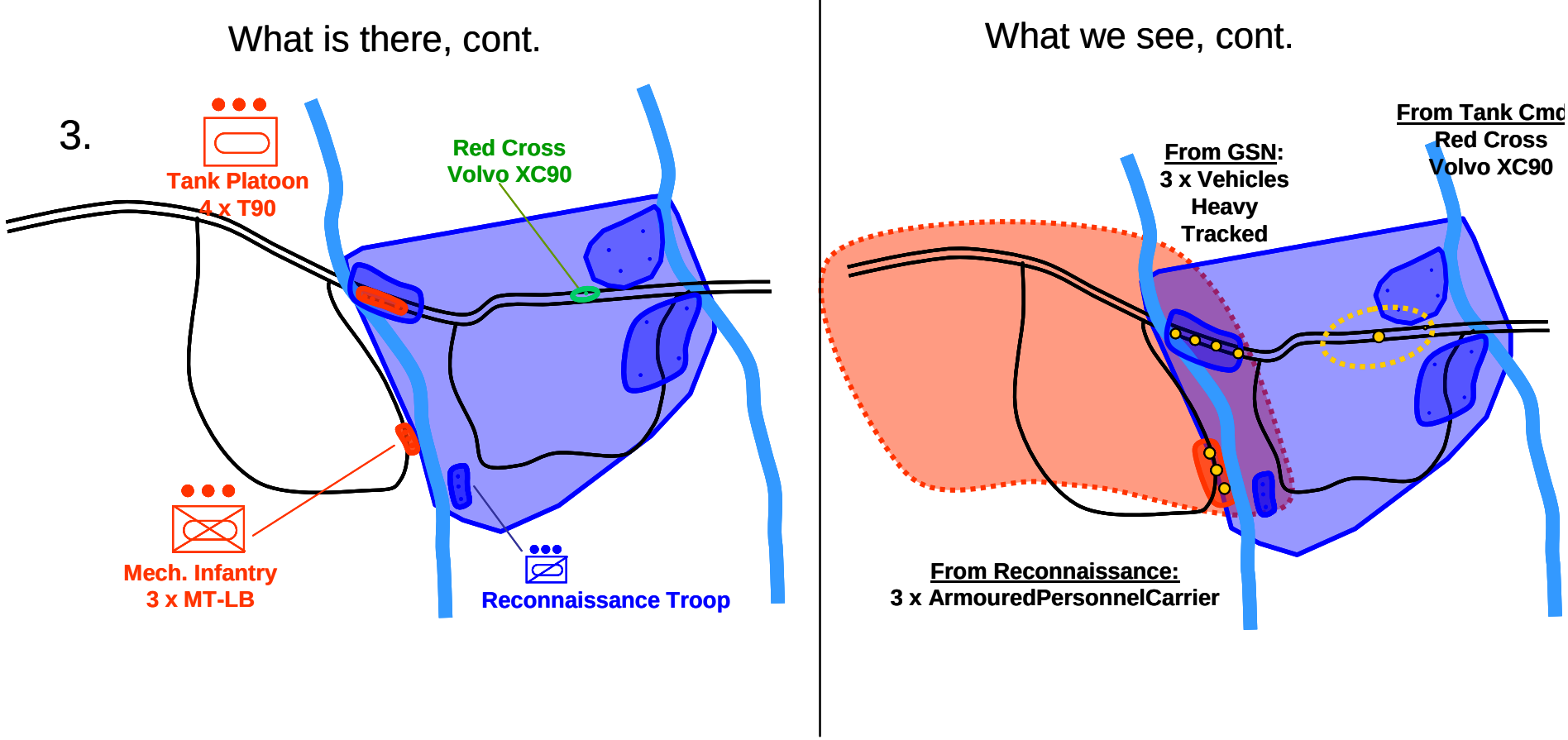


Figure 2: The scenario

3. Theoretical Background

This section provides a theoretical background of topics connecting to ontologies and reasoning. Starting with a brief description of data fusion, an area that comprises the subject of this project, we will continue with ontologies and their representation. Emphasis will be on OWL, especially OWL-DL, named so due to its correspondence with description logics, also described in this chapter.

3.1. Data Fusion and Situation Assessment

The IFE defined in the previous chapter is an application of *Data Fusion*. This research field addresses the process of combining data from separate sources and different kinds of sensors so that the resulting information is better than it would be using the sources individually. Steinberg defines data fusion as "the process of combining data or information to estimate or predict entity states" [Steinberg and Bowman, 2001].

The Joint Directors of Laboratories (JDL) of the US Department of Defence has developed a model of data fusion called the JDL-model, see Figure 3.

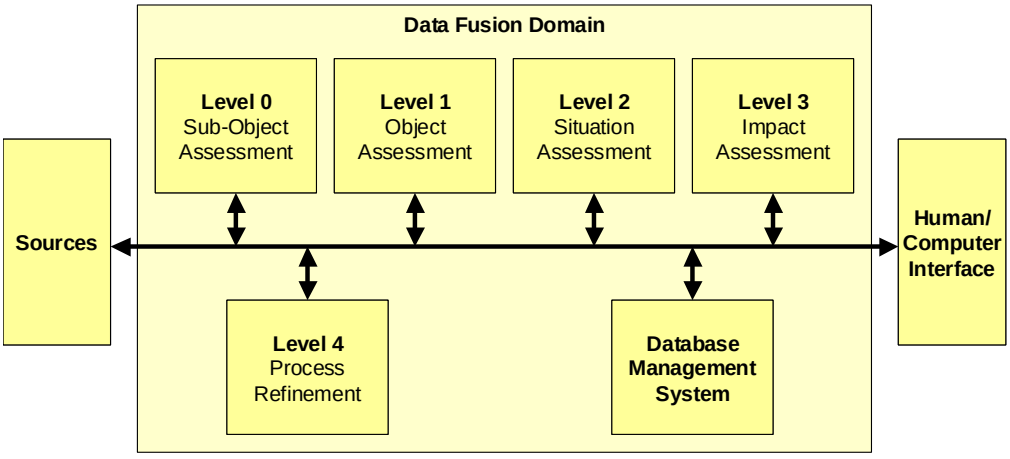


Figure 3: The revised JDL-model [Steinberg and Bowman, 2001]

The definitions of the different levels in the JDL-model presented in Figure 3 are given below:

- **Level 0 — Sub-Object Assessment:**
  Estimation and prediction of signal- or object-observable states on the basis of pixel/signal-level data association and characterization.
- **Level 1 — Object Assessment:**
  Estimation and prediction of entity states on the basis of inferences from observations.
- **Level 2 — Situation Assessment**:
  Estimation and prediction of entity states on the basis of inferred relations among entities.

- **Level 3 — Impact Assessment:**
  Estimation and prediction of effects on situations of planned or estimated/predicted actions by the participants.
- **Level 4 — Process Refinement:**
  Adaptive data acquisition and processing to support mission objectives.

Continuously the focus will be on level 2, Situation Assessment, which involves finding relations and associations among objects, and how the results can be used in level 1 fusion.

Since level 2 data fusion involves finding relations between entities, some kind of representation of these objects and relations that also enables reasoning is needed. Ontologies and reasoning are therefore introduced in the following sections.

3.2. Ontology

The use of *ontologies* is a cornerstone in the situation assessment technology developed at Saab, thus this thesis project looks into how they can be used to represent a domain.

The word "ontology" is defined, according to Merriam-Webster online dictionary [Webster], as:

1.) a branch of metaphysics concerned with the nature and relations of being
2.) a particular theory about the nature of being or the kinds of things that have existence

This term has been borrowed in computer science to represent a set of concepts and the relationships between them. A widely used definition is *"An ontology is an explicit specification of a conceptualization*" [Gruber, 1993]. A conceptualization is an abstract, simplified view of the world that is to be represented. The Semantic Web group of W3C describes ontologies like this: "*Ontologies define the concepts and relationships used to describe and represent an area of knowledge. Ontologies are used to classify the terms used in a particular application, characterize possible relationships, and define possible constraints on using those relationships*" [Semweb].

