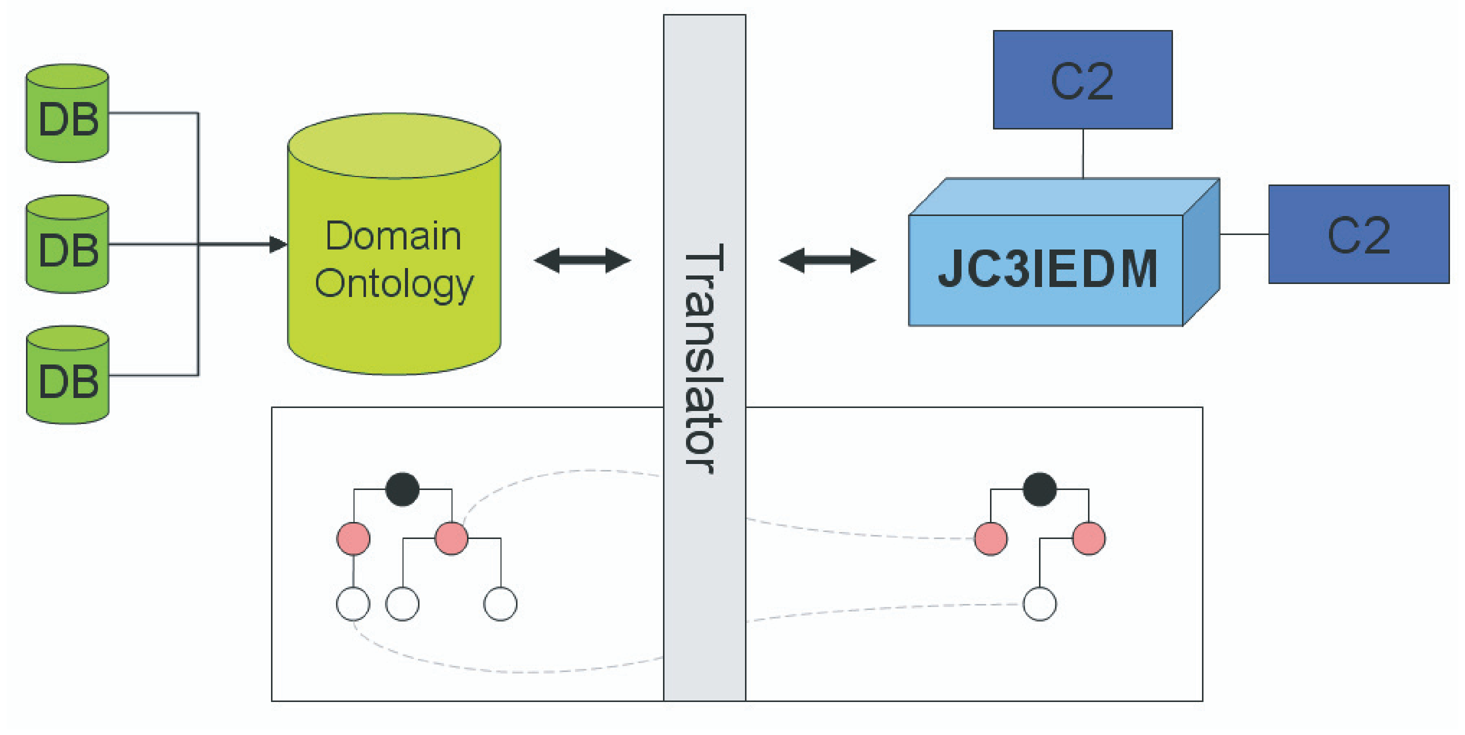
Ontological Interoperability

MARTIN EKLÖF, CHRISTIAN MÅRTENSON



FOI-R-- 1943 --SE Methodology report
ISSN 1650-1942 January 2006

Command and Control Systems

FOI
Defence Research Agency
Command and Control Systems
P.O. Box 1165
SE-581 11 Linköping

Phone: +46 13 37 80 00
Fax: +46 13 37 81 00

www.foi.se

Martin Eklöf, Christian Mårtenson

Ontological Interoperability

FOI-R--1943--SE Methodology report Command and Control Systems
ISSN 1650-1942 January 2006

Issuing organization FOI – Swedish Defence Research Agency Command and Control Systems P.O. Box 1165 SE-581 11 Linköping	**Report number, ISRN** FOI-R--1943--SE	**Report type** Methodology report
	Research area code 4. C4ISTAR	
	Month year January 2006	**Project no.** E79306
	Sub area code 41 C4I	
	Sub area code 2	
Author/s (editor/s) Martin Eklöf Christian Mårtenson	**Project manager** Tobias Horney	
	Approved by Martin Rantzer	
	Sponsoring agency FMV	
	Scientifically and technically responsible Per Svensson	
Report title Ontological Interoperability		

Abstract

Today, interoperability is a challenge for the Swedish Armed Forces (SwAF) and will probably remain so considering the increased engagement in international operations. In order to facilitate interoperability, Sweden participates in the Multilateral Interoperability Programme (MIP). MIP is a collaboration between several nations and manages the development of a common information exchange data model for the C2 arena, named JC3IEDM. In order to enable delivery of required information to the C2 domain, it is necessary that systems are compliant with the JC3IEDM. This can be accomplished by integration of domain-specific ontologies with the JC3IEDM.

This study explores the fundamentals of ontology integration, involving activities such as identification of suitable mappings between multiple ontologies, and automated translation of information between heterogeneous representation formats. Also, this study covers high level perspectives of ontology integration, including ontology management and tool support. The area of ontology integration has been explored in a multitude of research projects world-wide, and it has close connections with the area of database schema integration. However, more detailed studies and practical experiences are required in order to determine which methods and techniques are suitable for use within SWaF, in enabling semantic interoperability between various domains and the C2 arena (through the JC3IEDM).

Keywords
Interoperability, Ontology, JC3IEDM

Further bibliographic information	**Language** English
ISSN 1650-1942	**Pages** 57 p.
	Price acc. to pricelist

Utgivare FOI - Totalförsvarets forskningsinstitut Ledningssystem Box 1165 581 11 Linköping	**Rapportnummer, ISRN** FOI-R--1943--SE	**Klassificering** Metodrapport
	Forskningsområde 4. Ledning, informationsteknik och sensorer	
	Månad, år Januari 2006	**Projektnummer** E79306
	Delområde 41 Ledning med samband och telekom och IT-system	
	Delområde 2	
Författare/redaktör Martin Eklöf Christian Mårtenson	**Projektledare** Tobias Horney	
	Godkänd av Martin Rantzer	
	Uppdragsgivare/kundbeteckning FMV	
	Tekniskt och/eller vetenskapligt ansvarig Per Svensson	
Rapportens titel Ontologisk Interoperabilitet		

Sammanfattning

Interoperabilitet är en utmaning för dagens Försvarsmakt och kommer med stor sannolikhet att vara en svår fråga även i framtiden, speciellt med tanke på Sveriges ökade engagemang utomlands. För att skapa förutsättningar för interoperabilitet så deltar Sverige i Multilateral Interoperability Programme (MIP), ett samarbete mellan ett flertal länder som syftar till att ta fram gemensamma datautbytesmodeller för ledningsområdet. Den senaste produkten från MIP är JC3IEDM. I syfte att försörja ledningssystem med relevant information från skilda domäner är det viktigt att system är kompatibla med JC3IEDM. Ett sätt att uppnå detta mål är att integrera domänspecifika ontologier med JC3IEDM.

Detta projekt har utforskat grundläggande förutsättningar för integrering av ontologier, i form av aktiviteter för identifiering av överlappande koncept och relationer, samt automatiserad översättning av information mellan skilda representationssätt. Vidare har även mer övergripande perspektiv belysts såsom ”ontology management” och olika former av verktygsstöd. Det finns mycket forskning relaterad till integrering av ontologier och området har även stark koppling till integrering av databaser. Givet de verktyg och tekniker som existerar krävs det dock vidare arbete, framförallt praktisk tillämpning, för att avgöra vilka lösningar som kan tillämpas inom Försvarsmakten i syfte att uppnå semantisk interoperabilitet.

Nyckelord

Interoperabilitet, Ontologi, JC3IEDM

Övriga bibliografiska uppgifter	**Språk** Engelska
ISSN 1650-1942	**Antal sidor:** 57 s.
Distribution enligt missiv	**Pris: Enligt prislista**

Abstract

Today, interoperability is a challenge for the Swedish Armed Forces (SwAF) and will probably remain so, especially considering the increased engagement in international operations. Future Command & Control (C2) systems will require a high level of interoperability, meaning that information from disparate domains should be consumable in a seamless and automated fashion. In order to facilitate interoperability, Sweden participates in the Multilateral Interoperability Programme (MIP), a collaboration between several nations that manages the development of a common information exchange data model for the C2 arena named JC3IEDM[1]*. The purpose of the JC3IEDM is to enable interoperability in the context of multinational, combined and joint operations. Given this, it is crucial that systems are compliant with the JC3IEDM in order to enable delivery of required information to the C2 domain. However, it is not desirable that individual systems use the JC3IEDM for their internal representation of information, since this most probably must meet other types of requirements concerning level of abstraction and coverage. Individual domains will still require use of their established standards, data and information models.*

Traditionally, systems integration has mainly targeted the syntactic level and focused less on the semantics of information. Given a dynamic and complex environment, where systems from different domains exchange information in a seamless fashion, the inherent meaning of information must be considered as well. It is important that the semantics of information is defined explicitly and in a way that is accepted by all involved parties. An ontology is a formal specification of a common conceptualization of a given domain and is used to express information and knowledge in an unambiguous way. Through integration of ontologies, semantic interoperability can be obtained across several domains, which means that exchanged information is interpreted in an intended way and ambiguities are avoided.

This study explores the area of ontology integration for the purpose of enabling ontologies from diverse domains to be interoperable with the JC3IEDM. The fundamentals of ontology integration is described in the form of processes for ontology integration, comprising steps such as identification of suitable mappings between a set of ontologies and automated translation of information between heterogeneous representation formats. Also, this study explores high level perspectives of ontology integration, including ontology management and tool support. The area of ontology integration has been explored in a multitude of research projects world-wide. Furthermore, it has close connections with the area of schema integration, which has been explored extensively for the last decades within the database community. However, more detailed studies and above all practical experiences are required in order to determine which methods and techniques are suitable for use within SwAF, in enabling semantic interoperability between various domains and the C2 arena (JC3IEDM).

[1] Joint Command Control & Consultation Information Exchange Data Model

Table of Contents

1. Introduction

Today, interoperability of systems is a challenge for the Swedish Armed Forces, and given the increased engagement in operations on an international scale, interoperability will be even harder to reach and maintain in the future. In order to enable interoperability at an international level, the Swedish Armed Forces participates in the Multilateral Interoperability Program (MIP) [MIP] developing common information exchange data models for the C2 (Command & Control) domain. The latest product from the MIP collaboration is called JC3IEDM, which is an acronym for Joint Command Control & Consultation Information Exchange Data Model.

In order for a C2-system to use information from different domains in a seamless and automated fashion, systems producing the information must be compliant with the JC3IEDM. It is, however, not desirable to enforce use of the JC3IEDM for the internal information representation of the systems. Diverse domains will still be in need of domain-specific data standards and information models. To cope with this, it is required that a translation is made between local information representation formats and the JC3IEDM.

Traditionally, integration of systems has mainly targeted the syntactic level and focused less on the semantics of information. Given a dynamic and complex environment, where information from a range of heterogeneous domains is exchanged, it is also crucial to consider the inherent meaning of information. An ontology is a formal and explicit specification of a conceptualization [Gruber 1993], and can be used to represent and exchange knowledge about a domain in an unambiguous way. Through integration of ontologies, originating from different domains, semantic interoperability can be achieved, and thus, the likelihood of correct interpretation of information by various parties of a large and complex organization increases.

1.1 Purpose & scope

The purpose of this study is to explore how heterogeneous ontologies may be integrated to provide the basis for semantic interoperability. The overall aim is to enable systems from diverse domains to provide information to the C2 arena through integration of domain ontologies and the JC3IEDM. This report will address integration of ontologies from a general perspective to gain better understanding of how integration in the context of the JC3IEMD can be achieved. Further, it will provide insight into some high level processes of concern when managing a set of integrated ontologies. Finally, hands-on experiences from integration efforts involving C2IEDM[2] will be described.

Figure 1 provides a conceptual view of the desired state resulting from the integration process. A domain ontology expresses the semantics of a domain-specific information system or database. This ontology is integrated with the JC3IEDM and a translator provides means of changing the representation format of instances (data), thus enabling the C2 enterprise to consume information from a specific domain and vice versa.

[2] C2IEDM is the predecessor of JC3IEDM

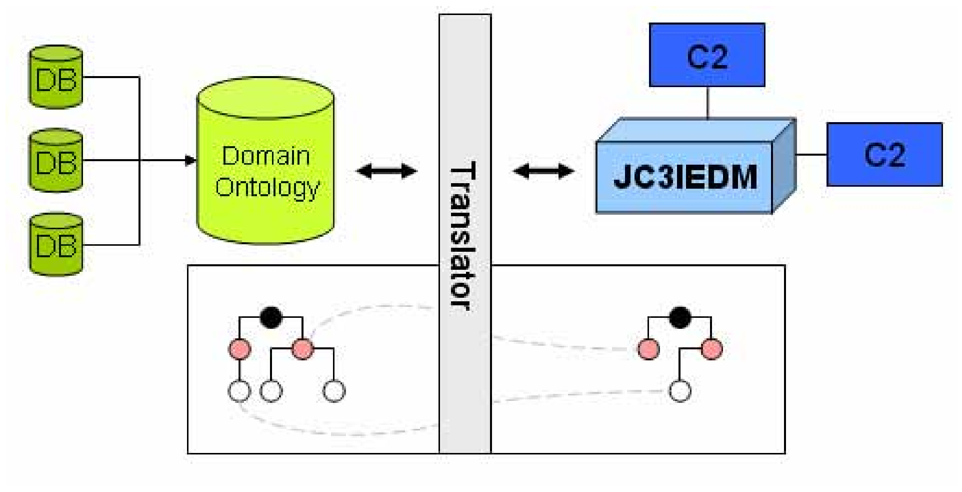


Figure 1. Conceptual view of integration of domain ontologies in the C2 enterprise.

