Developing a Compositional Ontology Alignment Framework for unifying Business and Engineering Domains

Said Rabah Azzam

November 2012

This thesis is submitted in partial fulfilment of the requirement for the award of the degree of Doctor of Philosophy at the University of Portsmouth.

Abstract

In the context of the Semantic Web, ontologies refer to the consensual and formal description of shared concepts in a domain. Ontologies are said to be a way to aid communication between humans and machines and also between machines for agent communication. The importance of ontologies for providing a shared understanding of common domains, and as a means for data exchange at the syntactic and semantic level has increased considerably in the last years. Therefore, ontology management becomes a significant task to make distributed and heterogeneous knowledge bases available to the end users. Ontology alignment is the process where ontology from different domains can be matched and processed further together, hence sharing a common understanding of the structure of information among different people.

This research starts from a Comprehensive review of the current development of ontology, the concepts of ontology alignments and relevant approaches. The first motivation of this work is trying to summarise the common features of ontology alignment and identify underdevelopment areas of ontology alignment.

It then works on how complex businesses *can be designed and managed by Semantic modelling which can help define the data and the relationships between these entities, which provides the ability to abstract different kinds of data and provides an understanding of how the data elements relate.*

The main contributions of this work is to develop a framework of handling an important category of ontology alignment based on the logical composition of classes, especially under a case that one class from a certain domain becomes a logic prerequisites (assumption) of another class from a different domain (commitment) which only happens if the class from the first domain becomes valid. Under this logic, previously un-alignable classes or miss-aligned classes can be aligned in a significantly improved manner. A well-known rely/guarantee method has been adopted to clearly express such relationships between newly-alignable classes. The proposed methodology has be implemented and evaluated on a realistic case study.

Declaration

Whilst registered as a candidate for the above degree, I have not been registered for any other research award. The results and conclusions embodied in this thesis are the work of the named candidate and have not been submitted for any other academic award.

Acknowledgment

I wish to express my love and gratitude to all my family, whose love and support have always been with me. In particular, I would like to give my special thanks to my beloved parents for their love and their continuing moral support throughout my studies. They had more faith in me than could ever be justified by logical argument.

Studying at the University of Portsmouth was the most rewarding experience I have ever had. To me, this is certainly related to the high standard of the facilities offered to students, the friendly atmosphere among the research students, and above all the excellence of supervision. For this reason I would like first to express my deepest appreciation and lasting gratitude to Dr Shikun Zhou His wide knowledge and his logical way of thinking have been of great value to me. His understanding, encouragement and personal guidance have provided a good basis for the present thesis. Without his guidance the successful completion of the research and this thesis might have been a very difficult task. His critique and helpful ideas showed us the way to proceed. I am really grateful for that. I really appreciate his positive comments for improving and bringing out the optimum consequences from my endeavours. Without his guidance and support I would never have been able to organize and complete my task well and within time constraints. The only thing I can tell him is thank you for everything.

Finally, thanks to all those who provided immense support, valuable advice, and inspired me through the journey of my PhD.

Publications

- S. R. Azzam , S. Zhou Applying Rely/Guarantee in Compositional Ontology Alignment The The Global Science and Technology Forum (GSTF) Journal on Computing (JoC) october 2012
- S. R. Azzam , S. Zhou Semantic Web Approach for Organizations Management International Journal of Engineering Research and Technology (IJERT) August - 2012 (Vol. 1 , Issue 6)
- S. R. Azzam, S. Zhou Assessment of Ontology Alignment Methodologythe 7th International Conference for Internet Technology and Secured Transactions (ICITST-2012) in London, UK
- S. R. Azzam, S. Zhou Implementation Methodology of Rely/Guarantee Plug-in for Protégé the 7th International Conference for Internet Technology and Secured Transactions (ICITST-2012) in London, UK
- S. R. Azzam , S. Zhou Ontology Alignment: A Review and its Future 2012 4th International Conference on Computer Technology and Development (ICCTD 2012)
- S. R. Azzam , S. Zhou Types of Representation knowledge Structure International Conference on Electrical Engineering and Computer Science date 16 –august 2012 shanghai
- S. R. Azzam , S. Zhou Developing compositional ontology alignment framework Internet Security (WorldCIS), 2012 World Congress on Date: 10-12 June 201
- J. Vipoopinyo, S. R. Azzam, S. Zhou Business Intelligence for Financial Information: An Overview of XBR (Proceedings of the Annual International Conference on Business Intelligence and Data Warehousing (BIDW 2010-2011)
- J. Vipoopinyo S. Zhou S. R. Azzam Business intelligence feedings on pervasive computing: An overview of XBRL Pervasive Computing (JCPC), 2009 Joint Conferences on Date: 3-5 Dec. 2009
- S. R. Azzam, T. Ayodele, J. Vipoopinyo, S. Zhou Integrating business policy into system engineering processes using ontology Pervasive Computing (JCPC), 2009 Joint Conferences on Date: 3-5 Dec. 2009

Table of Contents

Abbreviations

The following abbreviations are used throughout this thesis:

Ass/Com	Assignment/Commitment
BPM	Business Process Management
CRM	Customer Relationship Management
DAML	DARPA Agent Markup Language
DDR	Double Data Rate
DDR2	Double Data Rate 2
DDR3	Double Data Rate 3
DIMM	Double In-line Memory Module
ERM	Entity-Relationship-Models
ERP	Enterprise Resource Planning
ETL	Extraction, Transformation and Loading
FOAF	Friend of a Friend
KPI	Key Performance Indicators
OIL	Ontology Inference Language
OMG	Object Management Group
OMIEN	Ontology Mapping Enhancer
ONION	ONtology compositION system
OOL	Object Oriented Language
OWL	Web Ontology Language
OWL-DL	Web Ontology Language – Description Logic
RDBMS	Relational Database Management System
RDF	Resource Description Framework
SBO	Semantic Bridging Ontology
SBRD	Semantic Bridge for Relational Databases
SDK	Software Development Kit
SIMM	Single In-line Memory Module
SQD	Semantic Query Decomposition component
SRM	Supplier Relationship Management
SWRL	Semantic Web Rule Language
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Extensible Markup Language
W3C	World Wide Web Consortium
WWW	World Wide Web
XMI	XML Metadata Interchange
XML	Extensible Markup Language

List of Figures

List of Table

Chapter 1 Introduction

In the context of the Semantic Web, ontologies refer to the consensual and formal description of shared concepts in a domain. Ontologies are said to be a way to aid communication between humans and machines and also between machines for agent communication. The importance of ontologies for providing a shared understanding of common domains, and as a means for data exchange at the syntactic and semantic level has increased considerably in the last years. Therefore, ontology management becomes a significant task to make distributed and heterogeneous knowledge bases available to the end users. While full automation is the ultimate goal, not everything can be done by machine, user interaction is still essential in order to control, approve and optimize the alignment results. Ontology alignment is the process where for each entity in one ontology One of the main reasons that ontologies are aligned is to share a common understanding of the structure of information among people or software agents.

1.1 Problem Statement

The semantic interoperability is defined as changing and reusing of exchanged data with the intended meanings by the vision of Semantic Web systems. This state is tedious and difficult to achieve among different information systems. It is an obvious fact that distributed and heterogeneous environments will have inevitable chances of error in different applications.(Berners-Lee,Hendler, &Lassila, 2001).

At the level of syntax, structure and semantics, the heterogeneity of information occurs. When different data formats are used, heterogeneity problem occurs that is known as syntactic heterogeneity. Standardized formats like extensible markup language (XML) are used to resolve the issue (Bray, Paoli, &Sperberg-McQueen, 1997), also the Resource Description Framework (RDF) / Resource Description Framework Schema are used (RDFS) (Brickley&Guha, 2000; Lassila ,&Swick, 1999;McBride, Staab& Studer,2003)and Web Ontology Language (owl) (Dean,Connolly, van Harmelen, Hendler, Horrocks,McGuinness,Patel-Schneider, & Stein,2002; McGuinness& Van Harmelen, 2004; Smith,Welty&McGuinness,2004), to express the data in a uniform way, the automatic processing of shared information gets easier.

Even in the syntactic homogenous environments, the structured heterogeneity occurs when the information gets on second level. Obviously, the structural heterogeneity does not overcome the standardization of the format. For instance, the model motor vehicles are classified in few categories but the other would be distinguished on the basis of the physical structure or weight and it makes fine grained distinctions among the vehicles. In order to solve structural heterogeneity problems, manual encoded transformations and middleware components are used. When the two ontologies do not share the equal interpretations of information, semantic heterogeneity occur, the level of heterogeneity is partially solved when two different conditions prevail.

To overcome heterogeneity, defining the relation alignment is a common approach. By transforming the expression of one ontology with the other in a compatible form, these relations can be used.

At any level, this could happen:

- syntactic: semantic preserving transducers is one way
- terminological: Lexical information mapping
- conceptual: representing general transformations (complete set of prover for few languages);

In order to align the Ontology, many worldwide researches are being conducted to work on algorithms.

To acquire the development of knowledge based systems, ontology matching has a central role. Ontology matching is also known as ontology integration, semantic integration, ontology mapping). In a decision to use ontologies in information systems, Semantic Web is easier to approach which is a new technology. In recent years, the availability of increased ontologies is resulted as new trends have arrived in development of new ontologies. To integrate into the system, there are some essential characteristics of ontology that is the reusability or using the existing ontology in a new developed system.

Ontology matching is a problem that has initiated many researches. The development of many approaches ends up in focusing two the two aspects

- Using and matching the elements of ontologies lexically. The first approach is to detect relatedness between the elements by string-based and linguistics method and this all is done by the string similarities among the labels.

- By detecting the similarities within the ontologies. There is a main limitation in both the aspects that does not have lexical similarity or the structure of ontologies are related either or could not prove evidence.

This research focused on using assumption/commitment (Ass/Com) method to form ontology alignment when matching ontologies. Assumption/commitment is an existing method however mainly use for computational logic, It's idea and the paradigm has been adopted. we showed that the use of assumption/ commitment can compensate for lack of structure and lexical overlap, improves the matching result. While using much larger and detailed ontology Experiments have been conducted of aligning business ontology and engineering ontology,. The results of our experiments confirmed that the assumption /commitment can significantly boost the performance of the Alignment process.