A typical ontology consists of classes (concepts), individuals (instances) and properties. In order to be considered as an ontology, at least the following is required [McGuinness, 2002]:

- Finite controlled vocabulary
- Unambiguous interpretation of classes and term relationships
- Strict hierarchical subclass relationships between classes

Strict hierarchical means that if a class B is a subclass of class A, and a class C is a subclass of class B, then class C is also a subclass of class A. Ex: Mammal is a subclass of Animal and Lion is a subclass of Mammal, then Lion is also a subclass of Animal.

3.3. Ontology Languages

There are many languages to represent ontologies. This project has concentrated on a common ontology representation recommended by W3C, namely OWL. This section describes its predecessors in general, and OWL more in-depth.

3.3.1. RDF and RDF Schema

RDF, Resource Description Framework, is a W3C recommendation and a framework for describing resources on the web, such as title, author and modification date of a web page. RDF is written in XML and provides a model for data and a syntax so that applications, independent of each other, can exchange and use it. As stated previously, the minimum requirements for ontologies are finite controlled vocabulary, unambiguous interpretation of classes and strict hierarchical subclass relationships between classes. RDF does not fulfil this since it has no semantics to describe class-hierarchies.

RDFS, RDF Schema, is a knowledge representation language, providing the basic semantics needed to build ontologies. It is a W3C recommendation since 2004. RDFS extends RDF and makes it possible to build a simple hierarchy of classes, and a hierarchy of properties. Consider the following small ontology in Figure 4. It has three classes, which can be seen upon as sets of individuals.

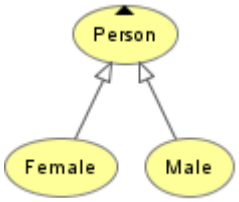


Figure 4

The class `Male` is a subclass of `Person`, which means that all Males are also Persons. We can assert further knowledge to the ontology by associating properties with the classes.

For example, we know that a person might have a sister. We can then add a property to the ontology; `hasSister`. In order to associate this with a class, we specify the property domain[1] and range. The domain of `hasSister` is set to "Person", and the range is set to "Female". This means, that if the knowledge that a person has a sister is asserted to the ontology, we know that the sister belongs to the class "Female".

With RDF Schema, it is possible to describe class hierarchies and associate properties with classes. It is however not possible to construct more complex expressions about classes, like stating that a person cannot be both a Female and a Male. These features allow us to build simple ontologies and perform simple inference.

3.3.2. OWL

OWL, Web Ontology Language, is a W3C recommendation for defining and instantiating web ontologies [Owlguide, 2004]. OWL extends RDF Schema and adds more vocabulary to describe classes and properties. It includes constructs such

[1] Properties with domain and range link individuals from the domain to individuals from the range. Example: `locatedAt(tank1, someBridge)` means "tank1 is locatedAt someBridge".

as disjoint classes, cardinality restrictions, characteristics of properties and enumerated classes.

OWL is categorised into three increasingly expressive sublanguages; *OWL Lite*, *OWL DL* and *OWL Full*, according to the following list, directly based on [Owlguide, 2004]:

- OWL Lite supports those users primarily needing a classification hierarchy and simple constraint features. For example, while OWL Lite supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives.
- *OWL DL* supports those users who want the maximum expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems. OWL DL includes all OWL language constructs with restrictions such as type separation (a class can not also be an individual or property, a property can not also be an individual or class). OWL DL is so named due to its correspondence with *description logic*s, which is the formal foundation of OWL.
- *OWL Full* is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. It is unlikely that any reasoning software will be able to support every feature of OWL Full.

Hereafter, descriptions about OWL in the report will apply to OWL-DL.

Main Features

The main components in OWL are Classes, Properties and Individuals. A *class* defines a group of individuals with similar characteristics. Classes are described using formal descriptions that state the requirements of the class. *Individuals* are instances of classes and thus represent objects in our domain. *Properties* are binary relations between individuals; they are used to enrich the class descriptions. Properties are further explained later on.

Class Descriptions

OWL provides basic building blocks to describe classes in different ways. A class description describes an OWL class by a class name or by specifying which individuals that belongs to it. There are six types of different descriptions [Owlref, 2004]:

1. A class identifier (i.e. a class name, mandatory in OWL-DL)
2. An exhaustive enumeration of individuals that together form the instances of a class
3. A property restriction
4. The intersection of two or more class descriptions
5. The union of two or more class descriptions
6. The complement of a class description

Property restriction describes an anonymous class of all individuals that satisfy the restriction. Restrictions can be of types Quantifier restrictions, Cardinality restrictions and hasValue restrictions.

Quantifier Restrictions are composed of a quantifier, a property and a filler. The quantifiers that can be used are the *existential* quantifier (∃), which in OWL is read as "someValuesFrom" and the *universal* quantifier (∀), which in OWL is read as "allValuesFrom".

The existential restriction, someValuesFrom, specifies the existence of a relationship for a property to an individual of a specific class. The example "∃ `hasWeapon Cannon`" consists of the existential quantifier, the property `hasWeapon` and the filler `Cannon`, and describes all of the individuals that have at least one weapon that is an individual from the class Cannon.

The universal restrictions, allValuesFrom, constrain the filler for a property to belong to a given class. The example "∀ `hasWeapon MachineGun`" consists of the universal quantifier, the property `hasWeapon` and the filler `MachineGun`, and describes all of the individuals whose hasWeapon property fillers are members of the class MachineGun. This means that *if* an individual with this restriction has a hasWeapon property, its filler must be of class MachineGun.

Cardinality Restrictions describes the number of relationships individuals of a class can have. We can use cardinality restrictions of type *minCardinality, maxCardinality* and *cardinality*. With the cardinality-restriction, the exact number of relationship an individual must participate in can be specified.

Class Axioms

Class axioms provide the ability to construct more detailed classes to perform reasoning on.

The following constructs are common:

- `subClassOf`
- `equivalentClass`
- `disjointWith`

`subClassOf` is defined in RDF Schema, but has the same meaning in OWL: If a class `B` is a `subClassOf` class `A` then the instances of `B` are also instances of `A`. We can now build more complex classes in combination with property restrictions (described above): If we let the class `Tank` be a subclass of `Vehicle` and a subclass of an anonymous class with the property restriction "∃ `hasWeapon Cannon`", then it is necessary for all individuals of class Tank to be a member of class `Vehicle`, and to have weapon `Cannon`. The axiom subClassOf thus, provides us with a tool to describe *necessary conditions* of a class. If a class, `Tank`, has the necessary condition of "∃ `hasWeapon Cannon`", it means that all instances that belongs to Tank must have a relation with an instance from the class `Cannon`, via the property `hasWeapon`.

The simplest form of an `equivalentClass` axiom is a statement of equivalence between two classes, for example, a statement that class `Automobile` is an `equivalentClass` to class `Car` means that all instances of `Automobile` is also instances of `Car`. With the `equivalentClass` axiom, it is possible to build classes

with *necessary and sufficient* conditions. Continuing with the class Tank, we know that to be a member of the Tank concept it is necessary to have cannon, but we might instead want to say that it is *sufficient* to have cannon in order to be a tank. In OWL, we would have to say that class `Tank` is equivalent with an anonymous class that has the restriction "`∃ hasWeapon Cannon`". This means that if an instance satisfies the restriction it is sufficient to determine that it is a member of the class Tank.

The `disjointWith` construct states that if a class `A` is `disjointWith` class `B`, then class `A` and class `B` cannot have any members in common. The statement "`Tank disjointWith Automobile`" asserts that members of Tank cannot be members of Automobile, and vice versa.

Properties and Property Characteristics

There are two main types of OWL-properties: Datatype-properties and Object-properties. Datatype properties specify characteristics of individuals by linking them to an XML Schema Datatype[2] value, for example string or integer. One example of a Datatype-property is "hasWheels"; linking the individual "aCar" to the data-literal "4" which is of type xsd:integer. Object properties differ from datatype properties in that it describes relations *between individuals* in an ontology. This can be for example "hasWeapon": linking the individual "aTank" to the individual "aCannon".