1.2 What is an ontology?

Ontology is defined in dictionaries, e.g. in Merriam-Webster Online Dictionary [MWOD], as

- a branch of metaphysics concerned with the nature and relations of being
- a particular theory about the nature of being or the kinds of existents

In computer science several areas refer to the term *ontology*, but the interpretation varies. Within the area of Artificial Intelligence ontology is referred to as a specific perspective, or an assumption, concerning the target application area to be represented. In Information Systems Modeling, an ontology is a meta-model that facilitates common understanding of a target system, early in the systems development process [Eklöf et al. 2004]. In Information Science, an ontology is seen as the result of the effort to specify an exhaustive and rigorous conceptual schema about a domain. Thus, an ontology is an abstraction of the reality, or part of the reality. Typically, an ontology comprises a hierarchical structure of relevant concepts and their relationships, but also rules that are applicable for the modeled domain.

In recent years the *Semantic Web* has attracted a lot of attention from both the industry and research communities. This can be explained by what the Semantic Web claims to achieve:

> *"an extension of the current Web, in which information is given well-defined meaning, better enabling computers and people to work in cooperation."*. [Berners-Lee 2001]

A cornerstone of the Semantic Web is ontologies, which will promote shared understanding, reuse of knowledge, precise searches for information etc. Given this, a number of ontology languages have been developed to support the above mentioned vision. Among the most widely used languages are RDF (Resource Description Framework) [RDF] and OWL (Web Ontology Language) [OWL], which have reached recommendation status in the process of becoming a W3C (World Wide Web Consortium) [W3C] standard. RDF is a simple data model used to refer to objects (resources) and their inter-relations. RDF Schema is a vocabulary used for specifying properties and classes of resources, including semantics for generalization hierarchies. OWL is based on RDF and RDF Schema but provides some additional constructs, thus, OWL is a more expressive language. E.g. in OWL it is possible to describe disjoint classes, cardinality of properties, equality and much more.

Ontologies exist at different levels of abstraction. Usually three main categories of ontologies are discussed in this context; upper level, mid level and domain ontology. The purpose of an upper ontology is to represent concepts that are basic, universal (generic) and of common sense. Concepts defined in the upper ontology should not be specific for a particular domain, but be of use in several domains (ideally an arbitrary domain). The common concepts specified by the upper ontology provide a knowledge base upon which more specialized ontologies, mid and domain ontologies, can be built. In this way, knowledge and semantics already specified in the upper ontology is reused and the process of ontology integration is facilitated. A domain ontology specifies concepts that are specific to a particular domain, whereas the mid ontology serves as a bridge between universal concepts expressed in the upper ontology and domain-specific concepts in a domain ontology [Eklöf et al. 2005]. Examples of ongoing initiatives to develop upper ontologies are SUMO[3] [SUMO] and DOLCE[4] [DOLCE].

Ontologies can also be classified according to their expressiveness. [McGuiness 2003] provides the following levels of expressiveness that can be used to classify ontologies:

– *Controlled vocabulary*: a list of terms
– *Thesaurus*: relations between terms are also provided, e.g. synonyms
– *Informal taxonomy*: an explicit hierarchy is defined (generalization and specialization), but there is no strict inheritance
– *Formal taxonomy*: explicit hierarchy with strict inheritance
– *Frames (classes)*: classes comprise sets of properties, which are inherited by specializations
– *Value restrictions*: values of properties are restricted
– *General logic constraints*: using logical or mathematical formulas, values of properties may be constrained by values of other properties
– *First-order logic constraints*: first-order logic constraints are allowed and relations may be more detailed, e.g. disjoint classes or inverse relationships

Based on the levels presented above, Figure 2 provides a classification of a number of well-known ontologies based on their expressiveness. Note the distinction made in this figure between light-weight and heavy-weight ontologies.

[3] SUMO – Suggested Upper Merged Ontology
[4] DOLCE – a Descriptive Ontology for Linguistic and Cognitive Engineering

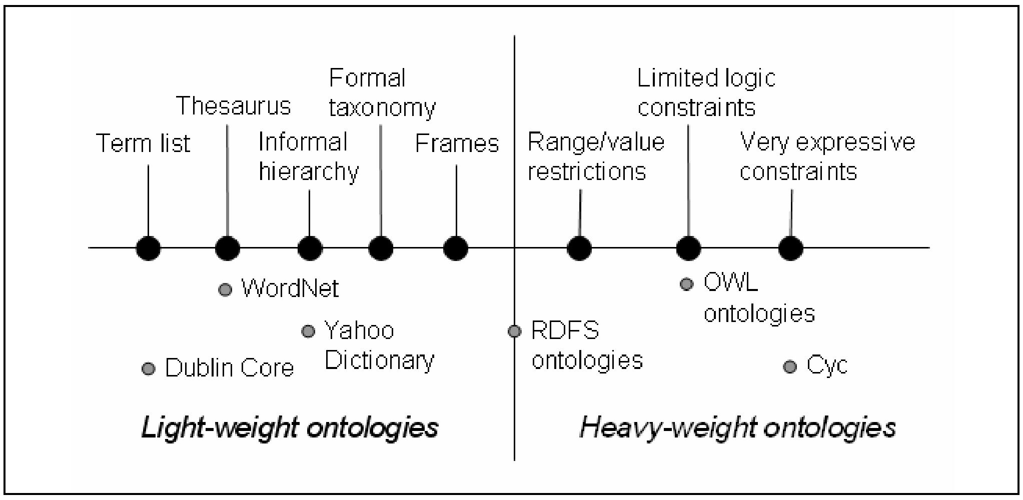


Figure 2. Classification of common ontologies according to their expressiveness [reproduced from Alexiev et al. 2005].

1.3 Ontology management

The exploitation of ontologies in information technology systems has proved successful. An increasing number of applications rely on formal ontologies and the global interest in the development of the Semantic Web promises even more to come. As a consequence, the research focus has shifted from the development of tools for ontology creation to techniques for handling multiple ontologies. These techniques constitute what is known as *ontology management*.

The purpose of ontology management is twofold, to facilitate ontology *reuse* and ontology *interoperability*. Ontology reuse is necessary to cut development costs. Typically a lot of effort is spent on developing ontologies from scratch. If instead parts from existing ontologies can be included, significant savings can be made. Ontology interoperability, on the other hand, is a key feature for expanding the utilization of knowledge. The goal is to enable automatic transfer of information between multiple applications, which are possibly relying on different conceptualizations of the domain of interest.

An ontology management system should comprise numerous functions in order to support users when creating, modifying, querying and integrating ontologies. The main components of ontology management can be divided into three categories: ontology integration, ontology evolution and versioning, and storage and retrieval [Ding et al. 2001].

Ontology integration deals with the many aspects of combining multiple ontologies. The role of the ontology management framework is to support the integration process in, e.g.,

- aligning different domain descriptions
- translating ontologies
- building ontologies from components

– checking the result of integration
– performing inference and queries across multiple ontologies

Ontology evolution and versioning handles the different situations that arise when ontologies are changed. The ontology management system should provide mechanisms to maintain the various versions and highlight their differences.

Storage and retrieval are intrinsic parts of an ontology management environment. Ontology libraries should allow uniform access to ontologies and provide pertinent information about each one of them, such as their authors, domain, and documentation. They should also support the import and reuse of ontologies, and let users extend and customize ontologies developed by others [Noy & Musen 2004].

In a large-scale industrial setting additional requirements on the ontology management solution may come into play. Scalability, availability, reliability and performance are important issues usually not considered in academic prototypes, but potentially of great interest in a commercial context [Das et al. 2001]. Likewise, security management with the possibility to support different levels of access for different types of users can be necessary to protect the integrity of data. On top of this, in some cases tools for distributed multi-user collaboration would be useful, for instance when ontologists, domain experts and end users need to work efficiently together from different geographic locations.

1.4 Outline of report

The focus of this report is on exploring the possibilities of achieving ontological interoperability, not to study techniques for ontological reuse. The obvious key enabler for the *creation* of interoperability is ontology integration. However, in order to *maintain* interoperability over time, ontology evolution and versioning must be considered as ontologies change. Hence, the emphasis of this report is on ontology integration; evolution and versioning is described more briefly, and storage and retrieval falls outside the scope and is not further discussed.

Chapter 2 provides an overview of the area of ontology integration. First, a basic set of definitions are provided that pin-point what ontology integration means. Second, the fundamental problems that occur when integrating ontologies are explained briefly. Also, the difference between ontology and database schema integration is explained. Finally, a number of common architectures for ontology integration are described.

Chapter 3 provides an overview of how to manage ontology evolution and versioning. This overview includes a basic set of definitions, a description of the ontology evolution process, and finally, a description of how to manage different versions of an ontology.

Chapter 4 describes methods and tools that are used to support ontology integration. This includes methods for discovery of similarities between a set of ontologies and how to represent and use discovered similarities, i.e. mappings between ontologies. Further, this chapter describes a number of tools of use in the ontology integration process.

Chapter 5 dives into a selection of academic and industrial projects working on semantic interoperability issues. Two projects are of generic interest and two are specifically concerned with C2IEDM. Focus is on presenting experiences and lessons learned from integration efforts.

Chapter 6 discusses ontology management from a high level perspective.

Chapter 7 proposes future work, primarily aimed at obtaining practical experience from ontology integration in the context of the JC3IEDM.

2. Ontology Integration

Historically, system integration has often targeted the syntactic level. However, the fundamental meanings, or interpretations, of concepts have often been neglected. This inherently leads to ambiguities, which prevent systems from being interoperable at the semantic level. For example, consider two systems, interoperable at the syntactic level, exchanging information concerning robots. The first system may interpret the concept of "robot" as a system for automated assembly of components within the industry, whereas the second system interprets the concept as a type of weapons system. Clearly, the integration of these systems must include changes to the fundamental data models employed.

In general, ontologies are considered applicable for resolving these kinds of conflicts, by representing the semantics of information in an explicit way. If ontologies are used in the systems development process, to form the basic agreement between involved parties, and to represent information and knowledge of the target domain, the potential for true semantic interoperability will increase. However, this assumes a global ontology, accepted by all parties and applied in all system development projects. Experience shows that this is not a viable option [Uschold & Gruninger, 2004]. Systems and users from diverse domains will proclaim diverse requirements on the information contained within such a universal ontology, and thus, the modeling and maintenance of the ontology will become complex and unmanageable. Consequently, there is a strong need for preservation of the semantics and structure contained within different ontologies, when integrating systems from diverse domains. An ontology encodes an individual's or corporate perspective of the world, which inherently causes conflicts when integration of different specifications is required. Even though ontologies are developed for a common domain by different individuals or communities, the end result will differ due to differences in the perception of the world [Maedche et al. 2003]. Thus, methods and techniques for coupling of disparate ontologies are sought, i.e. methods for ontology integration.

2.1 Terminology

Today, there is no consensus regarding the terminology used within the field of ontology integration. [Noy & Musen 1999] defines ontology alignment as "*… establishing different kinds of mappings (or links) between two ontologies, hence preserving the original ontologies.*" Further, they define ontology merging as: "*.. generate a unique ontology from the original ontologies*". [Klein 2001] defines merging and integrating equally, namely "*Creating a new ontology from two or more existing ontologies with overlapping parts, which can be either virtual or physical.*" [Pinto 1999] defines ontology integration as "*Building an ontology, by assembling, extending, specializing and adapting, other ontologies which are parts of the resulting ontology.*" Further, the author defines ontology merging as "*Building an ontology, by merging different ontologies on the same or similar subject into a single one that unifies all of them.*"

Regardless of the terminology used, two cases can be distinguished:

1. Combination where ontologies cover disparate subject areas
2. Combination where ontologies cover a common subject area

In addition to this, a second distinction can be made based on the definitions provided above, namely:

1. The combination can be *physical*, i.e. a new ontology is created in the new application domain and the old ones are discarded
2. The combination can be *virtual*, i.e. mappings are created, which are used in the new application domain to translate between source ontologies

In the following report we will use the term ontology integration, which in this context refers to combination of ontologies, covering similar abstractions. We also make a distinction between physical and virtual integration. Considering the context of this report, i.e. integration of heterogeneous system in the C2 arena (through use of the JC3IEDM), virtual integration is the most interesting approach to consider. Note that ontology integration refers to the complete process of combining two or more ontologies and is made up of several activities.

2.2 Problem

To better understand how to perform ontology integration, mechanisms causing conflicts between ontologies should be examined. The process of ontology integration is obstructed by several factors. [Klein 2001] defines two main levels of mismatches, namely language level and ontology level mismatches. Language level mismatches are caused by heterogeneous ways of specifying an ontology, e.g., constructs used to define classes in different ontology languages. Ontology level mismatches refer to differences in how a specific domain is modeled. Below, a brief description of these types of mismatches is given, based primarily on [Klein 2001].

The main aspects of language level mismatches are syntax and expressivity. The syntax employed by different ontology languages usually differs. For example, in RDFS[5] the following primitive is used to define a class of cars: *<rdfs:Class ID=”Car”>*, whereas in OIL[6] the following construct is used for the same purpose: *class-def Car*. In general, these types of mismatches are easy to resolve. A more complicated issue is mismatches in expressivity of ontology languages. For instance, in RDF Schema it is forbidden to define cyclic inheritance relations, a feature that is allowed in OIL.

Ontology level mismatches occur when ontologies cover a common subject area. In these cases problems arise even though the ontologies are represented in a common language. [Klein 2001] envisions three main categories of ontology level mismatches: conceptualization, explication and terminological mismatches. Conceptualization mismatches refer to problems related to the fact that different modelers will interpret a

[5] RDFS – Resource Description Framework Schema [RDF]
[6] OIL – Ontology Inference Layer [OIL]

domain differently. For example, conceptualization of a domain by different modelers will produce different sets of classes and relations between those. Also, given the application area of an ontology, the conceptualization of a domain will result in a particular granularity, or abstraction level. Explication mismatches refer to problems caused by the style of modeling chosen by the modeler, for instance, how concepts such as time, action and plans are represented, or which modeling conventions are employed. Terminological mismatches refer to cases when concepts are represented by different names (synonyms), or when the meaning of a concept differs depending of context (homonyms). Further, words from different natural languages (e.g. Swedish, English, French) might be used to name concepts, or syntactic variations of the same word might be used to name a concept [KnowledgeWeb]

Also, terminological mismatches address the occurrence of different encodings, e.g., representation of a date data type, or unit of measure for distances. Besides ontology mismatches, versioning of ontologies is important to address. As source ontologies are evolved over time, a versioning function is required to manage the influence on mappings between ontologies. Figure 3 provides a classification of problems related to ontology integration.