1.2 Research Goals

The goal of this Thesis is to

- Give comprehensive review of Semantic web , ontology , ontology alignment
- Demonstrate Semantic modelling toward business users
- Demonstrate the proposed Methodology

1.3 Major Contribution

This thesis makes the following contributions:

- In–depth review to the definition and structure of ontology. Also, various operations over ontologies and a different set of ontology matching methods have been proposed A detailed review of the ontology languages which are used to express ontology over the Web; all these terms were shown in order to provide a basic understanding of ontologies and of description logics, which are the basis of ontology languages. A comprehensive review of existing ontology alignment tools. Several existing ontology mapping methods were analysed and compared, since before creating a new approach it is essential to fully understand related work. Here, this means both theoretical work and existing approaches to the alignment of ontologies or other well-defined structures, including their weaknesses.

- An analyses of business and engineering organizations hierarchy, has been semantically model to show the advantages in integrated semantics to business process Semantic modelling it allows Business users to be more specific by adding more formality and ensuring consistency in business definitions. In essence this mean it describes your business terms and relating them to each other , inserting them into a hierarchy / taxonomy and adding rules Semantic data model serves that purpose. It's a user-oriented metadata layer that comes on top of the data and enriches it. Business developers model the data in such a way that it represents something meaningful to the end-users. The quality of the analytical solution depends on two things: the richness or the capabilities of the underlying semantic model and the knowledge business developer possesses regarding that model.

- Rely/ guarantee which is the technique used in parallel computing this research redefines it's use for Ontology Alignment in Algorithmic Way .The method assumes that not only a component is verified by just satisfying all of its commitments, but also verifies all the assumptions, being exposed from the internal or external environment. We adopted the idea and the paradigm so that the method can now be applied to business domain knowledge expression for machine learning /intelligent understands purpose.

1.4 Thesis Organisation

The remainder of the Thesis is organised as follows, Chapter 2 introduces the research background chapter 3 discusses related work of ontology alignment Techniques and Mapping Algorithms, discusses mismatching between ontologies and presents existing matching techniques, Chapter 4 , Show the design stage of business and engineering hierarchy structure , then it's semantic modelling process .Chapter 5 evaluates the ontologies, and applies traditional alignment methods Chapter 6 introduces the methodology taken in this research project to Develop compositional alignment method Chapter 7 presents the evaluation Chapter 8 proposedutilities to potential tool components.This thesis concludes in Chapter9 by summarizing the thesis and pointing the future research directions.

Chapter 2 Background of Semantic Web and Ontology

2.1 Introduction

This chapter gives a brief introduction to the subject of the Semantic Web, as well as the definitions of ontology. Also in this chapter, many ontology languages are explained in order to give an indication of the strengths and weaknesses of each language, which will help in determining the most suitable language for our purpose. The main concepts and structure of ontology are provided in order to specify the criteria of ontology design. Finally, the usability of ontologies in different software areas is discussed.

2.1.1 Semantic Web

The Semantic Web is an extension of the current World Wide Web. Its main purpose is to allow people to share information, regardless of which application or website they are using. In fact, the Semantic Web will provide a common framework which makes data available for reusing and sharing. This can be done applying two methods. The first method is to create common formats for data from different sources. This method is different from the way current websites exchange information because data in Semantic Web environments is ready to be integrated. The second method involves generating languages that show the relationship between data and real world objects. Current efforts are being done by researchers and other businesses (e.g. W3C) to produce such a general framework and languages for the Semantic Web with the ability to share and reuse data. Tim Berners-Lee (WWW inventor) stated that the Semantic Web is “an extension of the current Web, in which information is given a well-defined meaning, better enabling computers and people to work in cooperation” (Berners-Lee, Hendler, & Lassila, 2001).

Therefore, the Semantic Web (Fensel, 2001; Paolucci, Kawamura,Payne,& Sycara, 2002) has the ability to represent data in a rich, meaningful and readable form, which can be utilised by both humans and machines. Furthermore, in order to facilitate knowledge sharing, the Semantic Web provides different services such as publishing, discovering, and automatic annotation. Consequently, many tools and applications have been implemented to achieve the interoperability aim of the Semantic Web.

Previously, semantic web was known to have such a framework that enable content over the web to actually make them indexable. For instance, any text can be searched over the web by visiting a searching engine website. Textually, if any content needed tobe found d, we can simply type the text

and search engine shows result of the matching content. With the evolution of semantic web, the search engines are therefore able to find context based search results. This tremendously results in accurately fetching the information, for that, the requirement is to publish the content in a required framework.

With the passage of time, the idea of semantic web is to actually come up with such a framework that content based searches, any shared information, whether in the form of text, images, audios, videos as well the complete end-to-end systems can be published over the semantic web and can be utilized by others. The only issue here is the proper management and control over the publication of contents over semantic web. For the reason, different frameworks in the form of languages and structures are proposed, which is discussed later.

2.1.2 Semantic Web layers

The Semantic Web (Tidwell, 2000) has evolved to cope with the heterogeneous and the distributed environment of the Web. To ensure the success of upgrading the current Web to a superior standard, the content should be reformatted in a machine understandable style. Data annotation is the mechanism which facilitates this aim. The Semantic Web architecture is depicted in Figure 2.1.

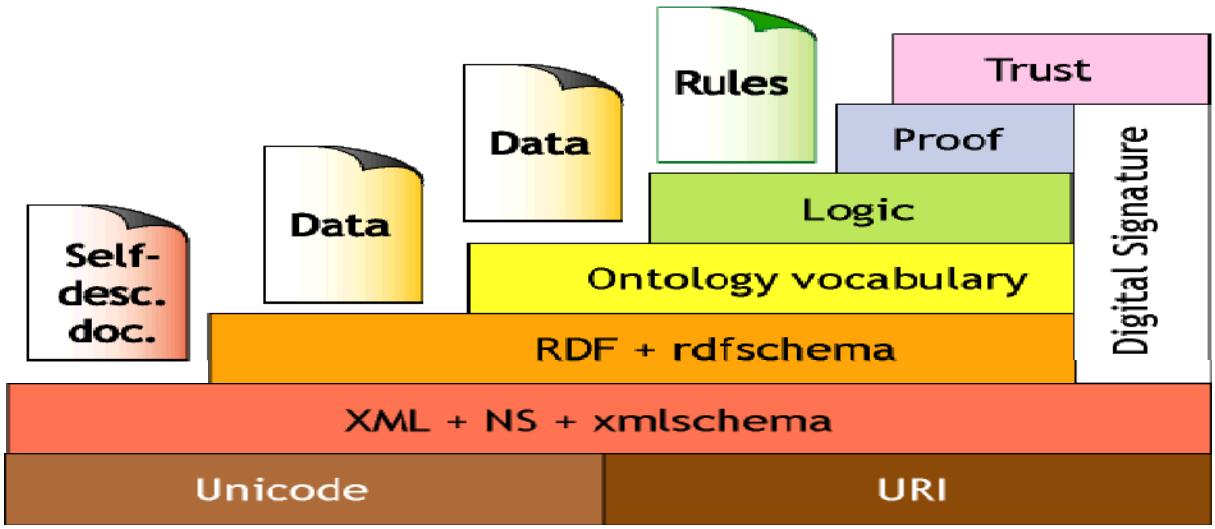


Figure 1.1: Semantic Web Architecture

The description of architecture layers is clarified below:

URI and Unicode: these layers have two functions: first, identify and locate resources available in the Web, a uniform system of identifiers (URIs) can facilitate these functions. They are also responsible to grant unique names to the resources. The Unicode is a standard for computer character representation.

Extensible Markup Language (XML) is a machine-readable markup language. One of the many reasons for this language being widely accepted in the WWW community is the flexibility of its format. Therefore it facilitates the current Web services such as e-businesses and e-commerce. It also helps in expanding the activity of software. XML Namespace is the document declaration. Users utilise the freedom of the XML format in arranging their documents with any arbitrary structure. However, the XML structure does not provide any semantic indication. In fact, XML language can be utilised by its syntax only.

Resource Description Framework (RDF) is considered to be the first Semantic Web layer. Since RDF can describe some semantic representation for information, metadata about Web resources will gain machine-accessible. In fact, RDF is a graphical model that can utilise the URI Web resources definitions in representing relations among resources. The developed version of RDF is an RDF Schema which supplies the class hierarchy structure and the relationship between classes and objects.

Ontology Vocabulary is a language which provides a common vocabulary and grammar for published data as well as a semantic description of the data used to preserve the ontologies and to keep them ready for inference. Ontology means describing the semantics of the data, providing a uniform way to enable communication by which different parties can understand each other.

Logic and Proof In the Semantic Web, the building of systems follows a logic which considers the structure of ontology. A reasoner could be used to check and resolve consistency problems and the redundancy of the concept translation. A reasoning system is used to make new inferences.

Trust is the top layer of the proposed Semantic Web architecture. This layer concerns the reliability of Web information. Consequently, the quality will be assured.

2.2 Ontology

Today the Internet websites need to be enhanced by formal semantic representations for the current web content. When this happens, the Web will be semantic and suitable for both human and machine use.

One of the Web roles is to build a source of reference for information on several subjects, while the Semantic Web is intended to annotate the current Web with semantic meaning. Ontology is the essential part that facilitates the Semantic Web, since it allows distributed systems to share information. This means that ontologies can help unrelated applications to collaborate with each other.

2.2.1 Ontology Definition

Gruber defined ontology early on as "an explicit specification of a conceptualisation" (Gruber, 1993). Since then the definition has been amended to express other features. The new definition of ontology is: a formal explicit specification of a shared conceptualisation (Mihoubi, Simonet, 2000), where:

I. Formal means building ontologies in machine understandable language; it should be defined using logic-based languages. Formality is essential to eliminate the vagueness of informal notations (e.g. natural language).

II. Explicit specification stands for providing descriptive names (i.e. terms) for ontology concepts and clear characterisation for the constraints on these concepts. It also includes the definition of the relation which shows how a concept relates to other concepts.

III. Shared indicates that different communities across the Web are generally agreed on the meaning of domain concepts (consensual knowledge), thus allowing applications to exploit and reuse ontologies.

IV. Conceptualisation means ontology concepts appear in comprehensible form; i.e. domain knowledge is defined in an abstract model. This abstract model focuses on how to capture people ideas about objects from real world, usually in a specific subject in a particular domain.

2.2.2 Ontology Objectives

- To agree on the understanding of the shared concepts of information structure among different communities.

- To make a domain knowledge analysable and reusable.

- To specify domain assumptions in an explicit manner.

- To make a partition between operational knowledge and domain knowledge.