By default, an individual in owl can relate to any number of individuals or values. To provide means to reason about whether two individuals in an ontology could refer to the same individual or not, relevant constraint has to be added to the properties. In OWL, the use of *property characteristics* provides greater meaning to properties. Below are examples of some of the property characteristics available:

Functional Property: If a property is functional, an individual can only connect to maximum one other individual via that property. For example if hasMother is a functional property and the individual Carl hasMother Maria, and hasMother Sofia, a reasoner will infer that Maria and Sofia is actually the same individual. Note that if Maria and Sofia were stated to be different individuals, a reasoner will conclude the ontology to be inconsistent.

Transitive property: If a transitive property *p* relates individual *a* to individual *b*, and individual *b* to individual *c*, a reasoner can infer that individual *a* is also related to *c* via property *p*.

Symmetric property: If a property *p* is symmetric, it means that if individual *a* is related to individual *b* via *p*, a reasoner can infer that individual *b* is also related to individual *a* via *p*.

[2] http://www.w3.org/TR/xmlschema-2/

3.4. Description Logic and Reasoning

Description logics (DL) is a family of knowledge representation languages. The main building blocks of a DL are *concepts* and *roles*, where concepts denote sets of individuals and roles denote binary relationships between the individuals.

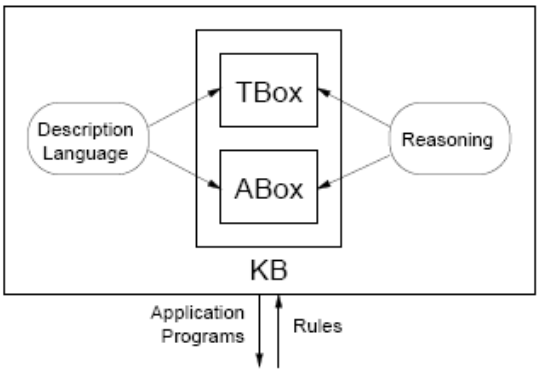


Figure 5: Architecture of a knowledge representation system based on description logic [Baader and Nutt, 2003]

Figure 5 shows a typical architecture of a knowledge representation system based on DLs. A knowledge base (KB) consists of two components, the TBox and the ABox. The TBox comprises the *terminology*; i.e. the knowledge about concepts and roles, the vocabulary. It is built by declarations that describe general properties of concepts. These declarations can be interpreted as sets of objects in a domain, for example; the concepts `Person` and `Female` denote the set of persons and females respectively. More complex concepts can be built by using constructors like intersection ($\prod$) and negation ($\neg$). The concept `Woman` can then be described like:

$$\texttt{Woman} \equiv \texttt{Person} \prod \texttt{Female}$$

The ABox contains the *assertional* knowledge, i.e. assertions about named individuals in terms of the concepts and roles in the TBox. For example,

$$\texttt{Female} \prod \texttt{Person(Lisa)}$$

$$\texttt{hasSibling(Lisa, John)}$$

states that the individual Lisa is a female person, and that Lisa has a sibling John.

A DL system does not only provide the ability of expressing taxonomies and store terminologies and assertions, but also offers services to *reason* about them. This means it is possible to infer additional knowledge that is not explicitly stated in the knowledge base. A typical reasoning task is *subsumption*, generally written $\texttt{C} \sqsubseteq \texttt{D}$. Determine subsumption is to check whether `D` is a more general concept than `C`, i.e. whether `D` subsumes `C`. Another reasoning task is to determine whether a concept description is *satisfiable*, which is the problem of verifying that a concept does not denote the empty concept (to check that the concept is non-contradictory). The basic reasoning task in an ABox is *instance checking*, which controls whether a given individual is an instance of a specific concept. Other common reasoning tasks in an ABox include *realization*, which finds the most specific concept an individual belongs to; and *retrieval*, which is to find all individuals of a specific concept. Both realization and retrieval can be achieved by means of instance checking.

4. Approach

With the techniques presented in last chapter as a starting-point, this section will suggest solutions to support an intelligence operator in associating incoming reconnaissance reports as well as a study of situation detection.

Before designing the ontology, it is important to have an understanding of the context in which to use the ontology. Figure 6 shows how information models in different levels of the system relate to each other. JC3IEDM (Joint Command, Control and Consultation Information Exchange Data Model) is a data model for interoperability and is defined by MIP[3]. The purpose of JC3IEDM is to enable sharing of data between systems to make it possible for different nations to collaborate. The JC3IEDM is generic and covers joint, land, sea, and air domain. Developers of systems that exchange information via the JC3IEDM can either use a subset of the data model that fits its purposes, or use its own data model and provide a mapping to and from JC3IEDM. Thus, the BMS in our case has its own information model that resembles the JC3IEDM. The IFE needs an ontology that is suited for reasoning but also enables mapping to the BMS information model.

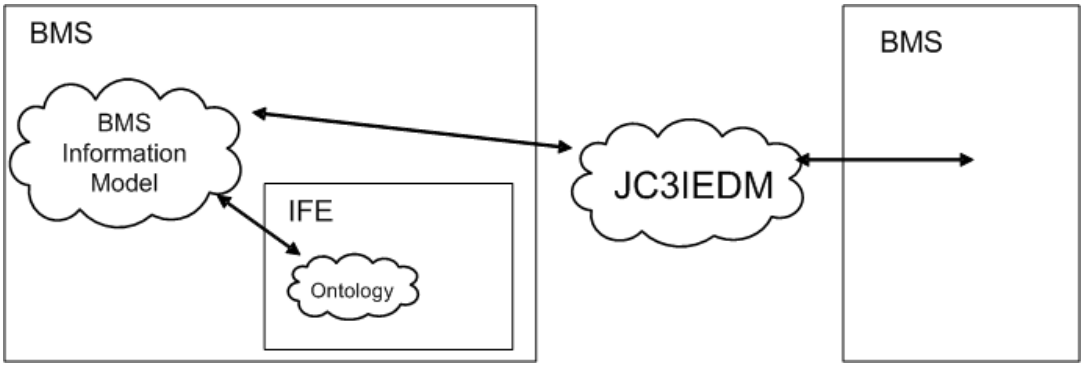


Figure 6: Information flow between different data models.

4.1. Building the Ontology

The design of this ontology has been preceded by investigating the JC3IEDM and the data model that is being used in the BMS system. Building an ontology for this kind of application requires extensive domain-knowledge, and would take to much time to acquire for a thesis project. Therefore, the ontology built for this thesis should not be seen as a complete example of how an ontology for the ground-target domain could look like, but a simplified prototype ontology adapted for the theories presented is this report.

First step in building the ontology is to define which classes to include. By studying the data models and a number of vehicles and weapon-systems that could be of interest for the scenario, a number of concepts were chosen for the ontology.

[3] MIP, Multilateral Interoperability Programme, is an international organisation which purpose is to achieve international interoperability between National Command and Control Information Systems. (http://www.mip-site.org)

Next step in the design is to arrange the classes in a hierarchy, (subclass-super class order). This was done by finding the most general classes and put them at the highest level in the hierarchy, and by finding the most specific classes and put them at the lowest level in the hierarchy. The ontology overview in Figure 7 shows this taxonomy, where you can see "`owl:Thing`" as the most general class. This owl construct represents the set that contains all individuals. All owl classes are subclasses of owl:Thing.

The most general class except owl:Thing is "`ResourceObject`" which in the BMS data model represents "An individually identified object that has military or civilian significance". Next to "`ResourceObject`" is the "`Attributes`"-class, which is constructed to be able to add relations to concepts (this will be described later on).

Two classes derive from "`ResourceObject`", `WeaponType` and `VehicleType`. In JC3IEDM, a vehicle type is defined as "*An EQUIPMENT-TYPE that is designed to operate on land routes (other than rail) with a primary role of transporting personnel, equipment or supplies*", and a weapon type is defined as "*An EQUIPMENT-TYPE of any kind used in warfare or combat to attack and overcome an enemy*".

As can be seen in Figure 7, "Tank" is defined to be a subclass of vehicle type instead of weapon type, which is the case in JC3IEDM. This is a consideration taken to fit the problem-domain in this project, where a reported military vehicle can turn out to be a tank, APC, or other vehicle.