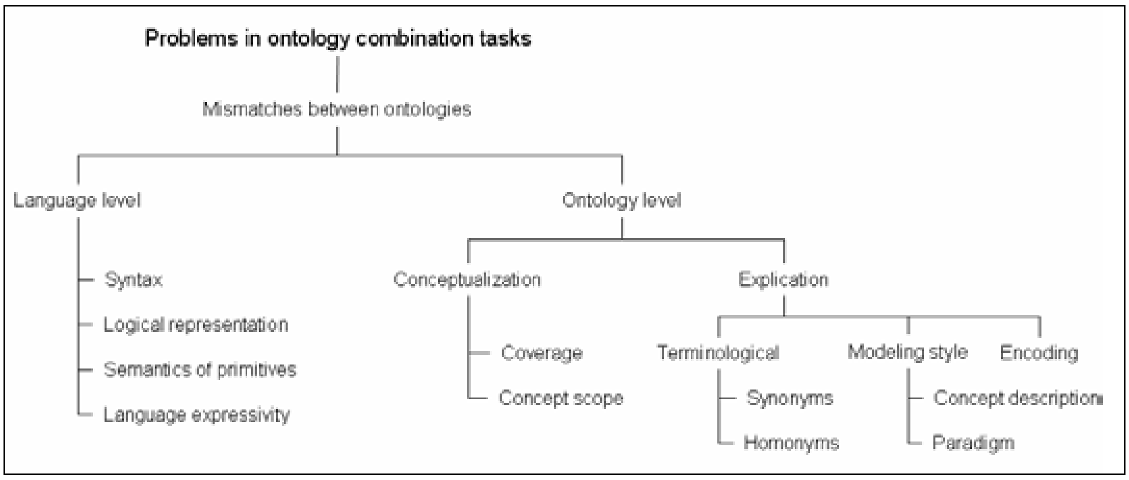


Figure 3. Classification of problems related to integration of ontologies [reproduced from Klein, 2001].

2.3 Schema vs. Ontology integration

The process of integrating ontologies shares several features with the process of schema integration, which has been studied extensively within the database community. Since the mid 80’s the integration of heterogeneous databases has been a topic of research and today, solutions for schema integration are frequently used in various domains, e.g. in the GIS domain [Fonseca 2001]. Database technology was introduced during the 1960’s and has become the backbone in many information systems. The most widely adopted structure of databases is the relational database. Other types of structures have also been employed, for instance object-oriented databases [Alexiev et al. 2005]. The schema integration process aims at building a global view of information based on a set of independently developed schemas, to enable queries over a set of heterogeneous databases [CROSI]. Even though schema and ontology integration appear to originate from the same problem, and the processes for accomplishing the integration is similar in both cases, there are fundamental differences. However, as stated in [Shvaiko & Euzenat 2005], techniques developed for each problem domain can be of mutual benefit.

Above all, there is a clear separation between schema and ontology integration in terms of how the semantics of information is handled. Typically, schema integration is carried out at the syntactic level, whereas ontology integration also considers the inherent meanings of concepts (the semantics) [CROSI]. Schemas and ontologies do share several features when it comes to expressiveness, for example the ability to declare objects, properties, aggregation, generalization and constraints. Entities in an ER (Entity-Relationship) model are equivalent to classes in an ontology, and attributes and relations are similar to relations or properties of an ontology. An ontology usually comprises a set of constraints, which are declared to express meaning, and ultimately enable automated reasoning. Constraints are most often not enforced to a similar extent in schemas. This hinders the semantic integration of schemas since the semantics is implicit. It is desirable to capture as much meaning as possible to facilitate schema or ontology integration [Uschold & Gruninger 2004]. The semantics of a database schema is often specified at design-time, but most often, it does not become a part of the database specification [Shvaiko & Euzenat 2005]. Being semantically richer, ontologies are usually closer to the modeler's view of the world, compared to database schemas.

Also, there is a key difference in the intended purpose of developing a schema or an ontology. A schema describes the contents of a database and is usually internal to an information system. An ontology is developed to describe a particular domain and is most often external to an information system [Fonseca 2001]. Ontologies are applied in various settings, for instance to support interoperability, search, or to provide a software specification. Schemas are mainly concerned with structuring of a database to support queries. The primary role of an inference engine in the context of an ontology is to enable derivation of new information. Optionally, the inference engine can be employed to ensure the integrity of instances and to check consistency of the ontology itself. Database engines are often optimized for query answering and consistency checking of data (instances). Also, in the ontology domain, consistency checking can be performed with or without instances and there is no sharp boundary between the ontology and the instances. [Uschold & Gruninger 2004].

In order to identify similarities in a set of schemas, lexical and structural features of the specifications are used. Approaches for ontology integration usually go beyond this level. For instance, it is common to analyze semantic relationships in the source ontologies (e.g. subclass-of or part-of relationships and properties of classes). The larger number of constraints usually expressed in ontologies, compared with schemas, are used to discover similarities [Noy 2004]. As stated in [Shvaiko & Euzenat 2005] the integration of database schemas usually employs techniques that in practice are based on guessing the semantics of the specifications. Ontology matching utilizes the semantics that is explicitly encoded in the specifications. Thus, the richer semantics found in ontologies cater for efficient, and possibly semi-automated, approaches to similarity discovery.

The difference in how schemas and ontologies treat semantics can be illustrated by comparing XML and RDF Schema. In general, XML is geared towards describing the structure of documents and focuses less on how the contained information is interpreted. This is an efficient solution in settings where the content of documents is known beforehand, but less efficient in dynamic environments where systems are integrated more sporadically and the contents of documents is not known in advance. Thus, XML Schema is useful for describing hierarchical structures, but does not provide extensive semantics for elements, or relations among elements. The semantics is stated implicitly and thus, the interpretation can vary extensively between different individuals or machines. In RDF, the semantics of information is declared in an explicit way, facilitating a common interpretation among a group of individuals or machines. Consider the following XML structure:

```
<person>
     <name>Martin<name>
</person>
<iso8601date>
     <W3Cprofiledate>2003-09-30</W3Cprofiledate>
</iso8601date>
```

A human is capable of grasping that <name> is related to <person> in the form of a ”hasName” relation, whereas <W3Cprofiledate> is a specific type of <iso8601date>. However, the semantics is implicit and hard for a machine to deduce. In RDF <person> is explicitly linked to <name> by a “hasName” predicate, and <W3Cprofiledata> is declared to subsume <iso8601date>. Here, the explicitly defined semantics caters for common interpretation of concepts.

More recent work also shows the benefit of using ontologies in the process of integrating database schemas. See for example [Cruz et al. 2004 or An et al. 2005], where the integration of XML Schemas, which inherently do not represent the semantics of information, is improved by incorporation of ontologies, thus bringing the pure syntactic matching of XML Schemas to the semantic level. Ontologies can associate semantics with data in databases, which will facilitate the process of integration. Thus, a number of database problems can be solved more easily by use of ontologies, e.g. federated databases, data warehousing and information integration.

2.4 General Process

This section describes common activities in the process of integrating ontologies. The description provided here assumes that a suite of tools are applied to facilitate the process. It is possible to perform ontology integration “by-hand”, but this is a time-consuming and costly endeavor. However, it is important to highlight the importance of human intervention in the process, preferably by subject matter experts that are familiar with the domains under consideration.

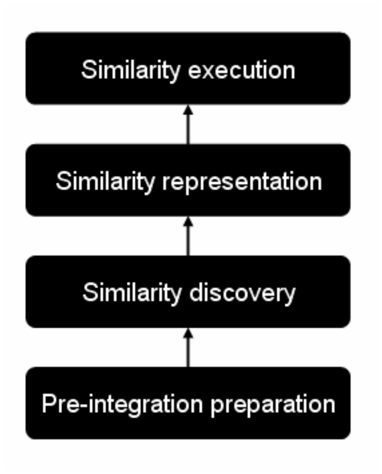


Figure 4. Activities of the ontology integration process according to [CROSI].

In Figure 4 activities of the ontology integration process, according to [CROSI], are outlined. In the *pre-integration* step the syntax of the source ontologies is unified, thus removing conflicts caused by syntactic heterogeneity. Also, the scope of the integration effort is determined and external sources are gathered, e.g. lexicons. In order to enable automation in the next phase, *similarity discovery*, source ontologies need a common representation language. The *pre-integration* phase may be automated to some extent by using ontology language conversion tools, for instance converting a DAML-OIL file to its corresponding representation in OWL. The uniformly represented source ontologies will then undergo similarity discovery in order to determine the correspondence. This phase of the process is aided by semi-automated tools, which will guide, preferably, a domain expert in defining the correspondences. Some tools, e.g. iPROMPT [PROMT] and Chimaera [Chimaera] create a new, merged ontology based on the source ontologies, whereas others, e.g. MOMIS [MOMIS] and ONION [Mitra et al. 2000], define mappings between ontologies (in section 4.3 these tools are described further). In the next phase, *similarity representation*, a formal way of representing identified correspondences, mappings, is chosen. The chosen approach will influence the potential for automation in the next phase, *similarity execution*. *Similarity execution* may produce a concrete global ontology (merge of source ontologies), a virtual global view of source ontologies, a set of articulation rules, and/or a query rewriting formula [CROSI].

[Maedche et al. 2003] provides an alternate view of the same process that also covers ontology evolution, see Figure 5. In this approach there are two separate dimensions, a vertical and a horizontal. The features of the horizontal dimension influence all aspects of the vertical one. The dimensions are elaborated further below.

The phases of the vertical dimension are more or less identical to the phases defined above, i.e.:

1. *Lift & normalization*: this phase refers to the activity of bringing source ontologies to a common representation format.

2. *Similarity*: this phase defines methods for identifying similarities between source ontologies.
3. *Semantic Bridging*: this phase covers establishment of the actual bridges between classes and properties of the source ontologies, based on the result from the similarity identification.
4. *Execution*: this phase will transform instances of a source ontology into the equivalent representation of another source ontology.
5. *Post-processing*: this phase takes the results from the execution phase in order to check and improve the quality of the transformation results.

However, the approach also considers more long term aspects such as ontology evolution and means of improving the quality of the integration process. The following aspects are managed in the horizontal dimension:

1. *Evolution*: this aspect covers means of managing the logical axioms defined in the semantic bridging phase to reflect changes made to the source ontologies.
2. *Cooperative Consensus Building*: this aspect treats the process of obtaining a common agreement concerning the bridging axioms between domains involved in the integration process.
3. *Domain Knowledge & Constraints*: this aspect refers to the possibility to provide background knowledge into the process of similarity evaluation, e.g. use of lexical ontologies to find similar concepts.
4. *Graphical User Interface*: mapping of ontologies is a complex task that requires input from humans, possessing a deep understanding of the target domains. Thus, extensive support for visualization is required to build a robust and effective integration process.

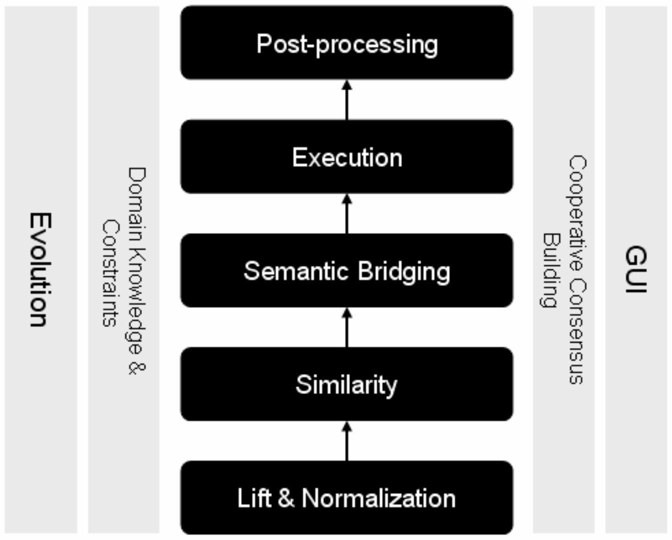


Figure 5. Activities of the ontology integration process according to [Maedche et al. 2003].

It is important to acknowledge that integration of ontologies is an iterative process that requires infrastructure support, standardized procedures and guidelines. Derived mappings will require evolution over time as domain-specific ontologies are changed or replaced.

2.5 Architectures

In almost every ontology integration approach, a "bridging ontology" is used to explicitly define the semantics of source ontologies. However, the application of the bridging ontology differs. Generally, three basic architectures are found [Wache et al. 2001] (see Figure 6):

1. Single ontology approach
2. Multiple ontologies approach
3. Hybrid approach

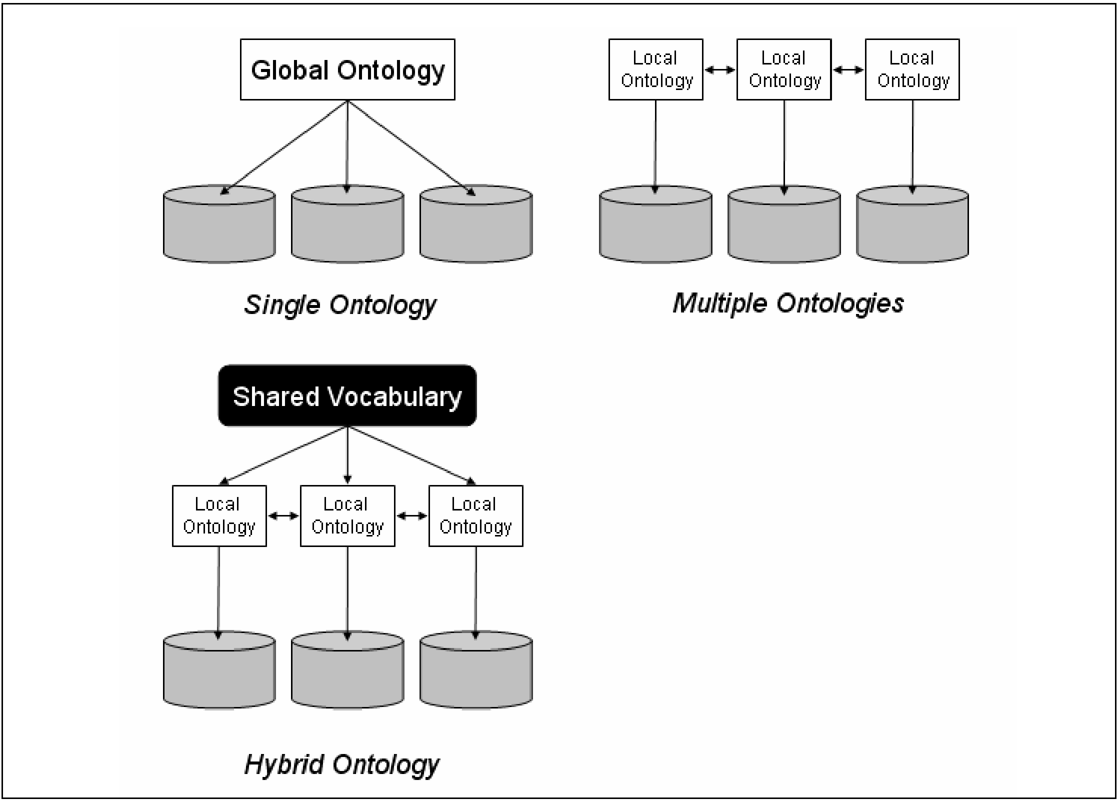


Figure 6. Common architectures in ontology integration approaches [reproduced from Wache et al. 2001].