- To analyse domain knowledge.

Those objectives (Noy & McGuinness, 2001) can only be applied on domain ontologies (i.e. global ontology). While application ontologies (i.e. local ontologies) can benefit from these characteristic after mapping them to the global ontologies.

2.2.3 Ontology Representation

Ontology consists of four principal elements: concepts, relations, axioms, and instances. The definitions of each element are presented below:

- A Concept (also known as a term or a class) is the essential abstract component of a domain. Typically, the class represents a group of common properties owned by many members. Also, classes are arranged in hierarchical graphs on two levels. Higher level classes are called parent classes and the subordinate levels are called child classes. A graph of concepts might organise classes in a lattice or taxonomic view; for example, the class 'Faculty' could have many subclasses, such as 'Department' and 'College'. Moreover, the concepts might have many different distinguishable properties.

- A Relation (also known as a slot) is used in ontology structure to provide a declaration for the relationships between concepts in a specific domain. In order to specify the two classes involved in a particular relationship, one of them will be described as a 'Domain' and the other one as a 'Range'; for instance the relationship 'Work' can have the concept of 'Employee' as a domain and 'Faculty' as a range.

- An Axiom (sometimes called a facet or role restriction) is utilised in ontology by forcing restrictions on values of both classes and instances. Logic-based languages, such as first-order logic, have been developed in order to express these constraints. Furthermore these languages can be facilitated as the verification process for the consistency of ontology structure.

- An Instance (also known as an individual) is a relationship between ontology concepts and relation and their real values; for instance 'Saudi Arabia' could be an instance of the class 'Asian countries', or simply 'countries'.

2.2.4 Structure of Ontology

The general and formal structure of ontology is demonstrated as follows:

$$5 - tuple\ O: = (\mathrm{C}, \mathrm{H}^{\mathrm{C}}, \mathrm{R}, \mathrm{H}^{\mathrm{R}}, \mathrm{I}),$$

Where:

- C: refers to a set of concepts ("rdf:Class"). These concepts are organised in a class/subclass (subsumption) hierarchy.

- R: stands for a set of relations that connects concepts ("rdf:Property"). All relations in ontology are binary types between only two concepts.

$$\mathrm{Ri} \in \mathrm{R} \text{ and } \mathrm{Ri} \rightarrow \mathrm{C} \times \mathrm{C}.$$

- H^C : represents the hierarchy of ontology concepts by using the special case of relation. ("rdfs:subClassOf").

$$\mathrm{H}^{\mathrm{C}} \subseteq \mathrm{C} \times \mathrm{C},$$

where H^C(C1, C2) denotes that C1 is a sub-concept of C2.

- H^R: depicts a relation hierarchy in the form of a relation ("*rdfs:subPropertyOf*").

$$\mathrm{H}^{\mathrm{R}} \subseteq \mathrm{R} \times \mathrm{R},$$

where H^R (R1, R2) denotes that R1 is a sub-relation of R2

- I: supplies the concepts with instances in a particular domain ("rdf:type").

Generally, taxonomies, thesauri, and ontologies are types of methods that represent the Semantic Web classification of domain concepts.

First, taxonomy is a way of organising the vocabulary of concepts in tree form or hierarchical models; this includes an explicit definition of domain concepts and their relationships. In the taxonomy method, only the relation of "subclass" and its inverse "superclass" are allowed. Second, thesaurus is used to represent vocabularies while considering relations between terms (Corcho & Gómez-Pérez 2001). In fact, these terms are the relationships obtained by taxonomy methods with richer conceptual

descriptions. Therefore, a thesaurus can be considered as an extension to the taxonomy method with more semantics and expressive features. Therefore a thesaurus includes some relationship characteristics, such as homography, equivalence and hierarchy. Any lexicon database, such as WordNet, can be considered as a thesaurus, since it captures concepts and preserve their semantic relationships.

Third, ontologies extend the taxonomies idea with rich relations between concepts and properties; also the axioms play a significant role in presenting ontology restrictions. Moreover, ontologies describe a domain by defining its concept set; therefore they can be seen as the skeletal establishment for a knowledge base.

The ontology community categorises ontologies in terms of their structure into two types: lightweight and heavyweight ontologies. Both types share some contents which include concepts, relationships between concepts, and attributes. However, axioms and constraints are considered only in heavyweight ontologies. Many kinds of languages can participate in implementing both lightweight and heavyweight ontologies (Gotoh, 1982). In terms of language formality, ontologies can be divided into four groups:

-Highly informal: mostly described by natural language; however, this type of language might not be considered as an ontology if it lacks machine-readability;

-Semi-informal: a natural language used to describe ontology structure, and it can be utilised as machine-readable language;

- Semi-formal: a language such as RDF expresses ontology structure and some restrictions in formal definitions;

-Rigorously formal: ontologies provide definitions of terms with formal semantics, as well as properties with essential characteristics; also, it shows a notion of a completeness theorem (e.g. Web Ontology Language OWL) (Taye, 2009).

Also ontology can be classified into global and local ontology. Global ontology can be seen as shared vocabularies between many local ontologies,it represents the global schema of a domain. However, local ontologies concentrate on different sub-domain fields. Moreover, a local ontology's function is to express more specific information whereas a global ontology contains only general level of concepts. Finally, a local ontology allows different groups to use their own terminology whereas a global ontology utilises the agreed terminologies.

In fact the expressiveness of ontology measured by the degree of explicit metadata is captured from the domain knowledge. Usually, ontologies try to catch the semantics of a particular domain, and the more relations and constraints can be captured, the more ontology may be considered expressive. The important factor that affects the expressiveness an ontology is language specifications which limit the language abilities (Uschold & Gruninger 2004). In order to provide ontology with a formal semantic structure, it should be expressed in knowledge-based language. After that, ontology can capture domain knowledge specifications and become machine-processable, and eventually appear in a well-defined design (Berners-Lee, Hendler, & Lassila, 2001).

2.2.5 Criteria for ontology design

Objective criteria is required to control the ontology design, since each designer makes his own design decisions in representing anything in the ontology. Therefore the criteria given below can be utilised in assessing ontology design.

- **Clarity**:

The ontology should express the intended meaning of the defined terms. In order to reach this aim, the definitions should be independent from social cases or computational specifications.

- **Coherence:**

Coherence implies that the inferences should be consistent with the ontology definitions.

- **Extendibility:**

New concepts can be defined while considering the existing terms without requiring ontology amendments.

Minimal encoding bias:

To ensure this aim, separate the ontology concepts from specific symbol encoding. Also, the definition of ontology terms should be formalised, since many knowledge agents may use different encoding systems.

Minimal ontological commitment:

An ontology should capture the minimal sufficient vocabularies for describing a domain. Nevertheless, it should give the participants freedom for customisation (Gruber, 1993).

2.2.6 Steps in Ontology Creation

In designing an ontology we should consider that:

- Devolving ontologies should go through iterative procedures.

- Any domain does not have one correct modelling design, since specifications vary from one application to another, and the future extensions might have new requirements.

The method used for creating ontologies, as described below, consists of a set of steps, with corresponding heuristics. The ontology designer should go through a series of steps which are known as the lifecycle of ontology:

- Specify the scope and the domain of the ontology;

- Reuse concepts residing in existing ontologies;

- Identify the significant concepts of the ontology;

- Define the class hierarchy;

- Enumerate properties for each class;

- Define the facets of relationships;

- Create ontology instances.

A class hierarchy can be created using three approaches (Uschold & Gruninger, 2004). These approaches concern the method of producing classes and class hierarchy; however, the creation of properties is not considered. These approaches focus on building the taxonomy aspects of ontology classes, as the majority of ontologies designs do. The three design approaches are (Vögele, Visser, & Stuckenschmidt, 2001):

Top-down:

This process determines the general terms of the domain, and then creates a specification of each term. In other words, we create the most general classes, and subsequently create the subclasses. Also, this process includes a categorisation for each subclass. Here each ontology designer tries to create the best classification to represent the application needs from his perspective.

Bottom-up:

This process defines the most specific concepts, followed by the grouping of these concepts into more general concepts. In other words, the leave classes of the hierarchy are defined, then generalised into a common superclass which could be a subclass of another superclass and so on.

Combination:

This process is based on the idea of integrating both bottom-up and top-down approaches. First, it defines the most significant concepts, and afterwards applies taxonomy methods by generalising or specialising these defined concepts: for example, dividing classes into three levels; top level, middle level, and specific level. Finally, we relate top level classes and specific level classes through middle level classes.

None of these three approaches is superior to the others. Rather, the choice of approach depends on the designer's personal view of the domain: if the designer establishes the design by the grounded specific terms then the bottom-up methodology is more appropriate, whereas if the designer has a system where the most general concepts are first in the hierarchy, then the top-down methodology is better. Some researchers claim that the easiest way to develop an ontology is by using the combination approach, since the middle concepts are more descriptive in the domain. In our method we chose the top-down approach, because the concepts already existed from the entities in the database model.

2.2.7 Challenges in Building Ontologies

There are many challenges in building an ontology. These challenges might affect global or local ontology, or both.

The main challenge is to have the ontology completely designed which means making sure all the relevant concepts in the domain are included. There are many suggestions for overcoming this challenge. The first is to enlist the participation of a domain ontology expert, especially for designing the global ontology. However, in the case of local ontology, the conceptual design of the database is sufficient and there is no need for a domain expert; any developer familiar with the domain is suitable. The second suggestion is to identify the most frequent concepts, which involves applying this technique to different real cases. Using real cases and analysing them will help the designer to infer all the related concepts of the domain. The third suggestion is to identify the synonyms of each concept with the help of a thesaurus e.g. WordNet: this will help in limiting the problem.

The second challenge is ontology evolution, which refers to the changes in ontology over time. These changes include the adding, modifying, and deleting of some concepts. Since the domain itself evolves over time, the design of the ontology will be affected.

The third challenge is the risk that the domain is too large. Hence the expert should determine the scope of the domain. This can be done by breaking down the domain into sub-domains; then for each sub-domain a local ontology can be created. The idea of local ontologies is to help manage the relationships between concepts. Further to this, local ontologies can be mapped to a global ontology which shows the whole domain. Another way to provide a view of the whole domain is to integrate local ontologies with the global ontology. With either method, the domain expert should ensure that the relationship between concepts is consistent.

2.2.8 Ontology Description Languages

In fact, ontology languages are considered to be the foundation of knowledge-based systems. Also, they can express knowledge specification in a rich and intuitive way, and in a machine understandable form.