When the taxonomy is defined, it is time to think about what attributes the different concepts shall have. This was also done by studying the different concepts included in the ontology and the BMS data model. Vehicles, for example, can have armour in varying degree, which is represented as Light, Medium, or Heavy armour. They can also either have wheels or tracks. All attributes included in the ontology are presented in Table 1.

To be able to describe concepts in a more interesting way, rather than just their names, we can add properties to the ontology so that we can build restrictions (Restrictions and properties are discussed in chapter "Ontology Languages"). Another step in the ontology building is to figure out what it is that define different concepts in the ontology. What it is that distinguishes an APC from a tank? As stated earlier; this is just a prototype-ontology, which means that the properties asserted to the classes is just for test-purposes and not the exact definitions of the different concepts.

After studying different kinds of Armoured Personnel Carriers, a few attributes seemed common for them: they have light armour, a machine gun as a weapon and are of medium weight.

Figure 7: Overview of the ontology

Property	Domain	Range	Characteristics
hasAttribute	ResourceObject	Attributes	
hasWeapon	ResourceObject	WeaponType	
hasArmour	ResourceObject	Armour	Functional
hasWeight	ResourceObject	Weight	Functional

Table 1

4.2. Association

4.2.1. Consistency checking

In [Kokar et al, 2004], a technique for finding associations among objects in an OWL ontology is demonstrated. This approach uses the OWL properties "sameAs" and "differentFrom" to set explicitly two individuals to refer to the same or to different individuals, and then use a consistency-checking tool to see if this construct caused the ontology to become inconsistent.

In our project, this technique has been used to find out which incoming reconnaissance reports that can be associated with existing objects in the situational picture. Incoming objects are associated with an already existing instance in the ontology, by adding the "sameAs"-statement. The reasoner then checks the ontology for any inconsistencies, and gives any of the following results:

- It is inconsistent for the two individuals to represent the same individual.
- It is consistent for the two individuals to represent the same individual.

If the association leads to the first result, we know that the two objects cannot be the same. If, on the other hand, the association does not lead to an inconsistency, the only thing we know is that the two objects *might* represent the same individual.

An ontology that includes little information about the classes, i.e. there is no property restrictions or other class-descriptions, all association-attempts with this technique will end up with the last result; that it is consistent for both statements to be true. To achieve results of value, the ontology has to include a rich set of properties and class expressions. Even though the ontology is well adapted to the domain and contains a rich set of class-descriptions, the method only supports the operator not to make incorrect associations, not the support of telling what associations are most likely. Most likely the result will be that two objects might or might not be referencing the same individual, and only occasional warn the operator of an incorrect association.

To provide better help for the operator to choose the right association among all the choices that might or might not be the same, we have looked into how ranking of the output could improve the result.

4.2.2. Ranking the results

The first step in ranking the results is to find a method to calculate similarity between objects in the ontology. A paper by Bizer et al [Bizer et al, 2005], describes how to use semantic matching to compare job descriptions and applicants' profiles based on their semantic similarity rather than keywords. They determine the similarity between two concepts in an ontology by calculating the distance between them. The definition below is used to calculate similarity:

$$sim_c(c_1, c_2) = 1 - dc(c_1, c_2),$$

where dc denotes the distance and c_1, c_2 two concepts in an ontology. The distance between two concepts is represented by the path from one concept to the other via the closest common parent. The model that is used to calculate the distance is

constructed in a way that semantic differences between concepts at a high level in the ontology is less similar than differences between concepts at a lower level. It also implies parent-child concepts have greater similarity than between siblings.

This approach might work well in an application with a simpler ontology without any class-descriptions and that only considers similarity between concepts. However, in our approach it is most likely that several incoming reports get the same classification, i.e. they will belong to the same concept, and they will thereby achieve a similarity measure of 1.

To take advantage of the rich ontology with class restrictions described in the previous chapter, we need a method that takes more than class membership into consideration when calculating a similarity measure.

Skoutas presents an approach to similarity measuring that takes both concept distance and class properties into consideration [Skoutas et al, 2007]. In order to rank the results, they take inspiration from two well-known evaluation methods from the area of Information Retrieval, *recall* and *precision*.

In Information Retrieval, precision is the number of relevant documents retrieved in proportion to all documents retrieved in search request, while recall is the number of relevant documents retrieved in a search request in proportion of all existing relevant documents. This means that if all retrieved documents from a search request are relevant, the precision score will be 1 (precision takes no consideration whether all relevant documents where retrieved). To get a recall score of 1 in a search request, all relevant documents must be retrieved, but it takes no consideration of how many irrelevant documents that were fetched.

A measure that combines the recall and precision to get a single value is the *weighted harmonic mean*, also known as the F-measure:

E 1
$$F_a = \frac{(1+a)\times precision \times recall}{a \times precision + recall}$$

Choosing a = 1, recall and precision have equal weight, while a > 1 weights recall more than precision.

Skoutas et. al takes these concepts from information retrieval and adapt them to the semantic similarity domain by comparing two classes based on their semantic description. Their approach is to calculate the recall and precision based on both the class hierarchy and class properties, and then combine the result to provide a single measure to the user.

Recall and Precision based on class hierarchy:

A common way of measuring similarity between concepts in a hierarchy is used, namely by the ratio of their common ancestors. Recall and precision are then calculated as follows, where $A(C)$ represents the set of super classes of a class C:

E 2
$$recall(C_1, C_2) = \frac{|A(C_1) \cap A(C_2)|}{|A(C_2)|} \quad precision(C_1, C_2) = \frac{|A(C_1) \cap A(C_2)|}{|A(C_1)|}$$

Recall and Precision based on class properties:

The ratio of common properties between two classes is used as a measure of their similarity. Recall and precision based on class properties are calculated as follows, where $P(C)$ represents the set of properties of a class C:

E 3 $$recall(C_1, C_2) = \frac{|P(C_1) \cap P(C_2)|}{|P(C_2)|} \quad precision(C_1, C_2) = \frac{|P(C_1) \cap P(C_2)|}{|P(C_1)|}$$

Calculating the similarity

We use the ideas presented by Skoutas et al and adapts them to fit the application presented in this report. Since all objects in the system are represented as individuals in the ontology, the similarity is measured between an incoming object and the individuals in the ontology that may be the same as the incoming object. The calculation of recall and precision in this project is as follows:

Recall is calculated as the proportion of items of the incoming object that are also items of the existing object.

Precision is calculated as the proportion of items of the existing object that are also items of the incoming object.

Recall and precision is calculated in regards to the properties associated with the individual, and with respect to the class hierarchy of the most specific class that the individual is a member of. A weighted mean is used to combine the recall and precision values. In this way, the parameters can be tweaked to fit the current domain. The equations for the combined values of recall and precision are given in E 4.

E 4 $$\overline{recall} = \frac{w_1 recall_c + w_2 recall_p}{w_1 + w_2} \quad \overline{precision} = \frac{w_1 precision_c + w_2 precision_p}{w_1 + w_2}$$

The F-measure is then calculated according to E 1, based on the results above.

An advantage of using recall and precision like this is that the similarity measure is represented by a continuous value in the range [0..1]. The domain of this project benefits from this since a large number of individuals may belong to the same class, but still differ from each other through their properties. Compared to the distance measure, where all of the individuals belonging to the same class would have received the same score, this technique provides better support for an operator.

4.3. Situation Detection

Based on the knowledge gained about OWL and reasoning, this section will study how these techniques can be used within situation detection.

As described in previous chapters, OWL gives the opportunity to build an ontology, assert knowledge with different class descriptions, populate it with individuals and perform reasoning to find inferred knowledge. Is it possible to use OWL as a tool for identifying situations? Can it be used to recognise that, for example, a city is to be attacked because hostile tanks have been detected at a road nearby? To model such knowledge, we need to be able to express relations that are more complex than described in previous chapters. Consider the relation `Fred hasUncle Kim`: it requires the knowledge of that Kim is male, has a sibling x, and x has a child Fred. There is no way to represent this with OWL. It is not possible in OWL-DL to infer any knowledge about complex implications, in which knowing a collection of facts implies some other facts.