In the single ontology approach, the semantics of information sources are expressed in a common shared ontology. The main area of application for this approach is when the abstractions of information sources are similar, i.e. the target domains are modeled in more or less the same way. This is not ideal considering the fact that development of ontologies is often influenced by the developer's view of things, background, knowledge etc. Moreover, the single ontology approach is also hampered by changes in the conceptualizations of the information sources, which will occur at some point in time. Changes at the local level must in these cases be reflected at the global level as well [Wache et al. 2001].

In the multiple ontology approach each information source has its own local ontology. The local ontologies are inter-linked using formalized mappings, defining corresponding concepts of the source ontologies. A set of mappings must be provided for each link established between information sources. The approach may seem trivial, but mapping between ontologies, having varying levels of abstraction and different views of the domains, is complex. The mappings must also be maintained over time to reflect changes in the conceptualization of information sources [Wache et al. 2001].

The hybrid ontology approach aims at overcoming the problems of the aforementioned approaches. In this approach, the local ontologies are developed using a global shared vocabulary. The shared vocabulary may constitute a basic definition of terms from the target domain, or a more detailed ontology. Since the local ontologies share a common vocabulary they are easily comparable, meaning that the mappings between local ontologies are more easily extracted. The hybrid ontology approach overcomes the main drawbacks of the single and multiple ontology approaches, but the model is based on the assumption that ontologies will be developed from scratch when a new information source is integrated. Thus, the approach does not cope with reuse of existing ontologies [Wache et al. 2001].

Another interesting approach is ontology clustering. Instead of sharing a common, single ontology, multiple and smaller shared ontologies may be applied. In ontology clustering, specialized ontologies are organized in clusters, each of which employs a shared ontology. The clusters are organized in hierarchical fashion with the most general cluster in the top of the hierarchy. Lower level clusters extend concepts defined in higher level clusters in order to obtain the desirable level of detail [Visser & Tamma 1999]. The use of clusters for ontology integration has been employed in the ONION approach, described briefly in section 2.5.

3. Ontology evolution & versioning

Creating an ontology is most often a very complex task. The development process involves end-users as well as ontology and domain experts, and there are many aspects to consider in making the model both complete and consistent. When the work is finally done and the ontology is put into use, this is unfortunately not likely to be the end of the efforts. The ontology usually still contains minor (or major) errors, and it will probably not cover all situations that are waiting in future applications. Over time various updates of the ontology will be needed. When different applications rely on the same ontology, updates can lead to systems malfunctioning and data corruption. Additional complications arise when other ontologies depend on the ontology that is being updated, leading to inconsistencies that can be hard to trace. Therefore, a methodology for dealing with ontology change and multiple versions of ontologies is needed.

3.1 Terminology

In literature, ontology change management comes in two flavors, as *ontology evolution* and *ontology versioning*. The two notions are inspired by their counterparts in database schema management. Schema evolution is the ability of updating a schema without losing the possibility to access data conforming to the old schema. Schema versioning is the ability to access both old and new data, but through different versions of the schema interface. The adaptation of these concepts to the world of ontologies is not straightforward [Noy & Klein 2003]. For example, ontologies do not make the distinction between ontologies and instances as clear as databases do between schemas and instances. The ontology itself is often used as data, making evolution in the sense of database schema impossible. Ontologies are also more often reused, possibly by a large number of applications and other ontologies. To make all dependent artifacts conform simultaneously to a change is therefore seldom viable. Hence, evolving an ontology means also maintaining a new version, and the two notions become indistinguishable.

In this report we will treat ontology evolution as a subprocess of ontology versioning, and use the following definitions:

Ontology evolution is ”*the process of modifying an ontology in response to a certain change in the domain or its conceptualization*” [Flouris & Plexousakis 2005].

Ontology versioning is “*the ability to handle changes in ontologies by creating and managing different variants of it*” [Noy & Klein 2003].

3.2 Ontology evolution process

In [Stojanovic et al. 2002] ontology evolution is described as a process in six phases (Figure 7). The process is cyclic, since the last phase – the validation of the introduced changes – can reveal the need for additional changes due to introduced inconsistencies.

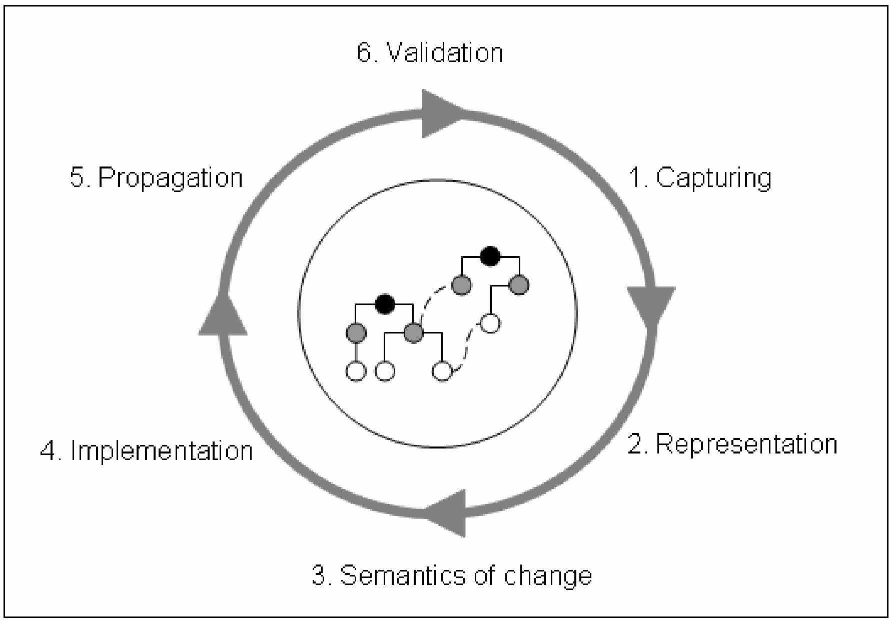


Figure 7. Ontology evolution [reproduced from Stojanovic et al. 2002].

Change Capturing

The initial step is to capture the changes that need to be implemented. The causes of changes can be divided into three categories [Noy & Klein 2003]:

1. *Changes in the domain*. Sometimes the actual domain that the ontology are trying to capture change and the ontology has to be altered correspondingly. An example could be two military units that are merged into one, or the introduction of a new type of aircraft.
2. *Change in conceptualization*. A change of perspective of the world might influence the way the conceptualization should be made. For example, when driving in urban terrain, houses may be considered as simple obstacles. When taking the perspective of a soldier on foot, the houses could instead be viewed as something that offers sheltered passage.
3. *Change in specification*. If the translation of an ontology is made into a representation form with different expressive capabilities, the ontology might have to be changed in order to preserve its current semantics.

The change capturing can either be done in a top-down fashion, where the changes are identified by e.g. an engineer as requirements for developing a new application, or in a bottom-up fashion through *change discovery*. Change discovery is performed by analyzing the behavior of the system. If changes in the system data or in the system usage patterns are discovered, ontology refinement can automatically be inferred and suggested to the ontology engineer.

Change Representation

The changes identified in the capturing step must somehow be expressed in an unambiguous way in order to be correctly implemented. Some changes can be quite complicated, using a number of simpler changes as building blocks. It is common to use a dedicated taxonomy or ontology for the representation of the changes [Haase et al. 2004].

Semantics of Change
This phase deals with the effects of the changes to the ontology itself. The task is to enable careful and systematic resolution of the changes to ensure consistency. As an example, consider the concept of a "Leopard" with the property "carries ammunition". It is clear from the context that we are dealing with a Leopard-tank. However, changing the ontology by deleting this property would make the concept of "Leopard" equally well refer to the animal.

Change Implementation
The objective of change implementation is to perform and keep track of the requested changes. Before changes take place though, the implementer should be presented a list of all consequences in order to prevent unwanted effects.

Change Propagation
When the modifications are implemented the effects are propagated to applications, application data and dependent ontologies. The change propagation phase is meant to recognize these effects and deal with them appropriately. Applications may need to be updated and in order to handle new inconsistencies in dependent ontologies, new evolution processes may have to be initiated.

Change Validation
Even though the preceding phases are carried out with care, there can be numerous situations where the changes need to be reversed. For instance, all implications of the changes might not have been fully understood or were just implemented for experimental reasons. By creating detailed evolution logs the ontology engineer should be able to completely re-create the initial ontology.

3.3 Ontology versioning

Instead of propagating implemented changes of an ontology to all dependent systems, the ontology can be split into different versions. This allows voluntary conformation to the changes and a much more flexible architecture. On the other hand, if the different versions are not compatible, interoperability may be lost. To support interoperability between multiple versions, an ontology versioning methodology is needed.

Ontology versioning includes mechanisms for identification of the different versions, for describing their relations and degree of compatibility. A desired feature to achieve is *transparent access,* to automatically relate versions with data sources, applications and other depended elements [Flouris & Plexousakis 2005].

In the ideal case the creation of new versions of an ontology is always accompanied by detailed change specifications. The compatibility of two different versions can then be traced as a sequence of changes. However, in a highly distributed development environment, as the case of the Semantic Web, this traced information cannot be taken for granted. In such a case tools are needed for automatic or semi-automatic version comparison [Noy & Klein 2003].

4 Methods & Tools

Today, no ontology management framework exists that supports all the necessary steps of the processes described in the previous chapters. Methods developed are most often devoted to various subtasks, and are produced by people or groups working independently and with different goals. Summing the efforts, it is clear that the field of ontology integration is the most mature. However, many aspects of versioning overlap with integration. For instance, establishing mappings between different versions is an integration task. Similarly, techniques for capturing differences between versions resemble the approaches of similarity discovery, although the objectives are opposite.

Most researchers agree that a fully automated process for ontology integration is not plausible for the time being [Noy 2004]. In most situations a human is needed to assert that relationships between mapped concepts are true. It is desirable to involve domain experts in the process that have specific knowledge of the domain/domains under consideration. Today, the mapping of two or more ontologies is usually an interactive process where the domain expert has the final decision on how to define the mappings [Dou et al. 2004]. However, there are methods and techniques that can be employed to simplify the process significantly.

4.1 Similarity discovery

This section describes classes of techniques for discovery of mappings in the ontology integration process. Moreover, two examples are provided to highlight some basic features of these techniques. Note that the purpose is not to provide a complete description of available methods, but rather highlight some general characteristics of techniques for ontology mapping. The reader is referred to [CROSI] or [KnowledgeWeb] for an in-depth description of techniques.

4.1.1 Classification of techniques

The mapping of two or more source ontologies is a semi-automated process requiring input from a user. Even though the process is not completely automated, using available tools and techniques for this task it to prefer over manual mapping. Tools for semi-automatic ontology mapping usually benefit from the following features of ontology specifications [Noy 2004]:

- Concept names and natural-language descriptions
- Class hierarchies
- Property definitions (domains, ranges and restrictions)
- Instances of classes
- Class descriptions

In Figure 8 a classification of schema-based matching approaches is provided. Even though this classification refers to approaches for schemata matching, corresponding approaches can be found for ontology mapping. [Shvaiko & Euzenat 2005] makes a distinction between heuristic and formal techniques and between implicit and explicit approaches. Heuristic techniques make qualified guesses of relations among similar labels or graph structures, whereas formal techniques employ model theoretic semantics to validate assertions made. Implicit techniques are based on syntactic features, whereas explicit techniques consider the underlying semantics of terms.

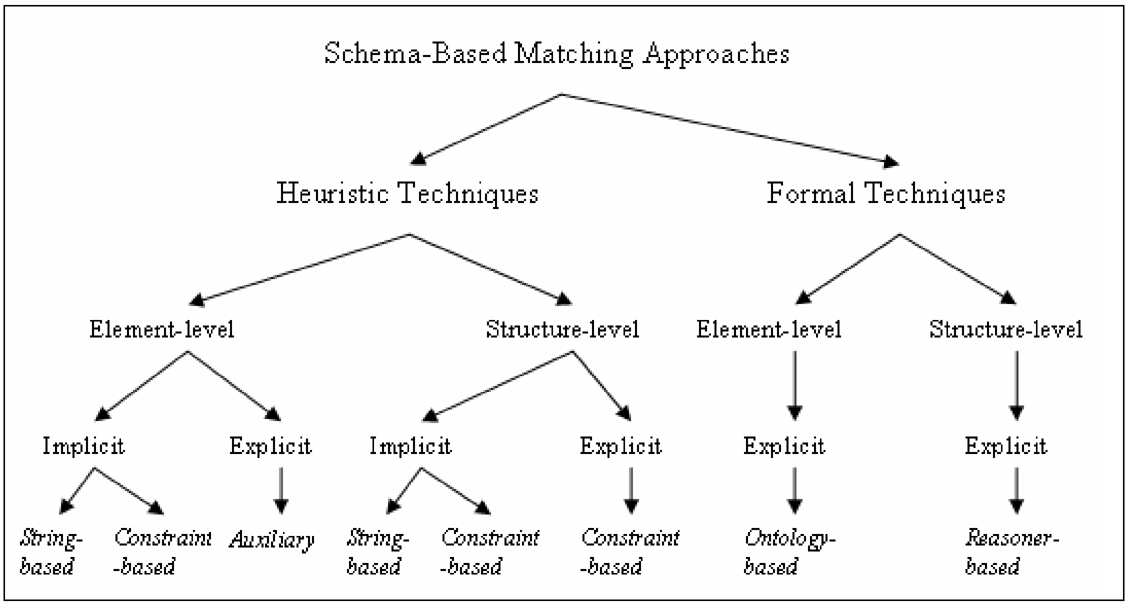


Figure 8. Classification of schema matching approaches [reproduced from CROSI].