Resource Description Framework (RDF)

RDF language (Beckett, 2000; Lassila, & Swick, 1999; McBride, Staab, & Studer, 2003) is a standard Semantic Web language which provides a description for Web resources; also, it is frequently used to represent data, properties, and Web resources content in order to exchange knowledge over the Web. It has been developed in a machine-understandable way in order to achieve interoperability between applications.

RDF is recommended by the W3C. It utilises the URIs in identifying resources, and it adopts the XML syntax. However, RDF has been designed for representing semantics.

Many applications utilise the RDF language; for instance, the improvement of search engines qualification stemmed from RDF ability in discovering Web resources. Another example is the intelligent software agents which facilitate RDF in exchanging and sharing knowledge.

The syntax of RDF model is built on three elements: subject, predicate, and object. In fact, the subject is any resource that can be defined by URI. The RDF statement would have the following interpretation, which is <subject> has a property <predicate> valued by <object>.

Resource Description Framework Schema (RDFS)

This ontology language adopts XML syntax and is built on top of an RDF model. The RDF Schema provides additional vocabularies to the core of RDF in order to gain more expression.

RDF Schema is effective in defining vocabularies required by applications. In fact it is a set of RDF resources which express properties exists in other RDF resources in machine-understandable format. Moreover, the RDF Schema expresses descriptions for classes and properties, as well as determining specifications for relations between properties and classes.

In general, RDFS may be identified by the URI reference of http://www.w3.org/2000/01/rdf-schema#, as well as a defined namespace 'rdfs', whereas the RDF namespace is 'rdf' and its URI reference is http://www.w3.org/1999/02/22-rdf-syntax-ns#.

Class, property and property constraint are the three elements of ontology which can be described by RDF /RDFS. RDF is a good basic language foundation which can be utilised in building other languages. However, it has a limited ability in resource description such as cardinality, domain and range localised constraints, existence descriptions, and property characteristics (transitive, symmetrical, and inverse). For example, 'male' and 'female' are two subclasses of the human class, and separating them from each other is impossible since RDF lacks the disjointed vocabulary. Therefore, it can be considered the expressive power of RDF to be limited, as mentioned in (Vögele, Visser, & Stuckenschmidt, 2001). To overcome these limitations, RDFS was invented. RDFS can be utilised in building class hierarchies (subclasses) and property hierarchies (sub-properties); also, it enables relationship construction and restricts domain and range, besides providing instances for classes

However, RDFS is not sufficiently expressive in further aspects. It is unable to describe equality and inequality between properties. Also, it cannot define enumeration. Union, intersection, unique, symmetric, transitive and inverse are relation characteristics which cannot be expressed by RDFS. Consequently, many ontology languages, such as DAML+OIL, and OWL, have been developed to tackle these deficiencies.

Ontology Interchange Language (OIL)

Ontology Interchange Language: OIL is a markup language built upon RDFS. One of its important features is providing modelling primitives which have been utilised by frame-based ontologies [Uschold, M., Gruninger. (2004)].

Annotated DAML+OIL Ontology Markup

DARPA Agent Markup Language (DAML) + Ontology Inference Layer (OIL), or DAML+OIL (Lassila, & Swick, 1999; McBride, Staab, & Studer, 2003) is a semantic markup language. It is an extension of RDF which tries to reduce some of the RDF deficiencies through building richer primitives. Also, it has the

ability to provide a description for domain structure in terms of classes and properties. One of its powerful characteristics is that it can formalise the meaning of language terms by applying the theory of a Description Logic (DL) model (Maedche, 2001).

The production of DAML+OIL is to amalgamate European languages with the American proposal.

Like other Semantic Web languages, DAML+OIL has its own limitations, one of which is the lack of property descriptions; for example, DAML+OIL does not offer composition or transitive closure, and does not allow comparison between data values. Moreover, it represents collection type by sets only; i.e. it does not offer lists or bags.

Web Ontology Language (OWL)

OWL language according to (Antoniou, 2004; Dean, Connolly, van Harmelen, & Patel-Schneider. 2002; McGuinness & van Harmelen, 2004; Patel-Schneider, Hayes, & Horrocks 2004; Smith & Welty, 2003; Dean & Schreiber, 2003), is a standard Semantic Web language for processing information over the Web, recommended by W3C. It facilitates RDF vocabularies and XML syntax. However, it overcomes drawbacks appearing in other languages such as RDFS and DAML+OIL. It also has the ability to provide an interoperable data environment for different domains and communities.

If a comparison takes a place between OWL and RDF, similarity between them can be found; however, OWL shows more capabilities with regard to semantic language vocabulary which enables it to provide machine interpretability. Evidently, OWL vocabularies have richer semantics than RDF and RDFS in representing property characteristics and the hierarchy of classes and properties. Overall, OWL was invented to utilise XML syntax and to adopt RDF and RDFS primitives; for example, it uses RDF terms and meaning in defining classes and properties. It was also designed to overcome RDF weaknesses.

OWL language is classified into three sublanguages, with various aspects to supply different goals: OWL Lite, OWL Description Logic (OWL DL) and OWL Full (Paolucci, Kawamura, Payne, & Sycara, 2002). The simple version of OWL is OWL Lite which provides classifying hierarchies and forms the constraints in a simple way. It expresses only maximum cardinality of relationships in a value of 0 or 1, and thus produces an easy design. The restriction is limited in OWL Lite, since it lacks some terms such as ‘union’ and ‘negation’. Therefore, it has the lowest expressiveness of OWL sublanguages. OWL Lite is considered as a sublanguage of OWL DL.

OWL DL utilises Description Logic in describing relations between objects and their properties. It tries to preserve the completeness of computational properties; therefore it is most expressive in describing concepts and relationships.

The sublanguage OWL Full provides the most expressiveness and the syntactic freedom of RDF but without preserving guarantees on computational complexity. OWL Lite and OWL DL are sublanguages of OWL Full.

OWL has two kinds of syntactic representation forms. The first one, uses this syntax throughout the examples, is known as the exchange syntax. This form represents ontology as a set of XML or RDF triple which is ready to be published and shared over the Web. However, the exchange syntax form is difficult to trace since it contains a large number of syntactic constructs for each class or property. The second form of OWL syntactic forms is the abstract syntax. This form is abstracted from the exchange syntax and appears similar to relational schema. One of the characteristics of this form is the frame-like style which constructs all the information about class or property in one collection. Thus it facilitates the assessment steps of the produced ontologies. The evaluation chapter utilises this syntax. Table 2.1 shows a comparison between the above discussed ontology languages, as well as the limitations of RDF, DMAL+OIL in contrast to OWL language. Therefore, OWL is considered to be the most expressive among Semantic Web languages.

The expression	RDF/RDFS	DAML+OIL	OWL
Class	√	√	√
rdf: Property	√	√	√
rdfs: subClassOf	√	√	√
Rdfs: subPropertyOf	√	√	√
Rdfs: domain	√	√	√
Rdfs: range	√	√	√
Individual	×	√	√
Same Class As	×	√	√
Same Property As	×	√	√
Same Individual As	×	√	√
Different Individual From	×	√	√
Inverse Of	×	√	√
Transitive Property	×	√	√
Symmetric Property	×	√	√
Functional Property	×	√	√
Inverse Functional Property	×	√	√
All Values From	×	ToClass	√
Some Values From	×	hasClass	√
Min Cardinality	√	√	√
Max Cardinality	√	√	√

Table 2.1: Comparison between ontology languages

2.2.9 Ontology Applications

To build a common understanding and consensus in knowledge areas, ontologies have become a 'hot topic' of research. Ontology first appeared in Artificial intelligence (AI) laboratories, before being used in other fields. The following are examples of ontology use:

I. Semantic Web (Berners-Lee, Hendler. & Lassila 2001): Ontology is the key enabling technology in the Semantic Web to support information exchange across distributed environments. It meets the promise of the Semantic Web in representing data in a machine-processable way.

II. Semantic Web Service Discovery (Berners-Lee et al., 2001): Ontology helps in describing the merchandise in the E-Business services, which also includes trying to discover the most suitable match for the requester obtained and establishing a communication between buyer and seller. These facilities are also applicable in an E-Commerce environment (Fensel, 2001).

III. Artificial Intelligence (Mihoubi & Simonet 2000): The AI research community was first to develop the idea of ontology. The intended goal of ontology is to produce sharing and reusable facilities for knowledge that will ensure communication between services, programs, or organisations across a given domain.

IV. Multi-agent (Fensel, 2001): in order to ensure the success of agent communications, there is a need for common understanding for domain knowledge and shared vocabularies. These requirements are offered in ontology models and any misunderstanding is thereby reduced.

V. Search Engines (Paolucci., Kawamura, Payne, & Sycara, 2002): Ontologies help Internet search engines in capturing common terms and defining synonyms for terms which usually exist in thesauri.

VI. Interoperability (Corcho, & Gómez-Pérez. 2001): In the Web, systems have two aspects, heterogeneous and distributed, and are thus called the "Interoperability problem". In order to resolve this problem, ontology can be used to facilitate interaction between heterogeneous systems by explicating the concept terms of each system in the form of a sharable and machine-understandable ontology.

VII. Systems Engineering: Ontology can be exploited in defining system requirements, and even for the developed systems; it helps in determining the extension purpose or maintenance specifications. Further, it can be used to resolve inconsistency in any system design.

2.2.10 Summary

This chapter introduced the topic of Semantic Web and its importance in upgrading the current Web. Moreover, it discussed the Semantic Web layer. After that, ontology definition was presented. The elements that participate in building the ontology were elaborated. Also the formal structure of ontology was discussed. Then it described the criteria that ensure the success of ontology design and the challenges which need to be overcome. After that, the three methods of building ontology, top-down, bottom-up and combination were mentioned. Then showed that the preferable method is the top-down and how our approach will utilise it in designing an ontology

Chapter 3 Ontology Matching Techniques and Alignment System

3.1 Introduction

This chapter lays these foundations of a standardized vocabulary. Therefore, it starts first explaining the terminology of ontology and alignment. Then the concept of similarity and its meaning for ontologies will be described, on both an abstract and a concrete level with examples.

Overview of related of existing ideas an extensive investigation on related work is necessary before an own innovative approach is developed. This chapter starts with theoretic considerations on alignment. It will then focus on other actual approaches for ontology alignment.