Related work within situation detection is an ongoing research project at Saab Systems, which has resulted in the paper "Rule Based Situation Assessment for Sea-Surveillance" [Edlund et al, 2006].

A simple ontology and a scenario have been constructed to test how OWL and reasoning can be used within situation detection. Figure 8 shows an overview of the ontology with classes as ellipses and instances as squares.

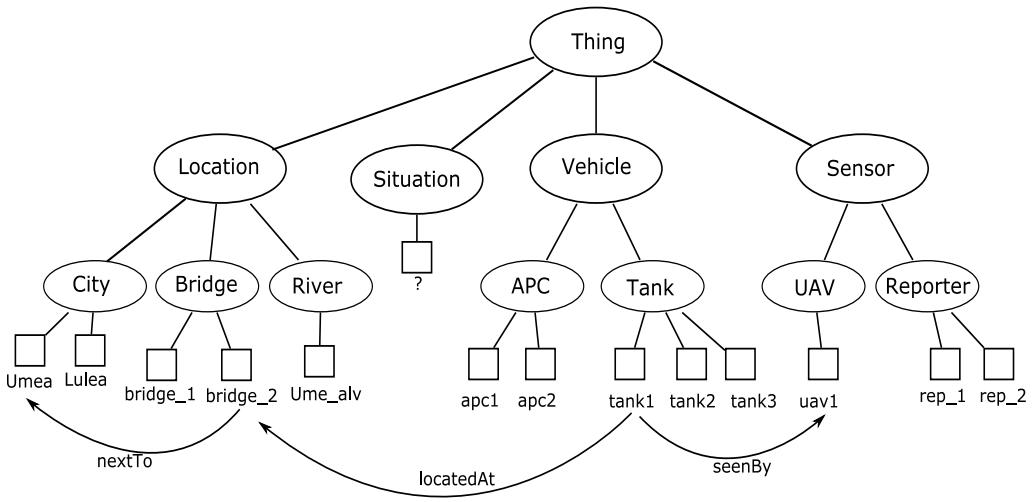


Figure 8 Overview of the ontology used for Situation Detection

In the ontology, the following constructs are used:

- locatedAt: domain[4]: Vehicle, range: Location
- seenBy: domain: Vehicle, range: Sensor
- nextTo domain: Location, range: Location

A simple scenario: A UAV has spotted a group of tanks on a bridge next to the city Umeå. This would indicate a situation that the city is under attack.

[4] Domain and range are described in *Ontology Languages.*

How can this situation be described with the ontology in Figure 8 and OWL? As stated earlier, OWL does not provide methods to assert knowledge to a class or instance by knowing a set of facts about other classes/instances.

Semantic Web Rule Language, SWRL is a rule language proposed to be a W3C standard for the Semantic Web. It extends OWL by adding implication in the form of if-then rules. This can be used to infer knowledge of the occurrence of a situation. To go back to the hasUncle example, a rule using SWRL would look like:

Person(?x) and Male (?y) and hasSibling(?y,?z) and hasChild(?z,?x) → hasUncle(?x, ?y)

When trying this rule in a rule engine, the engine asserted the hasUncle relation to the individual that satisfied the rule body (the antecedent part of the rule). Is it possible to build a rule that works for the scenario in the same way? We need to construct a rule that can describe that a city is under attack when hostile vehicles are seen nearby. A rule describing such a situation can be written like this:

Tank(?x) and Sensor(?y) and seenBy(?x, ?y) and locatedAt(?x, ?z) and nextTo(?z, Umea) → Situation(attack_in_umea)

The above rule will, however, not work in SWRL due to restrictions in the language that prevents having variables in the consequence part of the rule not present in the antecedent part of the rule. Another limitation of SWRL is that it is only possible to draw one conclusion out of a rule, i.e. only one operation can be performed in the consequence part of the rule.

To be able to use SWRL to identify a situation in our scenario, a workaround is needed. If the Situation class is removed, and instead an "isAttackedBy"-property is added to the City class, with domain "City", then a rule like the following can be defined:

Tank(?x) and Sensor(?y) and seenBy(?x, ?y) and locatedAt(?x, ?q) and nextTo(?z, Umea) → isAttackedBy(Umea, ?z)

When trying this in the rule-engine, it asserts the knowledge "isAttackedBy(Umea, tank1)" to the ontology. Hence, it is possible with a little workaround, to recognise situations with SWRL, even though it might not be an optimal solution.

New Approach

To investigate if SWRL can be used despite its limitations in a situation detection application, a new approach to the problem is tried.

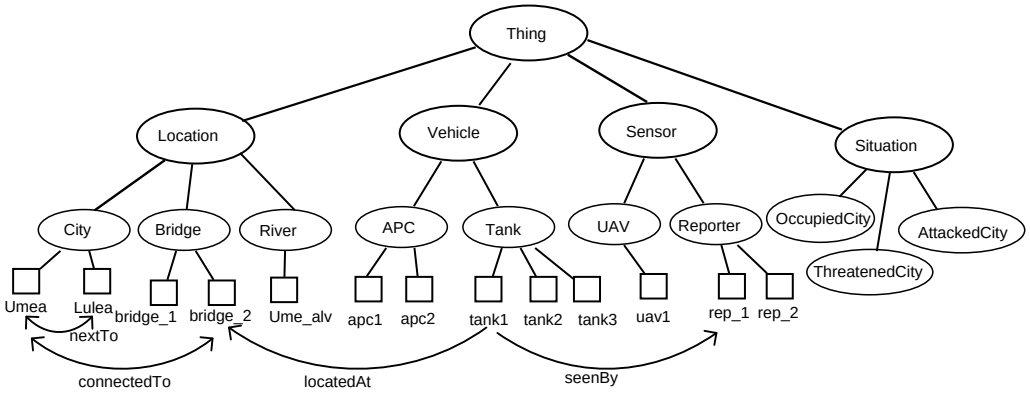


Figure 9 Overview of the ontology with situation sub classes added

The following changes have been made to the ontology:

- A new property is added, connectedTo, which links instances of locations to instances of cities.
- The Situation class is extended with three subclasses, "OccupiedCity", "AttackedCity" and "ThreatenedCity", which represents the situations that can be recognised.
- Two more cities is added as instances of the City class, Skellefteå and Övik, not visible in the figure

An *Occupied City* represents the situation where a tank is locatedAt a city.

An *Attacked City* represents the situation where a tank is at a location connectedTo a city.

A *Threatened City* represents a city that is nextTo a city that is either occupied or under attack.

Scenario:

Involves four cities; Luleå, (nextTo) Skellefteå, (nextTo) Umeå, (nextTo) Övik,

1.) A UAV has spotted a group of tanks on a bridge next to the city Övik. This indicates a situation that the city is under attack, "AttackedCity"
2.) An incoming reconnaissance report suggests that tanks are located in Luleå, which means that the city is occupied, "OccupiedCity"
3.) Cities next to cities that are either under attack or occupied are under threat, "ThreatenedCity"

Instead of using properties to represent situations, or creating new instances of Situation whenever a situation is recognised, instances of "City" are asserted to be members of a situation-class as well. In OWL, it is legitimate for an instance to be a member of more than one class.

The following SWRL-rules are constructed to test this new approach:

1.) Attacked City:

Tank(?x) and City(?y) and seenBy(?x,?q) and locatedAt(?x,?z) and connectedTo(?z,?y) → AttackedCity(?y)

2.) Occupied City:

Tank(?x) and City(?y) and locatedAt(?x,?y) and seenBy(?x,?z) → OccupiedCity(?y)

3.) Threatened City:

- City(?x) and City(?y) and nextTo(?x,?y) and AttackedCity(?x) → ThreatenedCity(?y)
- City(?x) and City(?y) and nextTo(?x, ?y) and OccupiedCity(?x) → ThreatenedCity(?y)

Results

The first rule resulted in that the city Övik was asserted to be a member of the class "AttackedCity". Rule number two made the city Luleå a member of the "OccupiedCity" class. Rule number three resulted in that the cities Skellefteå and Umeå were asserted to be members of the class "ThreatenedCity".