Another important distinction is element-level versus structure level approaches. This is referred to as local and global approaches in [KnowledgeWeb]. Local methods are fairly simple, operating on a limited scale of the ontology structure (element level), whereas global methods consider the entire structure of ontologies when discovering similarities (structure level). For instance, a global method can be applied to aggregate the result from basic local methods to yield the similarity estimate. In [KnowledgeWeb] a comprehensive description of local and global methods is given. Also, [CROSI] makes an in-depth analysis of available methods in this area.

Element-level approaches

String-based techniques are basic ways of comparing concept names, originating from disparate ontologies. The approach assumes that semantically similar concepts have similar syntactic features. More advanced forms of string similarity evaluation also consider textual descriptions associated with concepts and properties. This can be augmented with analysis of pronunciation, or soundex (a phonetic algorithm for indexing of names based on their pronunciation) to gain better performance [CROSI].

Language-based techniques consider concept names as words in natural language. The techniques are based on Natural Language Processing (NLP), which utilizes morphological properties of words, e.g. tokenization of words (OWL-based representation will become [OWL, based, representation]). Usually, language-based techniques are applied prior to string-based analysis [Shvaiko & Euzenat 2005].

More complex approaches for deriving the mappings do not utilize name similarity, but rather focus on the internal structure of concepts in the source ontologies. Constraint-based techniques facilitate similarity discovery by analyzing datatypes, cardinality etc. of a concept's attributes [Shvaiko & Euzenat 2005]. Also, in this type of analysis, different contexts can be applied that differentiate distinct types of relations among concepts, i.e. particular relations are included in the analysis while the importance of others is suppressed [CROSI].

In addition to these techniques, resources external to source ontologies can be used. The matching procedure may utilize synonyms and hypernyms as defined in generic thesauri, such as WordNet [WordNet], to provide meaning for concepts in the ontologies. Also, mappings produced in earlier mapping sessions can be reused. This is based on the idea that ontologies to be integrated share several features with already integrated ontologies. This is especially true for ontologies covering a common domain [Shvaiko & Euzenat 2005].

Structure-level approaches
An ontology forms a hierarchical structure which can be represented by a tree with labeled nodes. By analyzing this graph, such as data type properties of classes, or a concept's relations to other concepts, similarities can be discovered [CROSI]. Graph-based techniques assume that two concepts, originating from different but similar ontologies, may also have neighbors that are similar.

Taxonomy-based techniques analyze the graph structures of source ontologies, but only consider the specialization relation. For example, if the super-concepts of two concepts are similar, the concepts themselves are similar to each other. Likewise, if the sub-concepts of two concepts are similar, the concepts themselves are also similar.

The integration of ontologies can be facilitated by a repository of structures. In this case, pair-wise similarities of structures of ontologies are stored in addition to the ontologies themselves. The structure of an ontology refers to number of nodes, maximum path length etc. When ontologies are matched, the structures of these are compared with structures in the repository to find similarities. This will avoid a full mapping procedure in case the structures are dissimilar, and further, it enables reuse of existing mappings [Shvaiko & Euzenat 2005].

Other approaches rely on analysis of instances derived from source ontologies. In this way the underlying semantics of the ontologies are obtained by looking at how instances are classified. The assumption made in this case is that instances having similar semantics might share certain features [CROSI]. These techniques are usually applicable if a relatively complete set of instances is available.

4.1.2 Examples – PROMPT

In the following sections two examples are given to highlight how algorithms for discovery of mappings operate at the local scale (element level) and at the global scale (structure level). The algorithms presented here are implemented in an application named

PROMPT [PROMPT], which is a plug-in for the Protégé ontology development framework [Protégé].

Local Method - iPROMPT

The PROMPT application includes an algorithm called iPROMPT that operates on the local scale (element level). In iPROMPT, a merged ontology is created based on two source ontologies. As a start, an initial list of matches is produced. This is based purely on lexical similarity of class names

iPROMPT defines a set of operators to be applied in the process of ontology merging; these are:

- *Merge classes*. Merges two separate classes that are considered to be equivalent
- *Merge slots*. Merges properties of classes that are considered to be equivalent
- *Merge instances*. Merges equivalent instances, originating from separate source ontologies
- *Shallow copy*. Copies a class of the source ontology to the merged ontology, including properties of the class
- *Deep copy*. Copies a class of the source ontology to the merged ontology, including all parents of the class

In the subsequent section the *merge classes* operation is described in greater detail. Consider having two source ontologies as described in Figure 9. The lexical analysis has suggested merging of the *Person* and *PERSON* classes and the user has decided to follow that suggestion. In this case the following actions are performed:

- A new class, *C*, is created in the merged ontology, O_m. If the term used for the classes in the source ontologies are equal, this term is used for naming *C*, otherwise the user will choose a name. In our example *C* will be named *Person*.
- For each superclass, *S*, of *Person* and *PERSON*, already represented in O_m, make *S* parent of *C*. The same process is carried out for subclasses. Since there are no inherited classes in our example, the merged ontology, O_m, will remain the same.
- For each property, *P*, of *Person* and *PERSON* in the source ontologies, such as *affiliation* and *workplace*, that is not represented in O_m copy *P* to O_m. For each *P* in O_m, associated with *C*, attach *P* to *C*. In the example, the *affiliation*, *manages*, *workplace* and *leader_of* properties are copied to O_m and attached to *C*.
- Check the linguistic similarity of properties copied and attached to *C* in O_m. If properties are similar suggest merging of these. In the example there are no linguistic similarities of the properties. However, since they represent equal concepts, for instance *affiliation* and *workplace*, they can be merged by the user at a later time.

- Check the linguistic similarity of pairs of superclasses and subclasses to *C*. If similar terms are used, suggest merging of these classes. Since there are no inherited classes in the example ontologies, the merged ontology will remain the same.
- Check for duplicate paths from *C* to parent classes. If redundancies are identified, suggest removal of paths.

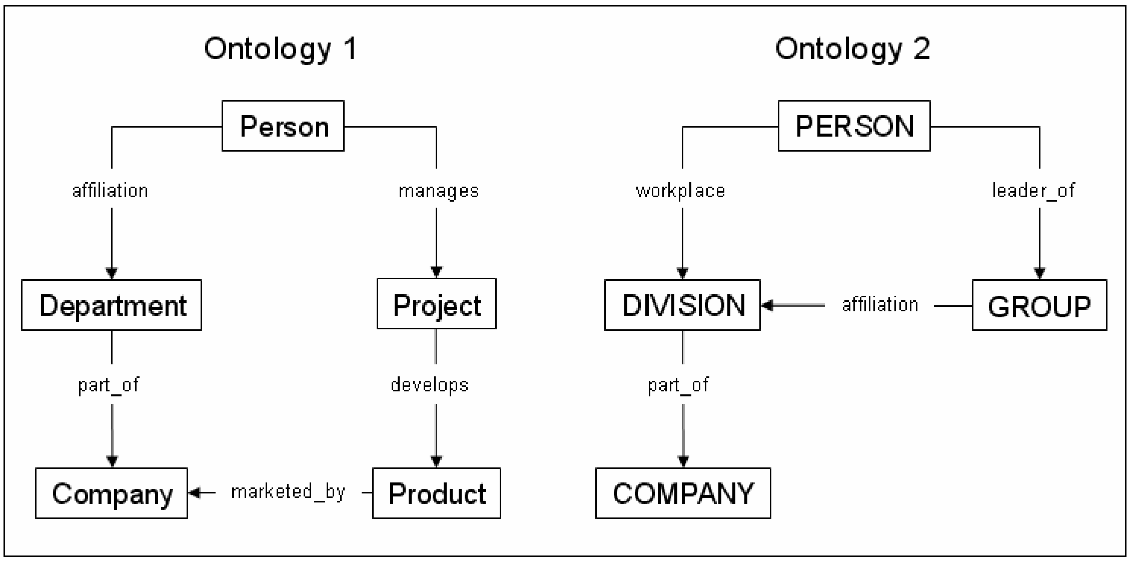


Figure 9. Example ontologies.

The merged ontology after applying iPROMPT, at the end of the first iteration, is presented in Figure 10. At this stage the *Person* concept has four different properties. However, these will be merged in subsequent iterations of the algorithm.

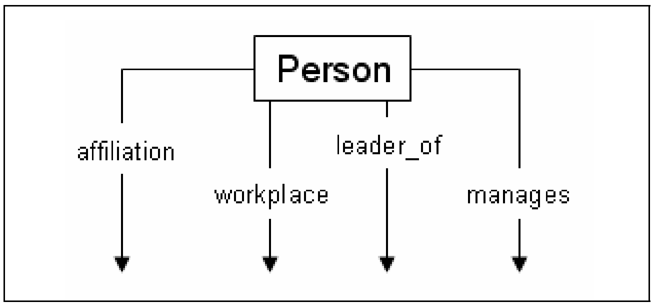


Figure 10. Result from iPROMPT after the first iteration.

Global Method - AnchorPROMPT

The AnchorPROMPT algorithm uses non-local contexts for semantic matching. AnchorPROMPT suggests a set of semantically similar concepts to be merged from two source ontologies based on a set of anchors, which are pairs of related concepts already identified. The anchors can be identified through lexical analysis as described above (the iPROMPT algorithm).

An ontology can be seen as a directed labeled graph, where each class is represented by a node, and relations between classes are represented by directed edges. In order to identify semantically similar concepts, AnchorPROMPT traverses this graph based on provided anchors. Consider two anchors, originating from two source ontologies:

- A_1 and B_2 are equivalent concepts of the sources O_1 and O_2 respectively
- C_1 and D_2 are equivalent concepts of the sources O_1 and O_2 respectively

In this case the algorithm will traverse the path from concept A_1 to concept C_1 in O_1, and the path from concept B_2 to concept D_2 in O_2, in parallel. Along this route, similar terms are identified.

Consider the case when two different ontologies, representing the structure of organizations, should be merged, see Figure 9. At this point of the process a set of anchors have been identified; these are:

- *Person* and *PERSON* represents equivalent terms
- *Company* and *COMPANY* represents equivalent terms

Based on these anchors, the algorithm should find other related terms in the source ontologies. To do this, all paths from *Person* to *Company* and from *PERSON* to *COMPANY* are generated. Given the example in Figure 9 a pair of paths is:

- *Person – Department – Company*
- *PERSON – DIVISION – COMPANY*

A pair of paths of equal length, is processed in parallel and for every pair of terms that are visited during the traversal, a similarity score is incremented. In this case the similarity score for the pair *Department – DIVISION* is incremented. The algorithm processes each pair of paths of equal length for the predefined anchors in this way, and increments the similarity score of pair of terms successively. In the end, the pair of terms that appear in the same position on all paths traversed will get the highest score. It is assumed that if pairs of similar concepts are connected to a second pair of similar concepts, it is likely that these concepts are also similar. The idea is to produce a large number of suggestions for concepts to be merged based on a minimal input, the set of anchors. In our example the algorithm will recommend merge of the Department and DIVISION classes.

4.2 Similarity representation & execution

This section describes how mappings, defined in the similarity discovery phase, can be represented and used. A high level view of some interesting approaches is provided with a more detailed study of a framework called MAFRA.

4.2.1 Use of derived mappings

At a minimum, it should be possible to execute requests such as [Uschold & Gruninger 2004]:

> "Given this message, encoded using ontology A, please return a translated message encoded using ontology B. And please use this particular mapping and that particular translation engine"

The expressive power of ontologies gives rise to the opportunity to represent mappings in a formal way. Thus, the mappings can be used by an inference engine to automatically translate between source ontologies. There are several approaches available for representing and utilizing the mappings to enable translation between ontologies. In [Dou et al. 2004] a system called OntoMerge is described, which uses an inference engine to perform translation between source ontologies. Within the OntoMerge framework, classes and properties of source ontologies are related through bridging axioms (formal logical expressions). In this context, an axiom should be seen as a translation rule, e.g. translate a class in one source ontology to the equivalent class in a second source ontology. The axioms are managed by an inference engine, which enables the coupling of source ontologies.

Other approaches utilize ontologies to declare mappings between source ontologies. [Crubezy & Musen 2004] defines a Mapping Ontology, whereas [Maedche et al. 2003] describes a Semantic Bridge Ontology. Common for these approaches is that instances based on such ontologies define actual mappings between source ontologies. A set of instances, defining a particular mapping, can be used by a tool (e.g. an inference engine) to perform the actual translation. A similar approach is described in [RDFT]. In this case a mapping meta-ontology named RDFT (RDF Transformation) is defined that is used to integrate disparate vocabularies.

Another interesting approach is that of [Euzenat, 2004]. In this work an API for ontology alignment has been implemented, mainly targeting OWL and RDFS ontologies. This API can be embedded in a Java application to provide means of managing mappings between two ontologies. Functions for extracting the alignment (mapping) through customized algorithms are provided, but also functions for outputting the alignment to various formats. The output can then be used to perform the actual translation between source ontologies. The API can also produce stylesheets that an XSLT-engine can process in order to carry out the transformation. A more interesting option is to produce rule sets in SWRL (Semantic Web Rule Language), which an inference engine can process in order to carry out the transformation. In this case the underlying semantics is managed more thoroughly, compared with the more or less syntactic transformations made by an XSLT-engine.

When considering ontology integration in the context of the JC3IEDM, an important issue to address is the type of mappings that are relevant. A highly expressive language is capable of representing complex relationships (mappings) between integrated ontologies. On the other hand great expressiveness may require vast computing capacity in the translation process, and mappings will be harder to maintain over time. Thus, it is important to resolve the requirements that the JC3IEDM will impose on similarity representation.

4.2.2 Example – Semantic Bridge Ontology

In the following section the Semantic Bridge Ontology (SBO), presented in [Maedche et al. 2003], is explained further based on a simple example. When using the MAFRA framework in order to map source and target ontologies, an instance of the SBO ontology is created that comprises instances of semantic bridges. The semantic bridges contain the required information to perform transformation of instances between source and target ontologies. In Figure 11 an example of ontology integration by means of the SBO is given.