The considered research field of ontology alignment and integration features a big number of terms such as alignment, mapping, mediation, merging, etc. Unfortunately, the definitions from different authors are confusing, partially inconsistent, and at times even contradicting. The goal of this section is therefore to adhere to one definition for each term. Ontology may refer as a language to develop such a platform that provides properties for semantic web. In the past, there has been conducted extensive research of what path will enable the semantic web to share the information as well as reutilizing the same and to make context-based searches. With different proposed solutions, the latest ones is the constructive language OWL Ontology, that enables developing a semantic environment. With OWL-DL Ontology, one can develop comprehensive scenarios to capture entire semantics of the system.

Ontology will therefore is a technique to develop a functional semantic web. The goal is to come up with the solution that machines can understand in order to automate the possible systems, developing an environment for data integrity and harmony by having intra-systems integrated solutions, and auto-decision makers to enable addressing issues. Thus, Ontology is a language to achieve the semantic web.

3.2 Ontology Alignment Definition

By Ontology Alignment, we mean to come up with the Ontology, having sensible meaning with reutilization of objects in other Ontologies. The other Ontologies can be OR cannot be interconnected. The main idea behind the Ontology Alignment is to utilize the already available classes within the Ontology from other Ontologies. This thus reduces the reoccurrence of same classes multiple times to single instance based. In other words, to make them in line with the entire ontology,while comparing two or more Ontologies, the alignment refers to checking the occurrence of classes and instances along with their relationship in both or more Ontologies, thus to make them reutilized from one Ontology to all other, having same semantics. For instance, Name is the class, which can be associated with the Person Class as well as with the Brand class. They both have the same semantics. Syntactically, they are string type with Full Name and Short Name. Both aforementioned classes will utilize the same, therefore can be aligned.

According to the dictionary (Smith & Welty, 2003) gives a general comprehensible sense for alignment. To align something means, "to bring into line". This very brief definition already emphasizes that aligning is an activity after which the involved objects are in some mutual relation. Here we define our use of the term ontology alignment similarly to (Dean & Schreiber, 2003). Given two ontologies, aligning one ontology with another one means that for each entity (concept, relation, or instance) in the first ontology, we try to find a corresponding entity, which has the same intended meaning, in the second ontology. An alignment therefore is a one-to-one equality relation. Obviously, for some entities no corresponding entity might exist.

3.2.1 Definition (Ontology Alignment).

An ontology alignment function, align, based on the set E of all entity e ∈ E and based on the set of possible ontologies O is a partial function

align : $E \times O \times O \rightharpoonup E$

We Write align $O_1\ O_2$ (e) for align (e, $O_1\ O_2$). We leave out O_1, O_2 when they are evident from the context and write align (e) instead. Once a (partial) alignment, align, between two ontologies O1 and O2 is established, we say entity e is aligned with entity f when align{e) = f . A pair of entities (e,f) that is not yet in align and for which appropriate alignment criteria still need to be tested is called a candidate alignment. In this work, if not explicitly mentioned otherwise, alignment is an equality alignment.

Apart from one-to-one equality alignments, as mainly investigated here and most of the related existing work, in real world one entity often has to be aligned not only to equal entities, but based on another relation (e.g., sub-sumption). Further, there are complex composites such as a concatenation of terms (e.g., name equals first plus last name) or an entity with restrictions

(e.g., a sports car is a car going faster than 250 km/h).

3.2.2 Definition (General Ontology Alignment)

A general ontology alignment function, genalign, based on the vocabulary, E, of all terms e € E, based on the set of possible ontologies, O, and based on possible alignment relations, M, is a partial function

Genalign :$E \times O1 \times O2 \rightharpoonup E \times M$

This last understanding is part of the alignment definition researchers have agreed on in the context of Knowledge Web [30], a European network of excellence of leading institutions on semantics technologies. Supporting the transition process of ontology technology from academia to industry is the major goal of knowledge web. This will be achieved through standardization education, and integrating research. Responding to the integration task the partners defined an alignment as a set of correspondences (i.e., quadruples): (e, f, r, l) with e and f being the two aligned entities r representing the relation holding between them and l (additionally) expressing the level of confidence [0.1] in the alignment statement.

3.3 Ontology Alignment Representation

In this section discusses the two possible representation formats that are most accepted and used in the alignment community.

The first possibility is to adhere to existing constructs, e.g., in OWL. OWL provides equality axioms for concepts, relations, and instances: owl:equivalentClass, owl:equivalentProperty, and owl:sameAs. It is also possible to express inequality through owl:differentFrom. The advantage of this representation is that OWL inference engines will automatically interpret the alignment and reason across several ontologies. The downside is the very strict equality. A confidence value cannot be interpreted accordingly. Complex alignments as mentioned before are not possible. The second possibility is based on work of (Patel-Schneider, Hayes, & Horrocks, 2004) The representation uses RDF/XML to formalize ontology alignments. After the general definition of the involved ontologies, the individual alignments

are represented in cells with each cell having the attributes entity 1, entity 2, measure (the confidence), and the relation (normally '='). This representation corresponds to the Knowledge Web definition of alignment (**Definition 3.1.2).** Due to its different parameters, it can easily be used for many alignment applications. Unfortunately, it is not directly in an ontology format. Therefore, an explicit import is required to transform the alignments into a suitable format for inferencing. For this importer one also needs to define how to handle confidence values of an alignment. Alternative representations are the MAFRA semantic bridging ontology(SBO), Contextualized OWL, the rule language SWRL, the OMWG mapping language (OML), and SKOS. An overview thereof is found in (Euzenat, Jerome, Scharffe, Frangois, Serafini, & Luciano 2006).

3.3.1 Ontology Alignment Example

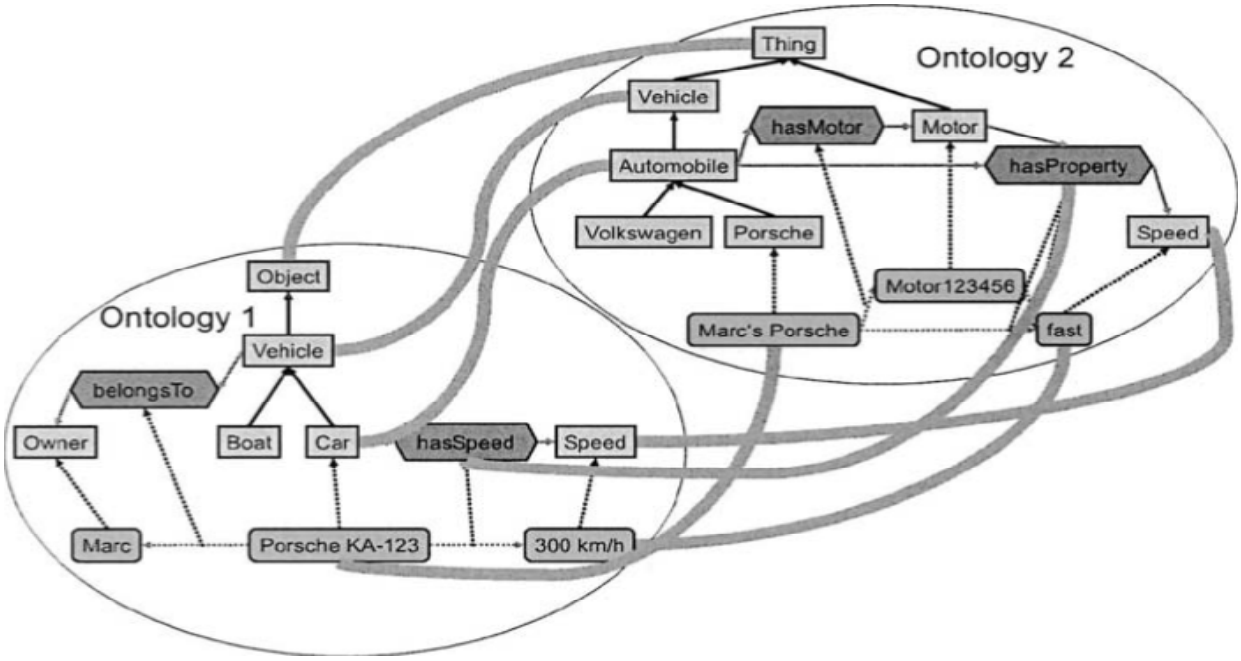


Figure 2.1: Ontology Alignment Example

Ontology O_1	Ontology O_2	Confidence
object	thing	1.0
vehicle	vehicle	1.0
car	automobile	1.0
speed	speed	1.0
hasSpeed	hasProperty	1.0
Porsche KA-123	Marc's Porsche	1.0
300 km/h	fast	0.9

Table 3.1: Alignment Table

The following example illustrates alignments. The example consists of two simple ontologies that are to be aligned. The two ontologies O1 and O2 describing the domain of cars are given in Figure 3.1. The first ontology has already been described in the ontology section; it is our running example. The second ontology covers the same domain but is modeled slightly differently. Beneath an overall thing concept, there exists a vehicle, which in turn has the subclasses automobile, Volkswagen, and Porsche. Further, there is a motor and speed. The automobile has a Motor which in turn has a property speed. A specific Porsche, Marc's Porsche, with the fast Motorl23456 is also represented.

Reasonable alignments between the two ontologies is given in Table 3.1. Each line contains the two corresponding entities from ontology 1 and ontology 2. In Figure 3.1, alignments are represented by the shaded channels each linking two corresponding entities. Obviously, things and objects, the two vehicles, cars and automobiles, as well as the two speeds are the same. The relations of having a speed and property correspond to each other, as they both refer to speed. In addition, the two instances Porsche KA-123 and Marc's Porsche are the same, which are both fast. Whereas the alignments might seem obvious to the reader in this case, the common agreement on alignments is not easy in general.

Returning to the formal definition of alignment, the set of result alignments is given as follows. To distinguish between the two ontologies, ontology 1 is given the namespace ol and ontology 2 the namespace o2.

align = {ol:object —> o2:thing, ol:vehicle -^ o2:vehicle,...}

An excerpt of the alignments is shown in the RDF-based representation of (Patel-Schneider, Hayes, & Horrocks, 2004) in Example below. First the two ontologies and their URIs are defined. In the following cells the actual alignments are shown. The first one aligns ol:object with o2:thing. The confidence measure is very high with a value of 1.0, and the semantic relation between the two entities is equality (in contrast to subsumption or part of).