With this method, situations are recognised without violating the SWRL-restrictions. It also seems more straightforward to have the situations as classes, and whenever a situation is identified, it is asserted to be a member of the corresponding Situation-class.

What situations cannot be represented?

With SWRL, it is possible to recognise all situations that can be described by knowing a set of facts. It is when information that is more complex is needed to recognise a situation that SWRL is problematic. Suppose that the situation "AttackedCity" is re-defined to be a situation where a set of tanks located next to a city is moving synchronised towards it. This cannot be described directly by SWRL since it is not possible to represent information *about* a relation.

W3C introduced SWRL as a draft recommendation in 2004. Lately, some support for SWRL has been included in several reasoners. Pellet, further described in section 5.1, has a preliminary implementation of reasoning with DL-safe rules encoded in SWRL.

The reason for this late introduction of support for SWRL might be that the limitations make the language less attractive to developers. To be able to benefit from SWRL in situation assessment some modifications to the language is desirable, like letting the consequence part contain variables not present in the antecedent part (dl-safe rule).

5. Implementation

A prototype has been developed to test the ideas presented in Chapter 4 . This chapter describes this application and the set of tools needed for the implementation.

5.1. Choosing a reasoner

A reasoner is used to classify objects and find inconsistencies in the ontology. There exist several reasoners for dl reasoning on the market, both commercial and free open-source. To find out which one to use, a small investigation was carried out. The following properties were important for the decision:

- Ability to reason in OWL-DL
- Handle reasoning with individuals
- Support the properties, class-constructs and data-types used in this project
- Well-documented API and good online support
- Free to use

The reasoners that were further investigated were Pellet, KAON2 and RacerPro.

KAON2:

KAON2[5] is a free-of charge (for non-commercial academic use) reasoner implemented in Java. KAON2 cannot handle enumerated classes, i.e. the constructs oneOf and hasValue.

RacerPro

RACER[6] stands for "Renamed ABox and Concept Expression Reasoner". RacerPro is a knowledge representation system that offers reasoning and can process OWL-DL ontologies. It is not free-of-charge, but there is a free time-limit license for higher education and research.

Pellet

Pellet[7] is an open-source, OWL-DL Java reasoner originally developed at the University of Maryland's Mindswap Lab. The development has now shifted to Clark & Parsia, which also offers commercial support and customization. It supports the full expressivity of OWL-DL, including reasoning with enumerated classes.

Our choice was Pellet, which seemed like the reasoner most suited for this project; it is free and supports the features needed. It also has multiple interfaces to the reasoner, datatype reasoning and an active mailing list for support. Furthermore, Pellet seemed to be the easiest reasoner to get started with.

[5] http://kaon2.semanticweb.org/
[6] http://www.racer-systems.com/
[7] http://pellet.owldl.com/

5.2. OWL Tools

In addition to the reasoner, a set of tools related to OWL was used to facilitate the development of the prototype. A graphical OWL-editor for building the ontology provides good overview and eases creation of classes and class-descriptions. An API to access the ontology programmatically is essential when building an application based on ontological reasoning.

5.2.1. OWL Editor

Protégé[8], a graphical editor for ontologies, was chosen for building the ontology. Protégé is free open source, written in Java and was developed by Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. It provides support for building and editing owl-ontologies and includes tools for easy creation of class-descriptions. It also supports the installation of a number of existing plug-ins, as well as possibilities for developing new ones. Extensions that were used in this project included a plug-in to edit and test SWRL-rules, and a plug-in for visualization.

5.2.2. Ontology API

There exist several APIs and implementations of OWL. Examples are OWL-API, Protégé-owl API and Jena. They all provide methods and classes to load, query and perform reasoning services on OWL ontologies. We picked the Jena API, which is a framework for building semantic web applications; it has support for RDF, RDFS and OWL. Pellet offers an interface to Jena, and thus the two applications work very well together. Jena is free, open-source, written in Java and has documentation on its website.

5.2.3. Rule Engine

To test the rules written in SWRL, presented in the Situation Detection chapter, a rule-engine that can handle SWRL is needed. Jess[9] is a rule-engine entirely written in Java, and has its own rule-language. To use Jess, a translation between SWRL and the Jess-language is required. There is a plug-in to Protégé, SWRLJessTab, which supports the execution of SWRL rules using the Jess rule engine.

[8] http://protege.stanford.edu
[9] http://www.jessrules.com/

5.3. Application Overview

Figure 10 shows an overview of the prototype application.

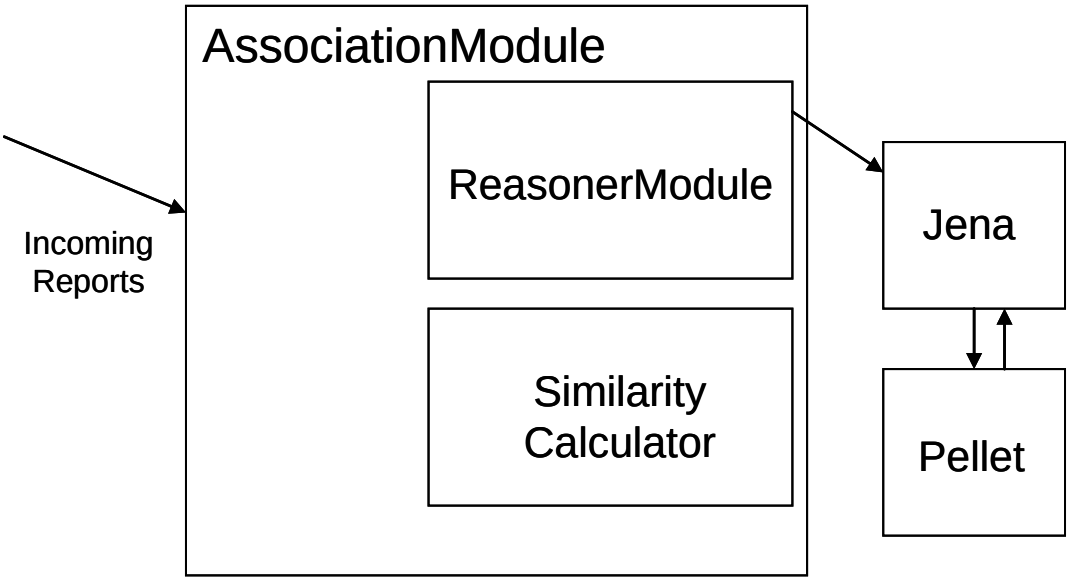


Figure 10 Overview of the application

Input

At start-up, the application loads the ontology from a specified owl-file. During run-time, incoming reconnaissance reports are added as individuals to the ontology, in this prototype the reports are simulated data.

Reasoner Module

The reasoner module is responsible for all reasoning performed by the system. It uses Jena to represent the ontology, and Pellet for the reasoning activities. For every incoming individual to the system, the reasoner module adds the individual and all known attributes to the ontology. A classification for the newly added object is then performed and returned to the system. The reasoner module is also responsible to keep the ontology consistent, i.e. to prevent individuals to be associated with each other if they are inconsistent.

Similarity Calculator

The responsibility of the similarity calculator is to calculate the similarity between an incoming individual and all existing individuals in the ontology. It uses the reasoner module to retrieve class membership and super classes of an individual as well as checking for inconsistencies between individuals. A list of ranked similarity results are returned by the module

Output

Output from the association module is a sorted list of the results for an incoming report and information about what objects that are inconsistent to be the same as the incoming report.

5.4. Consistency Checking

The consistency checking, described in section 4.2.1, is performed between an incoming object and all existing objects in the knowledge base. Before calculating the similarity between two objects, a consistency check is done by using the owl statement "sameAs".