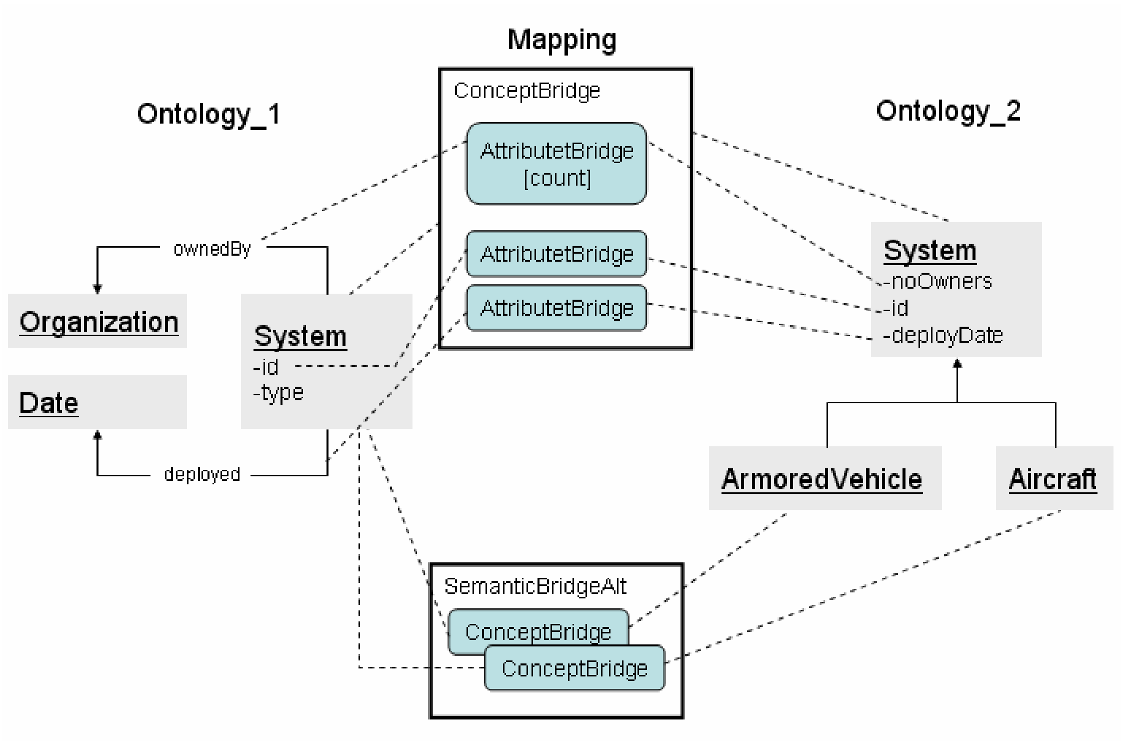


Figure 11. Example ontologies, integrated by means of the Semantic Bridge Ontology (SBO).

First, the ontologies to be integrated are specified according to the following construct:

```
<Mapping rdf:ID=”mapping”>
    <relatesSourceOntology rdf:resource=”&Ontology_1;”/>
    <relatesTargetOntology rdf:resource=”&Ontology_2;”/>
    <hasBridge rdf:resource=”#System-System”/>
</Mapping>
```

Ontology_2 in Figure 11 comprises a *System* class that is subsumed by the *ArmoredVehicle* and *Aircraft* classes. However, in this ontology the *System* class is abstract, i.e. when instances are created either an *ArmoredVehicle* or *Aircraft* is created.

In the example we are interested in transforming an instance of Ontology_1 to its corresponding representation in Ontology_2, thus Ontology_1 is the source and Ontology_2 the target. Ontology_2 defines a *System* class which could either be an *ArmoredVehicle* or an *Aircraft*, depending on the value of the *type* attribute. Thus, it is necessary to define two alternative *ConceptBridges*, namely *System - ArmoredVehicle* and *System - Aircraft*. In order to determine which *ConceptBridge* to use, a *SemanticBridgeAlt* is defined according to the following:

```
<SemanticBridgeAlt rdf:ID=”ArmoredVehicleOrAircraft”>
    <hasBridge>
          <Seq ordinal=”1”><bridge rdf:resource=”#System-
                ArmoredVehicle”/></Seq>
    </hasBridge>
    <hasBridge>
          <Seq ordinal=”2”><bridge rdf:resource=”#System-
                Aircraft”/></Seq>
    </hasBridge>
</SemanticBridgeAlt>
```

The alternative *ConceptBridges* are defined according to the following constructs:

```
<ConceptBridge rdf:ID=”System-ArmoredVehicle”>
    <subBridgeOf rdf:resource=”#System-System”/>
    <relatesSourceEntity rdf:resource=”#System”/>
    <relatesTargetEntity rdf:resource=”#ArmoredVehicle”/>
    <whenVerifiedCondition rdf:resource=”#isArmoredVehicle”/>
</ConceptBridge>

<ConceptBridge rdf:ID=”System-Aircraft”>
    <subBridgeOf rdf:resource=”#System-System”/>
    <relatesSourceEntity rdf:resource=”#System”/>
    <relatesTargetEntity rdf:resource=”#Aircraft”/>
</ConceptBridge>
```

The alternative *ConceptBridges* subsumes the *System – System ConceptBridge* and thus, they inherit its transformation of attributes. In addition to this, the *System – ArmoredVehicle* bridge defines a condition, *isArmoredVehicle*, to determine if the instance being translated is an *ArmoredVehicle*. The *ConceptBridges* are tested (executed) in the context of the *SemanticBridgeAlt*, which means that the *System – ArmoredVehicle* bridge is tested first. If the *isArmoredVehicle* condition is not true the *System – Aircraft* bridge is unconditionally executed, i.e. the *System* should be translated to an *Aircraft*. The *isArmoredVehicle* condition is defined according to the following construct:

```
<Condition rdf:ID="isArmoredVehicle">
     <putServiceArgument>
            <MapArg><from>1</from><to rdf:resource="#type"/></MapArg>
     </putServiceArgument>
     <putServiceArgument>
            <MapArg><from>pattern</from><to>AV</to></MapArg>
     </putServiceArgument>
     <inService>CascadeAndMatch</inService>
</Condition>
```

The construct states that the value of the *type* attribute should be compared with "AV", which is the code used for an armored vehicle in *Ontology_1*. In case this comparison is true, the *System* instance will be translated to an *ArmoredVehicle* instance.

The definition of how to transform class attributes is provided in the *System – System ConceptBridge*. Next, the construct for transforming the id attribute in this bridge is presented. This translation is defined by an *AttributeBridge* relating the source attribute with the target attribute. Further it defines a transformation to be used, in this case the *copyId* transformation.

```
<AttributeBridge rdf:ID="id-id">
     <relatesSourceEntity rdf:resource="#id"/>
     <relatesTargetEntity rfd:resource="#id"/>
     <accordingToTransformation rdf:resource="#copyId"/>
</AttributeBridge>
```

The CopyId transformation is defined according to the following:

```
<Transformation rdf:ID="copyId">
     <mapSourceArgument>
            <MapArg><from rdf:resource:"#id"/>
            <to>sourceString</to></MapArg>
     </mapSourceArgument>
     <mapTargetArgument>
            <MapArg><from>targetString</from>
            <to rdf:resource="#id"/> </MapArg>
     </mapTargetArgument>
     <inService>CopyString</inService>
</Transformation>
```

In this case the value of the *id* attribute of a *System* instance, derived from Ontology_1, is copied to the *id* attribute of a *System* instance (*ArmoredVehicle* or *Aircraft*), derived from Ontology_2. This is accomplished by using a built-in service of the inference engine, which is called *CopyService*. Handling the *ownedBy* attribute of the *System* class in Ontoloy_1 is more complex. This attribute refers to an instance of the *Organization* class, but its equivalent representation in Ontology_2 is an ordinary attribute, *noOwners,* of type integer. This is managed by the following *AttributeBridge* and associated *Transformation*:

```
<AttributeBridge rdf:ID=”owners”>
     <relatesSourceEntity rdf:ownedBy=”#id”/>
     <relatesTargetEntity rfd:noOwners=”#id”/>
     <accordingToTransformation rdf:resource=”#countOwners”/>
</AttributeBridge>

<Transformation rdf:ID=”countOwners”>
     <putServiceArgument>
           <MapArg><from>relation</from>
           <to rdf:resource=”#ownedBy”/></MapArg>
     </putServiceArgument>
     <mapTargetArgument>
           <MapArg><from>count</from>
           <to rdf:resource=”#noOwners”/></MapArg>
     </mapTargetArgument>
     <inService>CountRelations</inService>
</Transformation>
```

In this case an alternative built-in service of the inference engine is utilized, namely *CountRelations*. This service counts the number of relations a *System* instance has to *Organization* instances. The result is used to assign a number to the *noOwners* attribute of an instance of the *System* class, derived from Ontology_2.

4.3 Tool support

There are numerous tools available to support the process of ontology integration. Providing an in-depth description and assessment of these tools is beyond the scope of this report. However, to give some insights concerning the prospects of applying this technology in near future, a short description of some common tools within this area is given below. The reader is encouraged to read [CROSI], [KnowledgeWeb] and [Shvaiko & Euzenat 2005] for a more comprehensive description on the subject.

4.3.1 MOMIS

MOMIS is an acronym for Mediator envirOnment for Multiple Information Sources and is a framework for integration of heterogeneous information sources. Based on structured or unstructured content, MOMIS establishes a mediation schema used to provide a uniform query interface for the user. Components of the MOMIS framework are illustrated in Figure 12. The content of a source is translated to the ODL language by a wrapper. The mediator is responsible of breaking up a global query into sub-queries for connected sources. Further, it is responsible for assembly of the results provided by the sources through their wrappers [MOMIS].

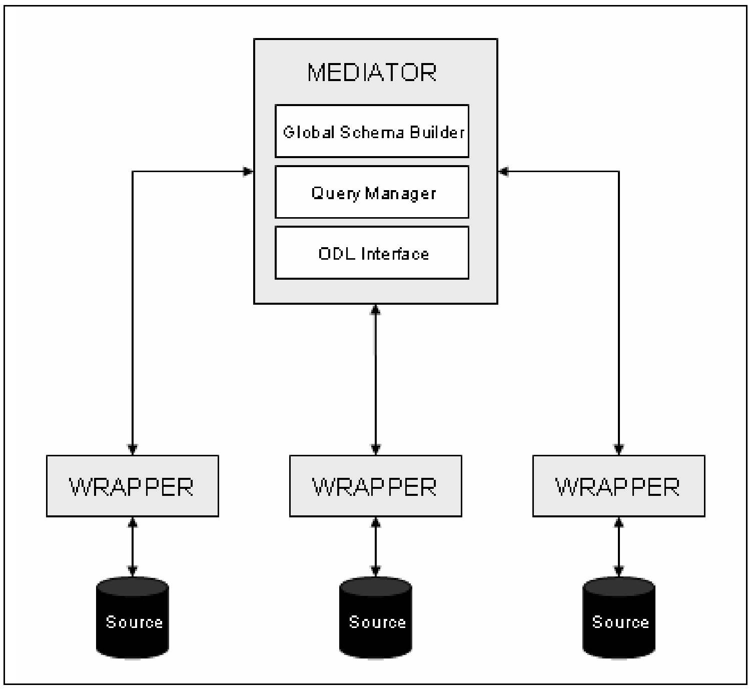
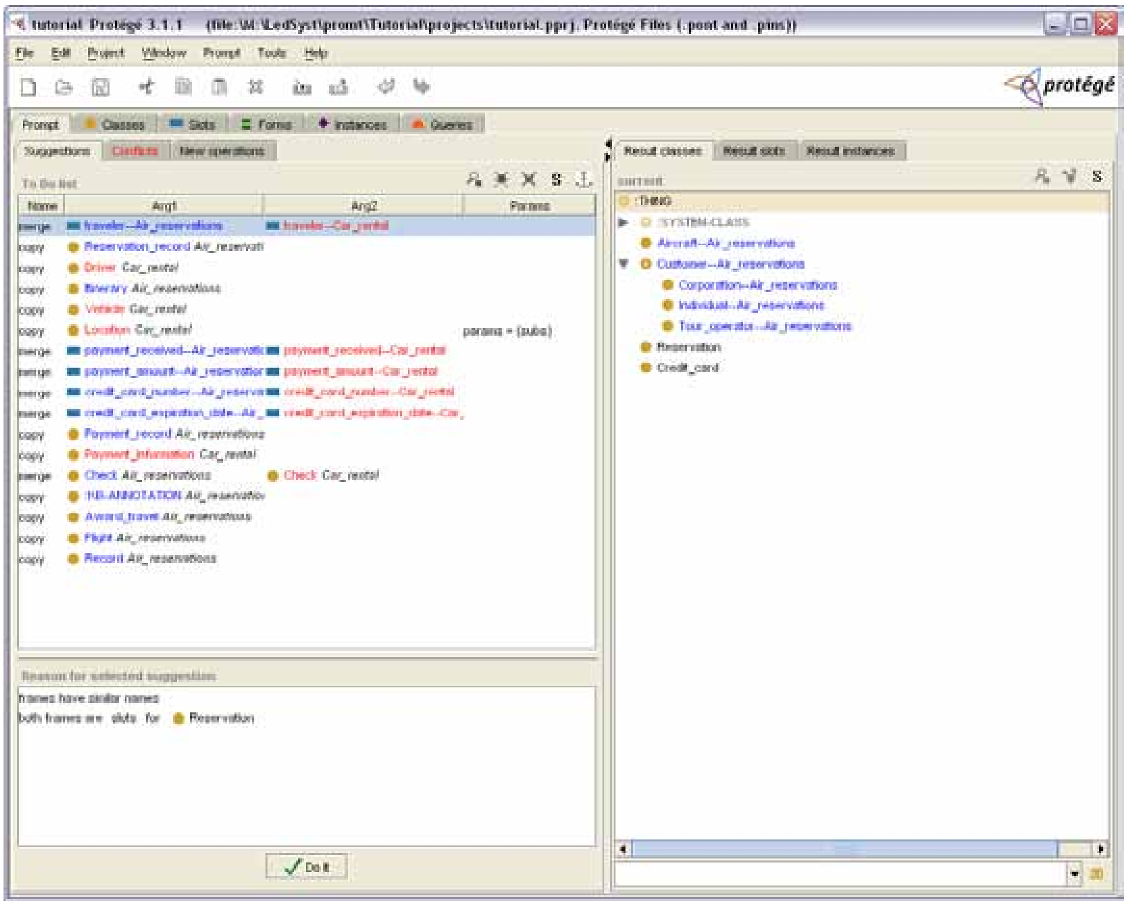


Figure 12. Architecture of the MOMIS framework.