(Example Alignment Representation in RDF(S)).

```
<rdf:RDF>
<Alignment>
  <xml>yes</xml>
  <level>0</level>
  <type>11</type>
  <onto1>ontology1.owl</onto1>
  <onto2>ontology2.owl</onto2>
  <uri1>http://aifb.uni-karlsruhe.de/ontology1.owl</uri1>
  <uri2>http://aifb.uni-karlsruhe.de/ontology2.owl</uri2>
  <map>
    <Cell>
      <entity1 rdf:resource='ontology1.owl#object'/>
      <entity2 rdf:resource='ontology2.owl#thing'/>
      <measure rdf:datatype='XMLSchema#float'>1.0</measure>
      <relation>=</relation>
    </Cell>
    <Cell>
      <entity1 rdf:resource='ontology1.owl#vehicle'/>
      <entity2 rdf:resource='ontology2.owl#vehicle'/>
      <measure rdf:datatype='XMLSchema#float'>1.0</measure>
      <relation>=</relation>
    </Cell>
    ...
  </map>
</Alignment>
</rdf:RDF>
```

3.3.2 Related Terms

Alignment, has many related terms, which will be defined and differentiated. The definitions are taken from (Dean & Schreiber, 2003; Ding, Ying, Fensel, Dieter, Klein, Michel, Omelayenko, & Borys, 2002), as well as (De Bruijn, Jos, Ehrig, Marc, Martin-Recuerda, Francisco, PoUeres, Axel, Predoiu, & Livia 2005). Unfortunately, the usage of the terms differs considerably.

This list consolidated common definitions:

Combining:

Two or more different ontologies are used for a task in which their mutual relation is relevant. The combining relation may be of any kind, not only identity. Therefore, no information on how the relation is established can be given at this point.

For instance, if we have two ontologies, having same semantics, we can combine them. From real life example, if we are travelling from place A to place B, having a distance of 100 KM, we can travel either in Car, Plane or Ship. This depends on the terrain. Therefore, the travelling will remain the same system ,while its first and the last node of departure and arrival will become different as per the object being used. Thus combining the different ontologies.

Integration:

For integration, one or more ontologies are reused for a new ontology. The original concepts are adopted unchanged, possibly, they are extended, but their origin stays clear, e.g., through the namespace. The ontologies are merely integrated rather than completely merged. This approach is especially interesting, if given ontologies differ in their domain. Through integration, the new ontology can cover a bigger domain in the end. Ontology alignment may be seen as a prestep for detecting where the involved ontologies overlap and can be connect with each other. The most prominent integration approaches are union and intersection (Wiederhold & Gio 1994), where either all entities of both ontologies are taken or only those which have correspondences in both ontologies.

For instance, if we have a domain of Information System, having its own Ontology, while we have an Ontology of Organization, we can Integrate two different domains of Organization Management and Information System Management, to come up with the broader domain of Information Management within Organizations.

Matching:

For matching, one tries to find two corresponding entities. These do not necessarily have to be the same. A correspondence can also be, e.g., in terms of a lock and the fitting key. A certain degree of similarity along some specific dimension is sufficient, e.g., the pattern of the lock/key. Whereas combining allows many different relations at the same time, matching implies one specific kind of relation. A typical scenario for matching is web service composition, where the output of one service has to match the corresponding input of the next service. Any schema matching or ontology matching

algorithm may be used to implement the Match operator. Matching corresponds to our definition of general alignment, however, where a fixed relation between the aligned entities expresses the kind of match.

For example, it's the matching that lets enable the other Ontology. That is, the only matching that being a valid input of another Ontology. From real-life scenario, we can have the model of RAM, which either could be of DDR, DDR2, DDR3 or SIMM/DIMM, that can be of matching as per the slot on the motherboard to have working Ontology in combination.

Mapping:

Ontology mapping is used for querying of different ontologies. An ontology mapping represents a function between ontologies. The original ontologies are not changed, but the additional mapping axioms describe how to express concepts, relations, or instances in terms of the second ontology. They are stored separately from the ontologies themselves. Often mappings can only be applied in one direction, e.g., the instances of a concept in ontology 1 may all be instances of a concept in ontology 2, but not vice versa. This is the case, if the mappings have only restricted expressiveness and the complete theoretical mapping relation cannot be found for the actual representation. A typical use case for mapping is a query in one ontology representation, which is then rewritten and handed on to another ontology. The answers are then mapped back again. Whereas alignment merely identifies the relation between ontologies, mappings focus on the representation and the execution of the relations for a certain task.

This terminology therefore doesn't depicts the Ontology, but actually how the two Ontologies are interconnected. For instance, Ontology of Selling the specific product in particular market is getting a useful information from the Ontology of Market Research Reports by third party publishers, e.g. Gartner. Thus, acting as a function to provide information for selling specific product.

Mediation :

Ontology mediation is the upper-level process of reconciling differences between heterogeneous ontologies in order to achieve interoperation between data sources annotated with and applications using these ontologies. This includes the discovery and specification of ontology alignments, as well as the actual usage of these alignments for certain tasks, such as mapping for

query rewriting and instance transformation. Furthermore, the term ontology mediation also subsumes merging of ontologies.

For instance, having an Ontology of Manufacturing of Automobiles and Architecting a House are heterogeneous Ontologies. However, there exists different classes that can be aligned and therefore can be reutilized in order to have interoperation between their used application. Like, Automobiles are designed for homes in mutual understanding with architecting the house, to have car appropriately.

Merging:

For merging, one new ontology is created from two or more ontologies. In this case, the new ontology will unify and replace the original ontologies. This often requires considerable adaptation and extension. Individual elements of the original ontologies are present within the new ontology, but cannot be traced back to their source. Alignment again is a prestep to detect the overlap of entities.

This therefore means to merge different Ontologies, thus having single Ontology. For instance, architecting a House is the ontology while developing a house is another ontology. However, they can be merged into one to come up with single ontology, building a house.

Transformation:

When transforming ontologies, their semantics is changed (possibly also changing the representation) to make them suitable for purposes other than the original one. This definition is so general that it is difficult to relate alignment to it.

For instance, transforming an ontology on writing books can be semantically changes when aligning to writing a novel or bibliography etc.

Translation:

We here define translation as an operation restricted to data translation (Popa, Lucian, Velegrakis, Yannis, Miller, Renee , Hernandez, Mauricio, Fagin, & Ronald, 2002), which may also include the syntax, e.g., translating an ontology from RDF(S) to OWL. The representation format of an ontology is changed while the semantics is preserved. As we are talking about semantic alignments, translation is an underlying requirement when the formats differ, but we do not address translation itself.

For instance, in a computer language, translating a program developed in JAVA to another programming language platform C#.NET to make it over .NET domain. The semantics remain the same while its compiler based instructions will be changes.

*Ontology alignment touches many related fields. Data integration but also ontology integration are not new topics. To ensure the reuse of existing ideas an extensive investigation on related work is necessary before an own innovative approach is developed. This chapter starts with theoretic considerations on alignment. It will then focus on other actual approaches for ontology alignment. We will reach out to related fields, in specific the integration efforts of the database community, which has been approaching this issue for many years. A good overview of related work is also given in (*De Bruijn, Jos, Ehrig, Marc, Feier, Cristina, Martm-Recuerda, Francisco, Scharffe, Frangois, Weiten, & Moritz 2006)..

3.4 Theory of Alignment

A very comprehensive survey on existing work for ontology alignment is given in (Kalfoglou, Yannis Schorlemmer, & Marco 2003). As the authors have a strong theoretical background, we here refer to their discussions. They classify the theoretical frameworks into three groups: algebraic approaches, information-flowbased approaches, and translation frameworks. These frameworks represent the mathematical foundations for any alignment approach.

3.4.1 Algebraic Approach

the authors (Bench-Capon, Trevor , Malcolm, & Grant 1999) extend the concept of abstract data type to formalize ontologies and their relations building upon a universal algebra. Besides the definitions of ontology, signature, and data domains, they more importantly define the notion of morphisms in this context. Morphisms are structure-preserving transformations and are the core of the algebraic approach. In this framework aligning two ontologies O1 and O2 means to search for a pair of ontology morphisms P1 and P2 from an unknown ontology Ox to the two ontologies, which need to be aligned as depicted in **Figure 3.2.** For the next step of ontology merging one can then rely on the construction of categorical pushouts (Hitzler, Pascal, Krotzsch, Markus, Ehrig, Marc, Sure, & York 2005), based on these morphisms.

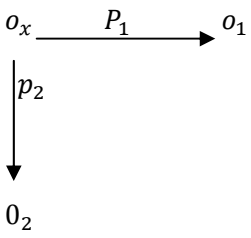


Figure 3.2: Morphisms on Ontologies

(Jannink, Jan, Pichai, Srinivasan, Verheijen, Danladi, Wiederhold, & Gio, 1998) propose a different algebraic approach. Their algebra is based on category-theoretic constructions, through which the contexts from knowledge sources are extracted and combined. See also (Mitra, Prasenjit, Wiederhold, & Gio, 2002) for the discussion of such an algebra. The approach is less formal than the previous one. Categories are defined as the union of concept specifications and their instances. Again, pushouts are applied to these categories. "Morphisms allow translation from one specification to another when there is no semantic mismatch. Therefore, they are applicable when intensions and extension are not distinguishable, such as in mathematical structure." In the end, their approach is used to identify semantic mismatches between the intensions and extensions of concepts, or alternatively, if no mismatch occurs, to align the elements.

3.4.2 Information-Flow-based Approach

(Kent & Robert, 2000) demonstrates how ontology sharing is formalizable within the conceptual knowledge model of Information Flow (IF) (Barwise, Jon , Seligman, & Jerry (1997).(Kent, 2000) assumes that there exists a common generic ontology specified as a logical theory. This common generic ontology consists of the terminology and semantics shared by diverse communities. Participating communities extend this generic ontology in their own ontologies. Specification links connect the community ontologies with the generic ontology. The yet open task is to align the community ontologies. The author proposes two steps for this. In a first lifting step from theories to logics, instances (the semantic elements) are added. The author assumes that there is a natural set of connections between instances. An instance of one community is connected to an instance of another community, when they agree on common inherited types (in our terminology: the concepts, relations, and axioms). Based on these instances it is possible to align the concepts. The second step, the fusion step, then creates a core ontology of community connections (virtual logic). "Instance connections and identifier types comprise a natural quotient construction on the participating community ontologies. The virtual ontology of community connections is computable as this quotient." The fusion of participating ontologies is done by creating a virtual ontology of community connections, i.e.. the linkage of instances. Through the new ontology, the logics of the involved community ontologies are therefore now interlinked.