Example 1

	Incoming Object (IO):	Existing Object (EO):
Type:	VehicleType	VehicleType
	ResourceObject	ResourceObject
Properties:		
	-hasWeight MediumWeight	-hasWeight LightWeight

In Example 1, when checking the consistency, the reasoner module will add the statement "IO sameAs EO" to the ontology. The reasoner identifies that an inconsistency has occurred, since an object cannot have both light and medium weight. The EO object will be marked as "Not same as" to the IO object, and the "sameAs" statement will be removed. In this prototype application, a similarity calculation is performed regardless of the consistency result, but the output separates the inconsistent objects. If no inconsistency is detected, the "sameAs" statement will still be removed, since it is not certain that two objects are the same just because it is consistent to be so. The purpose of the consistency checking is to prevent an association between inconsistent objects.

A consistency check is also done when an object is added to the knowledge base. This is to prevent adding of objects are inconsistent with the loaded ontology. For example, adding of a tank with light weight.

5.5. Similarity Calculation

The similarity between two individuals is calculated according to the description of recall and precision section 4.2.2. Below is a description of how the similarity is calculated between an incoming object and an already existing object in the system.

Example 2

	Incoming Object (IO):	Existing Object (EO):
Type:	ArmouredPersonnelCarrier	
	VehicleType	VehicleType
	ResourceObject	ResourceObject
Properties:		
	-hasArmour LightArmour	

```
        -hasWeapon Machine-Gun     -hasWeight MediumWeight
        -hasWeight MediumWeight    -hasAttribute Tracked
-------------------------------------------------------------
```

Recall and Precision based on properties:

Recall between two individuals is calculated as the *Proportion of properties of the incoming object that are also properties of the existing object.*

The precision is calculated as the *Proportion of properties of the existing object that are also properties of the incoming object.*

In the example above, which has one property-match, the recall will be 1/3 and the precision will be 1/2.

Recall and Precision based on class-hierarchy:

Recall between two individuals based on the class-hierarchy is calculated as *Proportion of the classes that the incoming object are member of that also the existing object are member of.*

The precision is calculated as the *Proportion of the classes that the existing object are member of that also the incoming object are member of.*

In the example, the incoming object is of type ArmouredPersonnelCarrier and the existing object is of type VehicleType. Figure 7 (p. 16) shows an overview of the ontology, where it can be concluded that the number of matches between the incoming object and the existing object are two; ResourceObject and VehicleType (owl:Thing is not included as a match, since all classes in an owl ontology inherits from owl:Thing). This means that the recall will be 2/3, since the incoming object are member of three classes, and the precision will be 2/2 = 1, since all classes that the existing object is a member of are also included in the parent-list of the incoming object.

Calculating the Combined F-measure

First step in achieving the combined F-measure is to combine the results from the class hierarchy and the property calculations. This is done by using a weighted mean, see E 4 for equation. For this project, recall and precision based on class-hierarchy has a lower weight than calculations based on properties. This is to prevent the many incoming reports classified as "VehicleType" to receive high scores, since they will share the same ancestors. The F-measure is then calculated according to the equation E 1. In this project, recall and precision have equal weight, and a is therefore set to 1.

To return to the example, the combined recall value will be $\frac{0.7 \cdot 2/3 + 1.3 \cdot 1/3}{0.7 + 1.3} = 0.45$ using a weight of 0.7 for recall based on class hierarchy and a weight of 1.3 on recall based on common properties. Consequently, the combined precision value will be $\frac{0.7 \cdot 1 + 1.3 \cdot 1/2}{0.7 + 1.3} = 0.68$.

With these values, the F-measure is calculated as: $\frac{(1+1) \cdot 0.68 \cdot 0.45}{(1 \cdot 0.68) + 0.45} = 0.54$

6. Results

This chapter presents the results when testing the implementation on the scenario from the problem description. The test reflects the three steps in the scenario, seen in Figure 2. Comments to the results are included in the end of the chapter.

6.1. Test Run

Starting criteria is the ontology according to section 4.1, *Building the Ontology*. Abbreviations are explained in the table below.

Label	Description
Recall_c	Recall based on class-hierarchy
Prec_c	Precision based on class-hierarchy
Recall_p	Recall based on properties
Prec_p	Precision based on properties
Comb_recall	Combined Recall
Comb_prec	Combined Precision
F-Measure	F-measure

Input Step 1:

The reports from the UAV are added to the ontology as individuals:

ID	Type	Attributes:
uav1	VehicleType	hasWeapon Machine-Gun
uav2	VehicleType	hasWeapon Cannon hasWeapon Machine-Gun

Result step 1:

```
Classification of uav1: VehicleType
Classification of uav2: VehicleType
```

Input Step 2:

The report from the Ground Sensor Network is added as:

ID	Type	Attributes:
gsn1	VehicleType	hasWeight LightWeight hasAttribute Wheeled

Results step 2:

```
Classification of gsn1: Automobile

Association Results for individual: gsn1
```

ID:	Recall_c	Prec_c	Recall_p	Prec_p	Comb_recall	Comb_prec	F-measure
uav2	0,67	1,00	0,00	0,00	0,35	0,23	0,28
uav1	0,67	1,00	0,00	0,00	0,35	0,23	0,28

Input Step 3:

Reports from the Reconnaissance Troop, the Ground Sensor Network and the Tank Commander are added as:

ID	Type	Attributes:
rec	ArmouredPersonnelCarrier	hasWeight MediumWeight hasWeapon Machine-Gun hasArmour LightArmour
gsn2	VehicleType	hasWeight HeavyWeight hasAttribute Tracked
tankcmdr	Volvo-XC90	hasWeight LightWeight hasAttribute Wheeled

Results Step 3:

```
Classification of rec: ArmouredPersonnelCarrier
```

Association Results for individual: rec

ID:	Recall_c	Prec_c	Recall_p	Prec_p	Comb_recall	Comb_prec	F-measure
uav1	0,67	1,00	0,33	1,00	1,00	0,45	0,62
uav2	0,67	0,67	0,33	0,33	0,45	0,45	0,45
Not sameAs: {gsn1	0,67	0,67	0,00	0,00	0,23	0,23	0,23 }

Additional Input to Step 3:

The operator chooses to associate `rec` with `uav1`, thus all attributes belonging to `uav1` are added to `rec` and `uav1` is removed from the knowledge base.

Classification of gsn2: VehicleType
Association Results for individual: gsn2

ID:	Recall_c	Prec_c	Recall_p	Prec_p	Comb_recall	Comb_prec	F-measure
uav2	1,00	1,00	0,00	0,00	0,35	0,35	0,35
uav1	1,00	1,00	0,00	0,00	0,35	0,35	0,35
Not sameAs: {gsn1	1,00	0,67	0,00	0,00	0,23	0,35	0,28 }
Not sameAs: {rec	1,00	0,67	0,00	0,00	0,23	0,35	0,28 }

Classification of tankcmdr: Volvo-XC90

Association Results for individual: tankcmdr

ID:	Recall_c	Prec_c	Recall_p	Prec_p	Comb_recall	Comb_prec	F-measure
gsn1	0,75	1,00	1,00	1,00	1,00	0,91	0,95
uav2	0,50	1,00	0,00	0,00	0,35	0,18	0,23
Not sameAs: {gsn2	0,50	1,00	0,00	0,00	0,35	0,18	0,23 }
Not sameAs: {rec	0,50	0,67	0,00	0,00	0,23	0,18	0,20 }

Comments to test runs:

Step 1: The result from the classification is “VehicleType”. Not enough information is known to make a better classification. The association-module does not return anything in this step, since no objects exists in the knowledge base.

Step 2: Step 2 includes the report of the light vehicle from the ground sensor network. The two attributes “lightweight” and “wheeled” gives enough information for the reasoner to classify the object as “Automobile”. The association-module compares the incoming object to the ones already in the system (uav1 and uav2 from step 1). They obtain a score of 0.28, which is not high enough to associate them with the gsn1 object.