4.3.2 PROMPT

PROMPT is a suite of tools for multi-ontology management, available as an extension to the Protégé ontology-editing environment. See [Protégé] for further information regarding this environment. The tools suite comprises four separate components, whose functionality is integrated through the Protégé environment. iPROMPT is an interactive ontology merging tool, AnchorPROMPT is a tool for discovery of similarities based on graphs, PROMPTDiff is a tool for discovery of differences between two versions of the same ontology, and PROMPTFactor is used for extraction of subparts of an ontology [PROMPT]. Figure 13 provides a snapshot of the user-interface of PROMPT.


tutorial Protégé 3.1.1 (file:\M:\LedSyst\promt\Tutorial\projects\tutorial.pprj, Protégé Files (.pont and .pins))

File Edit Project Window Prompt Tools Help
protégé
Prompt Classes Slots Forms Instances Queries
Suggestions Conflicts New operations
To Do list
Result classes Result slots Result instances
Do It

Name	Arg1	Arg2	Params
merge	traveler--Air_reservations	traveler--Car_rental	
copy	Reservation_record Air_reservati		
copy	Driver Car_rental		
copy	Itinerary Air_reservations		
copy	Vehicle Car_rental		
copy	Location Car_rental		params = (subs)
merge	payment_received--Air_reservatio	payment_received--Car_rental	
merge	payment_amount--Air_reservatior	payment_amount--Car_rental	
merge	credit_card_number--Air_reserva	credit_card_number--Car_rental	
merge	credit_card_expiration_date--Air_	credit_card_expiration_date--Car_	
copy	Payment_record Air_reservations		
copy	Payment_information Car_rental		
merge	Check Air_reservations	Check Car_rental	
copy	:KB-ANNOTATION Air_reservatio		
copy	Award_travel Air_reservations		
copy	Flight Air_reservations		
copy	Record Air_reservations		

Reason for selected suggestion

frames have similar names
both frames are slots for Reservation

:THING
- :SYSTEM-CLASS
- Aircraft--Air_reservations
- Customer--Air_reservations
  - Corporation--Air_reservations
  - Individual--Air_reservations
  - Tour_operator--Air_reservations
- Reservation
- Credit_card

Figure 13. User-interface of PROMPT, ontology merging tool.

4.3.3 Chimaera

Chimaera is a tool for ontology merging and diagnostics provided by means of a browser-based user-interface. The tool checks for similarities of class and property names in order to produce a list of candidates to be merged. Based on this list the user makes the final decision on which classes and/or properties that should be merged. Also, Chimaera provides an alternative view that outlines parts of the merged ontology that might contain conflicts, and thus requires rearrangement. The Chimaera tool is available on-line at [Chimaera].

4.3.4 ONION

In the ONION framework two types of ontologies exists. First, source ontologies, which are ontologies considered for integration. Second, articulation ontologies, which represent articulation rules that bridge the source ontologies. The articulation ontologies utilize clustering, i.e. a hierarchy of ontologies. The articulation ontology used for a specific mapping can in turn constitute a source ontology in another articulation ontology (mapping), e.g. the *ART 1-2* articulation ontology in Figure 14 is a source ontology in the mapping expressed by the *ART 1-2-3* articulation ontology. ONION employs a top-down strategy for the articulation ontologies, where the root node (the most general ontology) is created first and subsumed (or specialized) by new articulation ontologies as needed [Mitra et al. 2001].

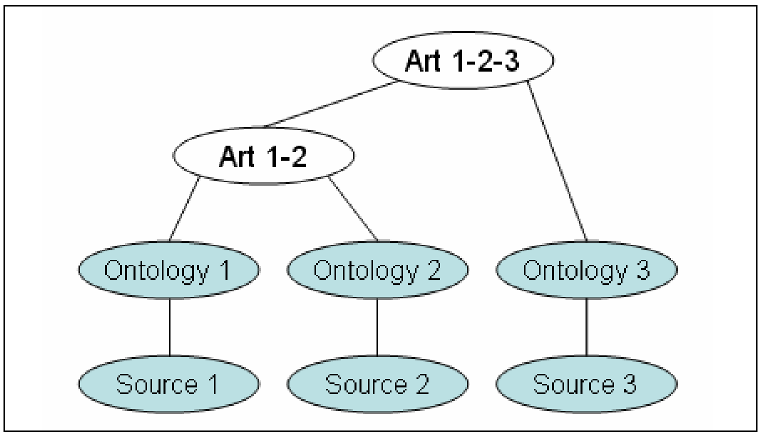


Figure 14. Articulation ontologies in the ONION framework.

4.3.5 Summary

Table 1 summarizes some properties of the ontology integration tools presented in the previous section.

Table 1. Summary of features of some common ontology integration tools [reproduced after Alexiev et al. 2005].

	MOMIS	PROMPT	Chimaera	ONION
Integration paradigm	Combination	Merging	Merging	Combination
Mapping pattern	Single shared ontology	-	-	Clustering
User model	Global	-	-	Global
Mapping support	Class; property; constraints	-	-	Class; property
Degree of automation	Semi-automatic	Semi-automatic	Semi-automatic	Semi-automatic
Interoperability	Custom wrappers	Any language supported by Protégé	Any language supported by OntoLingua	n/a
Visualization support	-	Ontology using Protégé	n/a	n/a
Evaluating ontology	Manual	Manual	Semi-automatic using diagnostic tests	n/a

Integration paradigm states what kind of end result that is achieved when applying the tool, e.g. PROMPT creates a new, merged ontology based on provided source ontologies, whereas MOMIS generates a set of mapping rules that are applied to translate between source ontologies. The *mapping pattern* property refers to the architecture that is applied to achieve ontology integration. See section 2.5 for more information on possible architectures. *User model* defines what “view” is provided when using the integration results, i.e. the user either sees information from the perspective of his own local ontology, or from the perspective of a common (global) ontology. *Mapping support* defines the mapping types that are allowed. *Degree of automation* states if the approach supports manual, semi-automatic or automatic integration. *Interoperability* refers to the type of ontology languages that each approach is capable of managing. *Visualization support* describes means of visualization that could aid the user in the integration process. Finally, *evaluating ontology* defines the method used to evaluate the result from the integration process.

Two main categories of ontology integration approaches are described in table 1. First, MOMIS and ONION create a set of mapping rules, and thus, create a virtual integration. Second, PROMPT and Chimaera creates a concrete, merged ontology based on the input. There is also a difference as far as the employed architecture in the virtual integration approaches is concerned. MOMIS uses a single shared ontology, whereas ONION employs clustering of ontologies. None of the approaches presented here represent a fully automated process for ontology integration. All approaches require input from a user, preferably having some knowledge of the domain/domains captured by the source ontologies.

Note that the tools mentioned here are more or less research prototypes. However, the PROMPT tools are implemented as plug-ins to the Protégé ontology environment, which has been used extensively within various communities. Thus, PROMPT in Protégé is possibly the most widespread and used tool for ontology integration today. Besides the tools described above the following tools/frameworks are worth mentioning: OBSERVER [OBSERVER], KRAFT [KRAFT] and GLUE [GLUE]. For a comprehensive list of information/ontology integration projects world-wide see [Integration].

5 Experiences from ontology integration

The tools described in the previous section are all more or less academic prototypes. Although both PROMPT and Chimaera are publicly available, and PROMPT is continuously updated with new versions (currently version 2.4.8), there are almost no industrial use-cases or experiences available concerning their use. Real life ontology integration projects seem so far to mainly have used manual methods or developed their own dedicated tools. Once again, turning to the more mature field of databases as a comparison, we find a rich plethora of schema matching algorithms but still little information on how to build a system that can be used in practice [Bernstein et al. 2004].

In the following paragraphs we present some of the few interesting findings we have made referring to ontology integration efforts. The first two are of generic interest, and the following two discuss integration projects involving C2IEDM.

5.1 Corporate Ontology Grid

The Corporate Ontology Grid (COG) project develops methods and technology for the deployment of an Information Grid using ontologies in an industrial setting. One of the main objectives is to "*demonstrate the technological innovation of automatically translating data between data formats on the grid by way of a semantic mapping to a central ontology*". The project is sponsored by the European Union and the results are meant to be disseminated to Europe's corporations. Many publications, such as white papers, scientific publications, and deliverables, are available on the project's homepage [COG]. Some of the initial experiences are also available as a book [Alexiev et al. 2005].

One of the partners in the COG project is the Italian car producer Fiat, who contributed to the study with a real-life use-case. The goal was to achieve semantic interoperability between the various data sources in their production system. A central ontology was created onto which the formats of the different sources were mapped. This resulted in the ability to locate and query distributed information through a single data-view, and automatically translate and transfer data between the sources spread throughout the enterprise. The information model was built using existing applications, data sources and input from domain experts.

The following is a selection of best practices and lessons learned from the COG project, outlined in [Alexiev et al. 2005]:

- *Domain vs. application modeling*. One problematic issue when constructing the shared ontology was how to balance the trade-off between domain and application interests. Defining the mapping of an application to the ontology was more easily accomplished if the ontology was modified to match the data view of the application. However, this implied worse conformance to the domain, eventually resulting in problems when integrating other applications. Similar trade-off complications did also occur when integrating data sources to the ontology.

- *Quality of global model depends on local models*. The global ontology was in part created on the basis of local data schemas (reverse-engineering). Hence, it was important that these schemas were well-documented and well-understood, which was not always the case. Much time was spent together with end-users in order to catch the proper meanings of all data.

- *Automation is hard to achieve in real-life situations.* This statement is primarily due to the fact that data sources were created in Italian and the ontology in English. This made many techniques for automatic similarity discovering impossible to use.

5.2 Cyc

Cyc, developed by Cycorp Inc., is the world's largest and most complete general knowledge base and common-sense reasoning engine [CYC]. Over the last two decades a large number of ontologies have been integrated with Cyc in order to extend the knowledge base. Some experiences of the integration work are presented in [Reed & Lenat 2002]. The authors claim that the major work is not to solve issues concerning upper ontologies:

"Much to-do has been made – by everyone from Aristotle to Sowa – about what the upper ontology (in which the most abstract and general sorts of concepts are defined) should be. But we have found empirically that most of the "action" – the minute-by-minute work of ontology mapping – is performed primarily at the middle and lower levels of the ontology"

Another lesson learned refers to Cycorp's efforts in integrating Open Directory in Cyc. The Open Directory is a huge web topic taxonomy available in RDF, and is free to update by anyone [Open Directory]. For the integration a semi-automatic tool was developed. However, as the Open Directory is continuously updated, the instance IDs change over time and mappings are lost or corrupted. The maintenance burden proved to be too heavy and the integration project was postponed until a fully automatic method can be developed.

The integration of WordNet in Cyc proved to be more successful. WordNet is an English lexical knowledge base with almost 150,000 words and phrases [WordNet]. The problem of continuous updates is shared with Open Directory, but WordNet handles this in a more systematic manner as backward compatibility is tracked by a versioning program.

5.3 Integrating C2IEDM and XBML

The US Army has developed an unambiguous language for their command and control systems, called the Battle Management Language (BML). For interoperability purposes a process has been initiated to map BML to C2IEDM. The process is described in [Turnitsa et al. 2004].

The team designing the process recognized two different approaches for the integration, top-down and bottom-up mapping. The top-down approach starts the mapping by considering *associated concepts*, larger semantic entities in which data is given in a broader context. In the next step tables are mapped, and finally possible missing properties are identified. The bottom-up approach reverses this process and starts with the mapping of simple properties. The structuring of these into tables and associated concepts is dealt with in a later step when they are needed for the applications.

The advantage of the top-down approach is that the mapping of larger entities of related concepts divides the problem into sub-classes, which reduces the overall complexity. The drawback is that there is a larger risk of mismappings between complex entities than between simple properties. In the case of integrating BML and C2IEDM the team chose to go top-down, as the structural similarities between the two ontologies were judged to be sufficient.

A lesson learned from the integration work so far (2004) is the additional complications due to the fact that the BML is only given as an instantiated relational database without an explicit data model. This leads to differences in the abstraction level of the models, which has to be handled with special care. Another experience is not to underestimate the value of learning the models well. As the C2IEDM is of considerable size, this can take some time. The authors report that "*even an experienced data modeler needs four to six weeks to understand the data model and an additional two to four weeks of practice with the model to understand the practical applicability*."

5.4 Integrating C2IEDM and OTH-T GOLD

Over-The-Horizon targeting GOLD (OTH-T GOLD) is a messaging format for the exchange of target information, supported by e.g. the US Global Command and Control System. The mapping between C2IEDM and the OTH-T GOLD ontologies has simultaneously been the focus of two independent research teams. Their shared experiences are summarized in [Dorion et al. 2005].

The authors' main observation is that differences in expressivity make life hard for ontology engineers. If one model misses the complete semantics to cover a certain concept, semantic loss is inevitable when pushing information back and forth. It is emphasized that "*the ontologist has to make assumptions on the acceptable level of semantic loss*". On the other hand, if a model is extensive enough to capture all detailed aspects of the domain, the complexity itself can be a problem and the risk for ontologists making mistakes is increased.

Another pitfall pointed out is that even though a mapping seems self-evident in one context, it may fail in covering another. The authors state that to handle this, "*higher level and more abstract semantic concepts must be known in order to succeed in mapping ontologies together*".

The final experience is a warning not to let the tools be part of the ontology. An example of this is a proposition in MIP of adding a NODE entity to C2IEDM in order to simplify database-to-database replication. Apart from being conceptually unappealing, this can lead to problems when using other tools.

6 Discussion

The mapping of heterogeneous information representation formats is a crucial capability in enabling interoperability of defense-related systems. Ontologies are the key ingredient in creating this capability by providing an explicit and common specification of knowledge contained within a domain. Having this specification, the integration with other domains is simplified since the inherent meaning of information is common (shared) and specified in a formal way.