(Kalfoglou, Schorlemmer, & Marco 2002) proposed the IF-Map methodology. They extended the approach of (Beghtol, Howarth, & Williamson 2000)., but also use algebraic considerations. In their

approach, they have community ontologies linked to a reference ontology. They then look for logic infomorphisms from which they derive the alignments.

3.4.3 Translation Framework

A translation framework has been provided by (Ciocoiu, Mihai, Nau, Dana , Gruninger, & Michael 2001), a logic based approach. Ontologies can be translated, if it is possible to express all statements of one ontology in the other ontology. The statements have to be equivalent with respect to their foundational theories - for this, the foundational theories have to be the same. Besides this strong translation, the authors also propose a partial translation where only sub-ontologies have to be completely translated or the ontologies can be extended with additional definitions. Finally, they introduce weak translations, where the theory only has to be interpreted in the other one. To actually set up this framework the authors propose an interchange ontology library. In this interchange library, all participating ontologies are modeled soundly and completely. From these axiomatizations it will then be easy to determine and construct the translation between any of the involved ontologies.

All the presented theories deliver valuable input of ideas on how to set up an ontology alignment approach. However, they cannot be applied directly, as they require complete intensional or extensional modeling of the involved ontologies. As one of the goals of this work is to be robust against possible missing or wrong information, we will extend the theories with considerations on similarity rather than logical equivalence. This will lead to new approaches for ontology alignment.

3.5 Existing Alignment Approaches

This section will show what alignment approaches already exist and what their specific characteristics are. This gives us a solid grounding for developing an own approach fulfilling the requirements we had identified.However, going forward with the details, we should know what actually the issues pertaining to existing alignment approaches actually are. For the reason, we also have mentioned the issues associated with the approach in order to have reader fully understand before applying the same technique.

3.5.1 Classification Guidelines for Alignment Approaches

Before describing the approaches, we will create classification guidelines for alignment approaches. This makes it easier to identify the methods that are specific for each individual approach. We will now explain the four main dimensions: input schema, alignment process, output schema, and use case. The

dimensions are inspired by (Rahm, Erhard, Bernstein, & Philip 2001; Shvaiko, Pavel Euzenat,& Jerome. 2005).

Input schemas:

Generally, this work focuses on aligning whole ontologies. However, for related work we look beyond ontologies. There are many input schemas, which need to be aligned.

• Apart from ontologies, there are structures with less semantic structure such as directed acyclic graphs, trees, or classification schemas. For integration, database schemas may also be the input. Specific schemas such as XML Schema or Entity-Relationship-Models (ERM) have other special semantics

• The syntactic representations, which various approaches handle, are another dimension of input, though XML makes a grounding.

• In general one also distinguishes between schemas with instances and without instances.

Process:

The main dimension is the process of aligning. In fact, the processes of ontology alignment approaches differ most, as there exist a large number of parameters and methods.

• The process may be based on a strict logic and infer alignments. Or it may rely on heuristics and collect evidence for alignments.

• Some approaches focus on the syntax whereas others include the whole semantics of the model. In addition, the degree of incorporated semantics differs. Are only linguistic properties taken into account? Does the approach rely on explicitly modeled constraints?

• The approaches may focus on the individuals (instances) or the structure (schema).

• In most cases one has more than one type of evidence pointing towards alignment. The question remains how they are combined, with automatic aggregation being one possibility and manual checking being another one.Automatic aggregation may be based on rules or learned classifiers, as presented in [64]. Multiple rules or learned classifiers may be also be combined.

• Additional external sources help to improve results. Some approaches make extensive use of this, e.g., by consulting dictionaries (such as Word- Net) or relying on additional instance bases.

• Finally, alignment approaches vary in the degree of automation, ranging from fully manual through automatic recommendations to fully automatic.

Output:

Likewise the output differs considerably for different approaches.

• Depending on the input schema, different elements are aligned: concepts, relations, and instances; nodes and vertices; classes, objects, and attributes; individual elements or whole structures.

• The simplest cases are one-to-one alignments. In real world, one will often encounter n-to-m alignments instead. The question what kind of relation is finally identified has to be answered as well: Equivalence of entities is most common, but subsumption, overlap, or complete orthogonality are other options. Most alignments are simple, but first steps towards complex alignment are made.

• In most of today's algorithms, alignments are not only provided per se any longer. A confidence value for the alignments is provided by most approaches. Often one can refer to confidence as probability, making it possible to reason with the inexact information.

Use Case:

In many surveys on ontology alignment, the use cases are not considered a dimension for distinction. Even though they might not directly be reflected in input, process, or output, they definitely influence the complete setting of the alignment process.

• Typical use cases are data integration, ontology (or schema) merging, and creation of mappings for translation between different schemas.

In fact, the use cases are the reason why different approaches cannot be directly compared.

3.6 Ontology Alignment Approaches

The subsequent concrete existing ontology alignment approaches will be described along the dimensions just introduced. A summarizing table will be presented in the end.

3.6.1 IF-Map

Another system inspired by formal concept analysis is IF-Map (Kalfoglou & Schorlemmer, 2003). It is an automatic method for ontology mapping based on the Barwise-Seligman theory of information flow

(Barwise & Seligman, 1997). The basic principle of IF-map is to align two local ontologies by looking at how these are mapped from a common reference ontology. It is assumed that such reference ontology is not populated with instances, while local ontologies usually are. IF-Map generates possible mappings between an unpopulated reference ontology and a populated local ontology by taking into account how local communities classify instances with respect to their local ontologies

The problem with using this approach to align our ontology is it assumes reference ontology is not incorporated with any of the instances. However, in majority of the cases, instances does exists. Therefore voiding this approach for majority of the Ontologies.

3.6.2 Glue

Glue (Doan & Domingos, 2002) Is a semi-automatic system that employs machine-learning techniques to find mappings. It uses multiple learning strategies to cope with different types of information, either in data instances or in the taxonomic structure of the ontologies, in order to make predictions. It consists of three main modules: Distribution Estimator, Similarity Estimator, and Relaxation Labeler. The Distribution Estimator takes as input two taxonomies O1 and O2, together with their data instances. Then it applies machine learning techniques to compute for every pair of concepts their joint probability dis tributions. The Distribution Estimator uses a set of base learners and a meta-learner. Next, GLUE feeds the above numbers into the Similarity Estimator, which applies a user-supplied similarity function to compute a similarity value for each pair of concepts. The output from this module is a similarity matrix between the concepts in the two taxonomies. The Relaxation Labeler module then takes the similarity matrix, together with domain-specific constraints and heuristic knowledge, and searches for the mapping configuration that best satisfies the domain constraints and the common knowledge, taking into account the observed similarities. This mapping configuration is the output of GLUE.

The problem with using this approach to align our ontology is to have utilization of heuristic knowledge. For the specific reason, as it operates on machine learning approaches, it should first be trained well to actually apply prior knowledge in order to perform heuristic. Therefore extensive training to let the agent be knowledgeable is to be applied before actually utilizing the approach.

3.6.3 ONION

In their tool ONION (ONtology compositlON system), (Mitra, Prasenjit, Wiederhold, Gio, Kersten, & Martin., 2000) provide an approach for resolving heterogeneity between different ontologies. Their

basic assumption is that merging of whole ontologies is too costly and inefficient. Therefore, they focus on creating so called articulation rules, which link corresponding concepts. As manual creation of these rules is not very efficient either, they use a semi-automatic approach, which takes into account heuristics on several simple relations, such as labels, subsumption hierarchies and attribute values. Dictionary information is also used for the alignment process. From these relations a match is presented to the user who then has to decide whether the alignment is valid or not. The articulation rules linking can be applied when an application inquires information from two ontologies. The work is based on their theory on composition algebras (Mitra, Prasenjit ,Wiederhold, & Gio,2001) .

As it uses the dictionary information as well as applies heuristic techniques to find the relationship between different classes or its instances to actually make the ontology aligned, it first proposes the matching results to user. The user is then to decide either to adopt it or not. thus a semi-automatic approach, comparatively effective as it proposes to user to etiher accept or reject the alignment, while its not much effective in the way that they match the chunks of systems rather evaluating the entire system and let it be filtered for specific domain.

3.6.4 PROMPT

PROMPT (Noy, Natasha , Musen, & Mark 2000) is a tool that provides a semi-automatic approach to ontology merging. It is based on the SMART algorithm. After having identified alignments by matching of labels, the user is prompted to mark the entity pairs that should actually be merged. During merging, PROMPT presents possible inconsistencies such as name conflicts or relations not pointing anywhere any longer. The user then decides on how to react and resolve the issues manually.

In this approach, user has to apply the markers to each entity, thus having a time consuming activity before actually merging the ontologies. SMART algorithm works as follows:

Setup: load files, set preferences, ...
Execute operation: perform automatic updates, detect conflicts, create suggestions
Select operation: choose from suggestion list, create a new operation,
Initial suggestions: identical names, synonyms,
Superclasses for top-level classes in alignment

It can therefore be seen that the algorithm cannot automatically be able to align the entire Ontology, rather user has to be involved extensively in putting up the markers.

3.6.5 Anchor-PROMPT

(Noy, Natasha, Musen, & Mark 2001) represented an advanced version of PROMPT that includes similarity measures based on ontology structures. So-called anchor points, alignment pairs in the ontologies, are identiiied first, normally through string-based comparisons of the entities or by directly having them assigned by the user. Based on these known anchor points the structures of the ontologies are traversed resulting in propositions of additional alignments of entities between the known anchor points. Specifically, paths are traversed along hierarchies as well as along other relations. Afterwards the results are again presented to the user, including an explanation, and the user decides whether to merge the proposed entities or not. This process is continued in several iterations.

It is relatively better than the PROMPT, however, it uses the structural approach. The problem in utilizing the structural approach is there exists different classes or instances that can have same structure but have totally different semantics. For instance, TV and LCD have same semantics of letting you watch your programs, while structurally they are different. Similarly, having a structure of Human Body and Computer has same structure, however semantically they are different. This approach therefore gets ineffective.

3.6.6 Prompt-Diff

(Noy, Natasha, Musen, & Mark 2002) produced a tool to compare different ontology versions. Different heuristic matchers are used to determine the similar entities. The matchers are combined in a fixed-point manner until no further changes occur. PromptDiff makes use of the fact that two versions of one ontology have considerable overlap.