Step 3: In the last step, three incoming reports are fed to the reasoner module. The “rec” object from the reconnaissance troop is added as an ArmouredPersonnelCarrier directly, and is therefore classified as such. All necessary attributes are added automatically since the knowledge of what class an individual is a member of infers knowledge of the necessary attributes of that class. The association-module returns that “rec” cannot be the same object as “gsn1” as they have different weight. The ranking result is a score of 0.62 for the uav1 and 0.45 for uav2. The operator chose to associate rec with uav1, which means that they are seen as the same object henceforward. The association does also lead to a merge of the two items, meaning that all attributes of uav1 are added to rec, and rec is removed from the knowledge base. An interesting point in this step is that the reasoner concludes the gsn1 not to be the same as rec, because of their disjoint weight attributes.

The “gsn2” object from the ground sensor network is classified as “VehicleType”, since the only information given is its weight and that it is tracked. The output from the association modules is that gsn2 cannot be the same object as gsn1 or rec, this would lead to inconsistency since an individual cannot have both heavy and medium weight. The ranking result is a 0.35 score for both uav1 and uav2.

The “tankcmdr” object is reported as a “Volvo XC-90”, and is therefore classified as a “Volvo XC-90”. It is inferred not to be same as gsn2 and rec, which depends on their different weights. The ranking score is 0.95 for gsn1, meaning they are very similar.

7. Conclusions

7.1. Achievements

This thesis have explored the possibilities to use ontological reasoning to support an intelligence analyst with the association of incoming reconnaissance reports. To gain further knowledge within the area, common techniques in the field have been studied, including ontology and ontology languages like OWL, description logic and reasoning. An approach that uses a reasoner to check for inconsistencies in the ontology to retrieve valid association-objects is presented. Concepts from information retrieval, recall and precision, is used to rank the valid objects based on how similar they are to the incoming object. Thus, an operator can be offered extended support in decision-making. In addition to investigate the support for association, a study of how the techniques of ontological reasoning can be used within the field of Situation Detection is included.

A prototype application was developed to enable testing of the theories presented in the approach. The implementation included input of an owl ontology, adding of individuals, inconsistency checking and similarity calculation between an incoming object and the existing objects.

The result from the test runs shows that it is possible to find the objects in the knowledge base that are inconsistent to be associated with an incoming report. This can prevent an operator to associate, as in the scenario, a tank with an automobile. Furthermore, the result shows a working classification of objects added to the knowledge base. The output does also supply a ranked list from the similarity calculation, whose result is more complex to interpret. While some of the incoming objects do achieve a high ranking score to similar objects, the ontology and scenario used in this project are too small and simple to be able to draw any real conclusions. Testing with richer ontologies and larger knowledge base is needed.

7.2. Discussion and conclusions

Even though it is hard to come to any conclusions about the ranking methods, there are several cases that an association module, especially the inconsistency checking, can provide support for decision-making. In combination with other tools, like kinematical reasoning[10], the results could be even better. Ontological reasoning can prevent an operator to associate two inconsistent objects that the kinematical reasoner would suggest, and the other way around; two ontological similar objects might not be feasible from a kinematical point of view. An aspect that is not taken into consideration in this project is whether the incoming reconnaissance reports are correct or not. In our calculation, we consider that all incoming data is correct. The possibility to reason with uncertainties is desirable. Recently, probabilistic reasoning is starting to find its way to the reasoners. Pellet has introduced probabilistic reasoning by *Pronto*[11], an extension to

[10] Discussed in 2.1
[11] More info on http://pellet.owldl.com/pronto/

Pellet that enables reasoning services for knowledge bases containing uncertain knowledge. Pronto is to a large extent an implementation of the ideas in [Lukasiewicz].

Difficulties in this thesis project has been to build a suitable ontology for the domain, and to decide what attributes that specifies a certain class and holds for all individuals of that class. Some properties are valuable though, like the weight attribute that can certainly separate different kinds of vehicles when using sensors like Ground Sensor Networks. To clarify the adequacy of using recall and precision as similarity calculation in this domain, more testing is needed. We do believe that ontological reasoning in combination with other tools gives benefits to an operator working with association of intelligence reports in a BMS.

The situation detection study concludes that a rule language like SWRL is needed to recognise situations in an ontology, since OWL cannot hold any knowledge about complex implications, in which knowing a collection of facts implies some other facts. A problem with SWRL is that it has several limitations that makes it less advantageous for situation assessment if not some modifications is done. It do however exist rule-engines that ignores if the rules are valid SWRL or not, which makes it possible to adjust SWRL to fit a situation assessment application.

7.3. Future Work

This project is just a first glimpse of how ontological reasoning techniques can be used within the association work in a BMS. A demonstrator that can be fully integrated in a Intelligence Fusion Engine is needed to be able to perform extensive testing and evaluation of the ideas presented in this report. Adding of probabilistic reasoning is also desirable. Effort to build an ontology best suited for this kind of reasoning task is also needed.

8. Acknowledgements

I would like to thank my supervisor at Saab Systems, Klas Wallenius, for all help throughout the project, especially for all support and valuable feedback during the writing phase. Thanks also to Andreas Lingvall, at the time part of the Data Fusion team at Saab Systems, for giving me the opportunity to work with this project and Johan Edlund, also in the Data Fusion Team, for great discussions in the early phase of the project.

I would also like to thank:

Michael Minock at the department of Computing Science, Umeå University, for supervising this master thesis and for giving feedback on my writing.

My colleagues in the Data Fusion team at Saab Systems for encouragement and inspiration.

My family and friends, for all support and encouragement to finish this thesis.

9. References

[Semweb] Semantic Web, http://www.w3.org/2001/sw/, last visited 2007-09-28

[McGuinness, 2002] D. McGuinness, "Ontologies Come of Age", in D. Fensel, J. Hendler, H. Lieberman and W. Wahlster, editors, Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, MIT Press, 2002

[Skoutas et al, 2007] D. Skoutas, A. Simitsis, T. K. Sellis, "*A Ranking Mechanism for Semantic Web Service Discovery*" In Proceedings of the 4th International Workshop on Semantic Web for Services and Processes (SWSP '07), in conjunction with the 2007 IEEE International Conference on Web Services (ICWS '07), Salt Lake City, Utah, USA, July 9-13, 2007

[Gruber, 1993] T. Gruber, "*A Translation Approach to Portable Ontology Specifications*" Appeared in Knowledge Acquisition, **5**(2):199-220, 1993.

[Kokar et al, 2004] M Kokar, C. Matheus, J Letkowski, K Baclawski, P Kogut, "*Association in Level 2 fusion*" In Proceedings of SPIE Conference on Multisensor, Multisource Information Fusion, Orlando, FL., April 2004

[Steinberg and Bowman, 2001] A. N. Steinberg, C. L. Bowman. "Revisions to the JDL Data Fusion Model", in D. L. Hall, J Llinas, editors, Handbook of Multisensor Data Fusion, CRC Press, New York 2001.

[Webster] Merriam-Webster Online Dictionary, http://www.webster.com/, last visited 2007-09-21

[Owlguide, 2004] OWL Web Ontology Language Guide, W3C Recommendation 10 February 2004, Editors: Michael K. Smith, Chris Welty, Deborah L. McGuinness, http://www.w3.org/TR/owl-guide/ last visited: 2007-09-25

[Owlref, 2004] OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004, Editors: Mike Dean, Guus Schreiber, Authors: Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, Lynn Andrea Stein, http://www.w3.org/TR/owl-ref/ last visited 2007-09-25

[Baader and Nutt, 2003] F. Baader, W. Nutt. “Basic Description Logics”, in F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider, editors, The Description Logic Handbook, Cambridge University Press, Cambridge 2003

[Bizer et al, 2005] C. Bizer, R. Heese, M. Mochol, R. Oldakowski, R. Tolksdorf, “*The Impact of Semantic Web Technologies on Job Recruitment Processes*”, in Proceedings of the 7th International Conference Wirtschaftsinformatik, Bamberg, Germany, February 2005

[Edlund et al, 2006] J. Edlund, M. Grönkvist, A. Lingvall, E. Sviestins, ”*Rule Based Situation Assessment for Sea-Surveillance*”, In Proceedings of SPIE Conference on Multisensor, Multisource Information Fusion, Orlando, FL, April 2006

[Lukasiewicz] T. Lukasiewicz, “*Probabilistic Description Logics for the Semantic Web*”, Research Report, March 2007