As stated several times before, the field of ontology management as a whole is immature and tool support is still fairly weak. Available tools are targeting specific problems areas, such as similarity identification and execution, but do not usually cover the perspectives of higher level management. This does not mean that one can simply overlook these issues when pursuing an ontology integration task. Choices made when designing mappings may influence how easy the ontology will be to change in the future and how easily the changes may be documented. The importance of performing traceable changes depends on the context of the ontology in question. If it has several dependent ontologies and changes are likely to be frequent, it could be wise to adopt a versioning methodology and a standardized model for denoting changes. Given the increased interest in recent years in ontologies, and the Semantic Web in particular, increased tool support for ontology management will probably be a reality in the near future.

The need for semantic interoperability solutions most often originates from the need of two or more applications to exchange data. The prevailing solution for this during the nineties was to state a standard data model for everyone to adhere to. However, this simple and theoretically appealing idea proved hard to realize. [Rosenthal et al. 2004] discusses some experienced successes and failures of this integration philosophy in a military context. The lessons learned are that very large enterprises cannot hope to construct a single data model, and that agreement of large data models is much more likely when participants are few. The authors are devoted supporters of the new data strategy of the US Department of Defense, which emphasizes that data model standards should not be peremptory top-down directives, but developed and agreed upon in certain *Communities of Interests* (COI). The term COI is used to describe a collaborative group of data owners and producers who must exchange information in pursuit of their shared interests, and who therefore must have a shared vocabulary for the information they exchange. Determining the participants and scope of such a COI is a delicate matter but of vital importance for the success of any integration effort. A recipe for the creation of COIs in a NATO environment can be found in [Lasschuyt 2003].

The decision of the Swedish Armed Forces to participate in MIP means joining a large COI and compulsory adherence to the MIP solution, including the use of JC3IEDM for C2 information exchange. This does not mean that the local C2 data model must be based on JC3IEDM. Neither must information exchange between national C2 systems and other types of systems[7], forming other potential COIs, use the MIP solution. The suitability of JC3IEDM as the one and only information exchange model within C2 is strongly questioned in [Lasschuyt et al. 2004]. The authors claim that in its current form, the model is unbalanced in its levels of detail and too large to be practical. On the other hand, JC3IEDM also has many advantages. It is inherently extendible and has a strong support in the community. In [Turnitsa & Tolk 2005] C2IEDM is evaluated as an interoperability-enabling ontology between the communities of C2 and Modeling and Simulation. The conclusion is that even if there is room for improvements, the model supports almost all basic needs for such a semantic bridge.

[7] For an overview of the different information domains relevant for data exchange with Swedish C2 systems, see [Mårtenson & Svensson 2005].

7 Future work

This study shows the broadness of the area of ontology management. Numerous research projects have produced a range of solutions and tools in order to accomplish semantic interoperability through integration of ontologies. Also, the database community has put a lot of efforts into enabling database schema integration during the last decades, an area that has much in common with integration of ontologies. Given this, the inevitable next step would be to gain more specific hands-on experiences from integration of the JC3IEDM with one or more domain-specific ontologies (databases). This work is motivated by the following factors: there are no extensive experiences from integration efforts involving the JC3IEDM reported in the literature and it is hard to evaluate the applicability of proposed solutions for ontology integration without practical experiences. We propose the following activities for the next phase of the project:

1. Identify one or more suitable ontologies to be integrated with the JC3IEDM. If no suitable ontologies exist, these have to be developed (based on some existing system)
2. Apply existing tools for the *similarity identification* phase on selected ontologies and the JC3IEDM. In this step a mapping will be established
3. Use tools and existing solutions for *similarity execution* in order to accomplish automated translation. In this step a formal way of representing the mapping will be chosen, and an appropriate translation implementation will be selected
4. Demonstrate translation of instances from source (domain) to target (JC3IEDM) ontology
5. Evaluate the applicability of applied tools and solutions:
    - The benefits of using semi-automated tools for similarity identification
    - The level of expressivity required for formal representation of the mapping, i.e. the most beneficial approach given trade-off between expressiveness and computability
    - Effectiveness of applied translation approach

8 References

[Alexiev et al. 2005]	Alexiev V., M. Breu, J. de Bruijn, D. Fensel, R. Lara, and H. Lausen, *Information Integration with Ontologies*, John Wiley & Sons Ltd, Chichester, England, 2005.
[An et al. 2005]	An Y., A. Borgida, and J- Mylopoulos, *Constructing Complex Semantic Mappings Between XML Data and Ontologies*, Proceedings of ISWC2005, 2005.
[Berners-Lee 2001]	Berners-Lee T., J. Hendler, and O. Lassila, *The Semantic Web*, Scientific American, http://www.scientificamerican.com, accessed 2006-01-13.
[Bernstein et al. 2004]	Bernstein P. A., S. Melnik, M. Petropoulos, and C. Quix, *Industrial-Strength Schema Matching*, SIGMOD record 33(4), 2004
[Chimaera]	Chimaera homepage, http://www.ksl.stanford.edu/software/chimaera/, accessed 2006-01-13.
[COG]	The Corporate Ontology Grid project, http://www.cogproject.org, accessed 2006-01-13.
[CROSI]	CROSI – Capturing Representing and Operationalising Semantic Integration (semantic integration technologies survey), http://eprints.ecs.soton.ac.uk/10842/01/crosi-survey.pdf, accessed 2006-01-13.
[Crubezy & Musen 2004]	Crubezy M., and M. Musen, *Ontologies in Support of Problem Solving*, In *Handbook of Ontologies*, Springer-Verlag, Berlin, Germany, 2004.
[Cruz et al. 2004]	Cruz I., H. Xiao, and F. Hsu, *An Ontology-Based Framework for XML Semantic Integration*, Proceedings of IDEAS2004, 2004.
[Cyc]	Homepage of Cycorp Inc., http://www.cyc.com/, accessed 2006-01-13.
[Das et al. 2001]	Das A., W. Wu, and D. L. McGuinness, *Industrial Strength Ontology Management*, Proceedings of SWWS, 2001.
[Ding et al. 2001]	Ding Y., D. Fensel, M. Klein, and B. Omelayenko, *Ontology management: survey, requirements and direction*, On-To-Knowledge project, 2001, http://www.ontoknowledge.org/.
[DOLCE]	a Descriptive Ontology for Linguistic and Cognitive Engineering, http://www.loa-cnr.it/DOLCE.html, accessed 2006-01-13.
[Dorion et al. 2005]	Dorion E., C Matheus, M Kokar, *Towards a Formal Ontology for Military Coalitions Operation*, Proceedings of 10th ICCRTS, 2005.

[Dou et al. 2004]	Dou D., D. McDermott, and P. Qi, *Ontology Translation on the Semantic Web*, LNCS Journal of Data Semantics II, 35-57, 2004.
[Eklöf et al. 2004]	Eklöf M., P. Hörling, P. Svan, R. Suzic, and C-H, Yi, *Information & Knowledge Management for NBI (Network-Based Intelligence)*, FOI-R—1417—SE, 2004.
[Eklöf et al. 2005]	Eklöf M., R. Suzic, and C-H. Yi, *Network-Based Intelligence (NBI) – Knowledge Base Development and Use*, FOI-R—1757—SE, 2005.
[Euzenat, 2004]	Euzenat J., *An API for ontology alignment*, International Semantic Web Conference, 2004.
[Flouris & Plexousakis 2005]	Flouris G., and D. Plexousakis, *Handling Ontology Change: Survey and Proposal for a Future Research Direction,* TR-362, FORTH-ICS, 2005.
[Fonseca 2001]	Fonseca T., *Ontology-driven Geographic Information Systems,* PhD Thesis, http://www.spatial.maine.edu/~fred/fonseca_2001.pdf, accessed 2006-01-13.
[GLUE]	Machine learning to find semantic mappings in web documents, http://www-faculty.cs.uiuc.edu/%7eanhai/research.html, accessed 2006-01-18.
[Gruber 1993]	Gruber T. R., *A translation approach to portable ontology specification,* Knowledge Acquisition 5(2), 199-220, 1993.
[Haase et al. 2004]	Haase P., Y. Sure and D. Vrandecic, *D3.1.1 Ontology Management and Evolution – Survey, Methods and Prototypes*, Deliverable in the SEKT project, 2004, http://www.sekt-project.com/.
[Integration]	Data Integration Projects World-Wide, http://www.ifi.unizh.ch/stff/pziegler/IntegrationProjects.html, accessed 2006-01-18.
[Klein 2001]	Klein M., *Combining and relating ontologies: an analysis of problems and solutions*, Proceedings of IJCAI-01, 2001.
[KnowledgeWeb]	KnowledgeWeb, State of the art on ontology alignment, http://knowledgeweb.semanticweb.org/, accessed 2006-01-13.
[KRAFT]	Knowledge Reuse and Fusion/Transformation, http://www.csd.abdn.ac.uk/%7eapreece/Research/KRAFT.html, accessed 2006-01-18.
[Lasschuyt 2003]	Lasschuyt E, *Information Interoperability Domains,* RTO-SCI-137/9 available on http://www.rta.nato.int
[Lasschuyt et al. 2004]	Lasschuyt E., M. van Henken, W. Treurniet, and M. Visser, *How to Make an Effective Information Exchange Data Model,* RTO-IST-042/9

[Maedche et al. 2003]	Maedche A., B. Motik, N. Silva, and R. Volz, *MAFRA – A MApping FRAmework for Distributed Ontologies*, Proceedings of the EKAW 2002.
[McGuiness 2003]	McGuiness L., *Ontologies come of age*, In *Spinning the Semantic Web*, MIT Press, Cambridge, MA, 2003.
[MIP]	Multilateral Interoperability Programme, http://www.mip-site.org/, accessed 2006-01-13.
[Mitra et al. 2000]	Mitra P., G. Wiederhold, and M. Kersten, *A Graph-Oriented Model for Articulation of Ontology Interdependencies*, Proceedings of Conference on Extending Database Technology, 2000.
[Mitra et al. 2001]	Mitra P., G. Wiederhold, and S. Decker, *A Scalable Framework for the Interoperation of Information Sources*, Proceedings of SWWS'01, 2001.
[MOMIS]	Mediator envirOnment for Multiple Information Sources, http://dbgroup.unimo.it/Momis/, accessed 2006-01-13.
[MWOD]	Merriam-Webster Online Dictionary, http://www.m-w.com/, accessed 2006-01-13.
[Mårtenson & Svensson 2005]	Mårtenson C., and P. Svensson, "*Kartläggning av informationsdomäner för LedsystT i perspektivet FMLS 2010*", FOI Memo 1201, 2005
[Noy, 2004]	Noy N., *Semantic Integration: a survey of ontology-based approaches*, SIGMOD record, 33(4), 65-70, 2004.
[Noy & Klein 2003]	Noy N., and M. Klein, *Ontology evolution: Not the same as schema evolution*, Knowledge and Information Systems, 5, 2003.
[Noy & Musen 1999]	Noy N., and M. Musen, *An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support*, Proceedings of AAAI-99, 1999.
[Noy & Musen 2004]	Noy N., and M. Musen, *Ontology versioning in an ontology management framework*, IEEE Intelligent Systems, 2004.
[OBSERVER]	Ontology Based System Enhanced with Relationships for Vocabulary hEterogeneity Resolution, http://siul02.si.ehu.es/OBSERVER/, accessed 2006-01-18.
[OIL]	Ontology Inference Layer, http://www.ontoknowledge.org/oil/, accessed 2006-01-13.
[Open Directory]	Open directory project, http://dmoz.org/, accessed 2006-01-23
[OWL]	Web Ontology Language, http://www.w3.org/2004/OWL/, accessed 2006-01-13.

[Pinto 1999]	Pinto H., *Some Issues on Ontology Integration*, Proceedings of IJCAI-99, 1999.
[PROMPT]	the PROMPT tab for Protégé, http://protege.stanford.edu/plugins/prompt/prompt.html, accessed 2006-01-13.
[Protégé]	Protégé ontology environment, http://protege.stanford.edu/, accessed 2006-01-13.
[RDF]	Resource Description Framework, http://www.w3.org/RDF/, accessed 2006-01-13.
[RDFT]	Resource Description Framework Transform, http://zoe.mathematik.uni-osnabrueck.de/QAT/Transform/RDFTransform/, accessed 2006-01-13.
[Reed & Lenat 2002]	Reed S. L., and D. B. Lenat, *Mapping Ontologies into Cyc*, Proceedings of AAAI, 2002.
[Rosenthal et al. 2004]	Rosenthal A., L. Seligman, and S. Renner, *From semantic integration to semantics management: case studies and a way forward*, SIGMOD record 33(4), 44-50, 2004
[Shvaiko & Euzenat 2005]	Shvaiko P., and J. Euzenat, *A Survey of Schema-Based Matching Approaches*, Journal of Data Semantics IV, 146-171, 2005.
[Stojanovic et al. 2002]	Stojanovic L., A. Maedche, B. Motik, and N. Stojanovic, *User-Driven Ontology Evolution Management,* Proceedings of EKAW, 2002.
[SUMO]	Suggested Upper Merged Ontology, http://ontology.teknowledge.com/, accessed 2006-01-13.
[Tolk 2005]	Tolk, A., *A Layered Web Services Architecture to Adapt Legacy Systems to the Command & Control Information Exchange Data Model (C2IEDM)*, Proceedings of European Simulation Interoperability Workshop, 2005.
[Turnitsa & Tolk 2005]	Turnitsa C., and A. Tolk, *Evaluation of the C2IEDM as an Interoperability-Enabling Ontology,* Proceedings of SIW-05, 2005.
[Turnitsa et al. 2004]	Turnitsa C., S. Kovurri, A. Tolk, L. DeMasi, V. Dobbs and W. P. Sudnikovich, *Lessons Learned from C2IEDM Mappings within* XBML, Proceedings of Fall Simulation Interoperability Workshop, 2004.
[Uschold & Gruninger 2004]	Uschold, M., and M. Gruninger, *Ontologies and semantics for seamless connectivity*, SIGMOD record 33(4), 58-64, 2004.
[Visser & Tamma 1999]	Visser, P., and V. Tamma, *An Experience with Agent-based Ontology Clustering*, IJCAI-99, 1999.

[W3C] World Wide Web Consortium, http://www.w3.org/, accessed 2006-01-13.

[Wache et al. 2001] Wache H., T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hubner, *Ontology-Based Integration of Information – A Survey of Existing Approaches*, Proceedings of IJCAI-01, 2001.

[WordNet] Online lexical reference system, http://wordnet.princeton.edu/, accessed 2006-01-16.