The drawback to utilize is the Ontology may not have entities well defined. Thus, similar entities may or may not be found. If no similar entity exists, this algorithm fails to actually deliver the aligned ontology.

3.6.7 Chimaera

Chimaera (Ganter, Bernhard , Wille, & Rudolph 1999), an interactive tool for ontology merging. Its basic ontology format is OKBC, but it can also handle other languages. After executing a linguistic matcher, Chimaera uses the results for triggering the merging operation. During this process, the human has to decide whether to merge or not. Chimaera also provides proposals on reorganizing the taxonomy once a merge has been processed. Overall, Chimaera allows diagnosing and manual editing for ontology merging. The actual alignment of entities however is based on simple measures.

Although this tool is impressive, but for the basic alignment only. The issue with it lies that it is not even a semi-automatic alignment tool. It only provides platforms to syntactically check the entities and if find any, proposes the solution to user. The user is actually the person that will align after approving the proposed solution to merge the same structural entities, while the person has to do himself/herself the complete alignment by going through the entire ontology.

3.6.8 FCA-Merge

The FCA-Merge approach has been presented by (Stumme, Gerd , Maedche, & Alexander 2001). As the name already suggests, its goal is to merge ontologies. It is based on formal concept analysis as described in (Ganter, Bernhard , Wille, & Rudolph 1999). Given two ontologies in a first step FCA-Merge populates them with instances that are extracted from a set of documents. This step is necessary, as most ontologies do not have sufficient instances, but these are required for formal concept analysis. Based on these instances the ontologies are represented as concept lattices, i.e., concepts are seen as sets of instances. At this point lexical information is used to retrieve domain specific information. Using formal concept analysis the two contexts are integrated and a new lattice is created thereof. Pruning steps are applied to keep the size of the lattices small. In a last step, the lattice has to be transformed back into an ontology. This step has to be done manually. To solve conflicts, such as duplicates, FCA-Merge has an automatic support to guide the user through the process. One should mention that FCA-Merge deals only with concept hierarchies and underlying instances - alignment of relations is not supported.

The drawback to utilize is it relies on lexical information of specific domain. It only merges the ontologies rather actually aligning them. Set of Documents is therefore should be well documented with all the semantics of the entities so that the approach can find the relevant alignment entity from the set of document of the source Ontology. As being heavily rely on pre-documented material, it therefore makes it ineffective.

3.6.9 HCONE merge

is an approach to domain Ontology matching and merging exploiting different levels of interaction with a user (Kotis, Vouros,Konstantinos, & Stergiou. 2006; Vouros & Kotis, 2005; Kotis & Vouros 2004). First, an alignment between the two input ontologies is computed with the help of WordNet. Then, the alignment is processed straightforwardly by using some merging rules, e.g., renaming, into the new merged ontology. The HCONE basic matching algorithm works in six steps

1. Chose a concept from first ontology

2. Retrieve the WordNet senses of this concept;

3. Retrieve hypernyms and hyponyms of these senses.

4. Exaluate for the most frequent terms within the vicinity (senses, hponyms and herperonyms) their frequency of occurence.

5. Build a query from the related concepts related to the initial concept in the input ontology.

6. Use Latent Semantic Indexing for determining the best sense to be used in the query

This method is way better than all the previous. Though it's a time consuming as well as requires extensive machine resources, but it actually check the concepts of the entity from its WordNet. Senses are then mapped and checked with senses in our Ontology and thus evaluation is performed, resulting in way better aligned Ontology. The issue remains there that WordNet should have an extensive repository to actually reconcile the concept of the entity in order to merge the ontologies.

3.6.10 S-Match

a generic semantic matching tool. It takes two tree-like structures and produces a set of mappings between their nodes. S-Match implements semantic matching algorithm in 4 steps. On the first step the labels of nodes are linguistically preprocessed and their meanings are obtained from the Oracle (in the current version WordNet 2.0 is used as an Oracle). On the second step the meaning of the nodes is refined with respect to the tree structure. On the third step the semantic relations between the labels at nodes and their meanings are computed by the library of element level semantic matchers. On the fourth step the matching results are produced by reduction of the node matching problem into propositional validity problem, which is efficiently solved by SAT solver or ad hoc algorithm (Giunchiglia, Shvaiko, & Yatskevich, 2004; Giunchiglia, Yatskevich, & Giunchiglia. 2005).

As it works on the method of testing entities on the basis of its requirements by comparing multiple entities having similar structure and therefore adopting the semantics from already aligned ontology to assign the same concept to this method, it makes relatively better aligned ontology. This has the same problem as earlier discussed, which is to have extensive WordNet to actually check the semantics through traversing its structural properties with the one, maintained in repository

APPROACH	***INTEROPERABILITY LANGUAGES***	***ONTOLOGY STRUCTURE***	***STRATEGY***	***ADDITIONAL RESOURCES***	***LEVEL OF AUTOMATION***
IF-Map	RDF,KIF,Ontologua, Protégé -KB, Prolog	Concept	Linguistic, Heuristic, Reasoning	Reference Ontology	Automatic
GLUE	-	Concept, Properties, Instance	Probability	-	Automatic
ONION	IDL, XML-Based	Concept, Properties	Linguistic	Word Net	Semi-Automatic
PROMPT	OWL, RDFS	Concept, Properties, Instance	Linguistic, Heuristic	-	Semi-Automatic
Chimaera	Ontologua		Linguistic, Heuristic	-	Semi-Automatic
FCA-Merge	-	Concept, Instance	Linguistic, Heuristic	-	Semi-Automatic
H-CONE	OWL-DL	Concept, Properties,	Linguistic, Reasoning	Word Net	Semi-Automatic
S-Match	-	Concept	Linguistic, Reasoning	Word Net	Automatic

Table 1.2: Characteristics of studied ontology mapping and merging systems

Approach	***Automation***	***String Based***	***Language Based***	***Linguistic Resources***	***Graph Based***	***Taxonomy Based***	***Repository***	***Model Based***
IF-Map	Full	Heuristic				Heuristic		Semantic Reasoning
GLUE	Semi					Probabilistic Reasoning		
ONION	Full	Heuristic				Probabilistic Reasoning		
PROMPT	Semi	Heuristic						
Chimaera	Semi	Heuristic						
FCA-Merge	Full		Heuristic	Heuristic				
H-CONE	Semi						Repository	
S-Match	Full	Heuristic	Heuristic	Heuristic		Heuristic		Semantic Reasoning

Table 3.3: Summary of Mediation Systems

3.7 Summary of Tables

The obersvervation been made that automated includes IF-MAP, ONION, FCA Merge and S-Match while other remains Semi-Automatic. Although we have seen that automatic cannot completely aligned the Ontology, as discussed earlier due to the limited specified in each approach. However, in Semi-automatic, there is a reliance over end-user that gets suggestions from the software, while proposing an idea to end-user either the person wants to pursue with the proposed solution or not. Other than FCA-Merge and H-Cone works over the Heuristic String based searches. This means, all other required well-trained intelligent Agents (IAs) that are well updated of the string based matching for alignment of entire ontology. However, it can be seem that FCA-Merge is heuristic in terms of language based searches. This is more appropriate to use the dictionary based approach to align the already maintained repository. While, language based, it also applicable for linguistic based which means to have multiple semantics of single class thus aligning heuristically as per the context of the Ontology.Taxonomy based approaches include IF-MAP and S-Match which actually needs IAs to get trained to taxonomically align the Ontology, while Probabilistic approaches, working on the chances, includes GLUE and ONION. It also depends on the past records to come up with the percentage chance of occurrence to near-to-accurate auto-alignment of Ontology.Similarly, both aforementioned also are Model-based semantically reasoning approaches to align the Ontology. Semantic alignment through reasoning is way better than any heuristic approach as the Ontology can better be aligned as per their context. Combining both the techniques nearly aligned Ontology to perfect. With the above tables , S-Match approach has almost all the attributes that actually dependent to accurately align the Ontology. Thus it can be recommend to the readers to utilize the same in their Ontology and come up with the better results than other mentioned approaches.

3.8 Summary

This chapter described ontology alignment plays an important role in solving interoperability in heterogeneous systems and in many application domains and the mismatching that could appear when trying to integrate, map, align or match two ontologies. It also describes many matching strategies and operations. Finally, it mentions some tools and systems that have been developed in these areas.

Chapter 4 Case Studies

4.1 Introduction

This chapter will analyse 2 case studies a business hierarchy and engineering hierarchy and give detailed explanation of the both designed structures .

Starting by an analyzation of the different aspects of the organization .Then move forward for in-depth analysis of each function, also discuss the role of Information to actually going towards automation of the organization, performance monitoring of business processes through KPI dashboards and effective decision making through Business Intelligence systems.

First need to have a target for which industry we are focusing on .Then move to two main parts i.e. what would be the internal and external factors or forces that may influence the business. Most importantly, internal factors can be managed easily in comparison to external factors however external factors are a bit difficult to manage, being involvement of out-of-control stakeholders. It will described it in detail of what these actually mean.

4.1.1 Industry-Knowledge

Industry-Knowledge is prerequisite to design any structure of an organization. For business needs, we have to cover different aspects. We have different functions which include Finance, Legal, Resources, Maintenance, and Production etc. However, it is not a good idea to have a flat hierarchy which is difficult to manage for any business. It should be well structured. Though the communication will get slow, but due to utilization of recent technologies, this is not an issue. The other drawback is there is a high-dependency on Jack-of-All people, being flat hierarchy, while in structural, there are specialized people for each function.

With the aforementioned idea of having a structural organization, we can group the similar functions having same attributes and segregate them as **Internal** and **External**, being top level node. Then further divide the concept to main category, include Financial, Ethics, Legal and Infrastructure.

Broadly, defined organizations as follows:

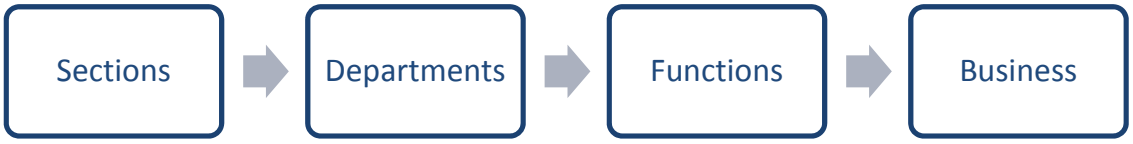


Figure 3: Organization Structure

The following is the proposed structure that an organization should be adopting. In coming sections they are discussed in detail.

4.1.2 Business Hierarchy Structure

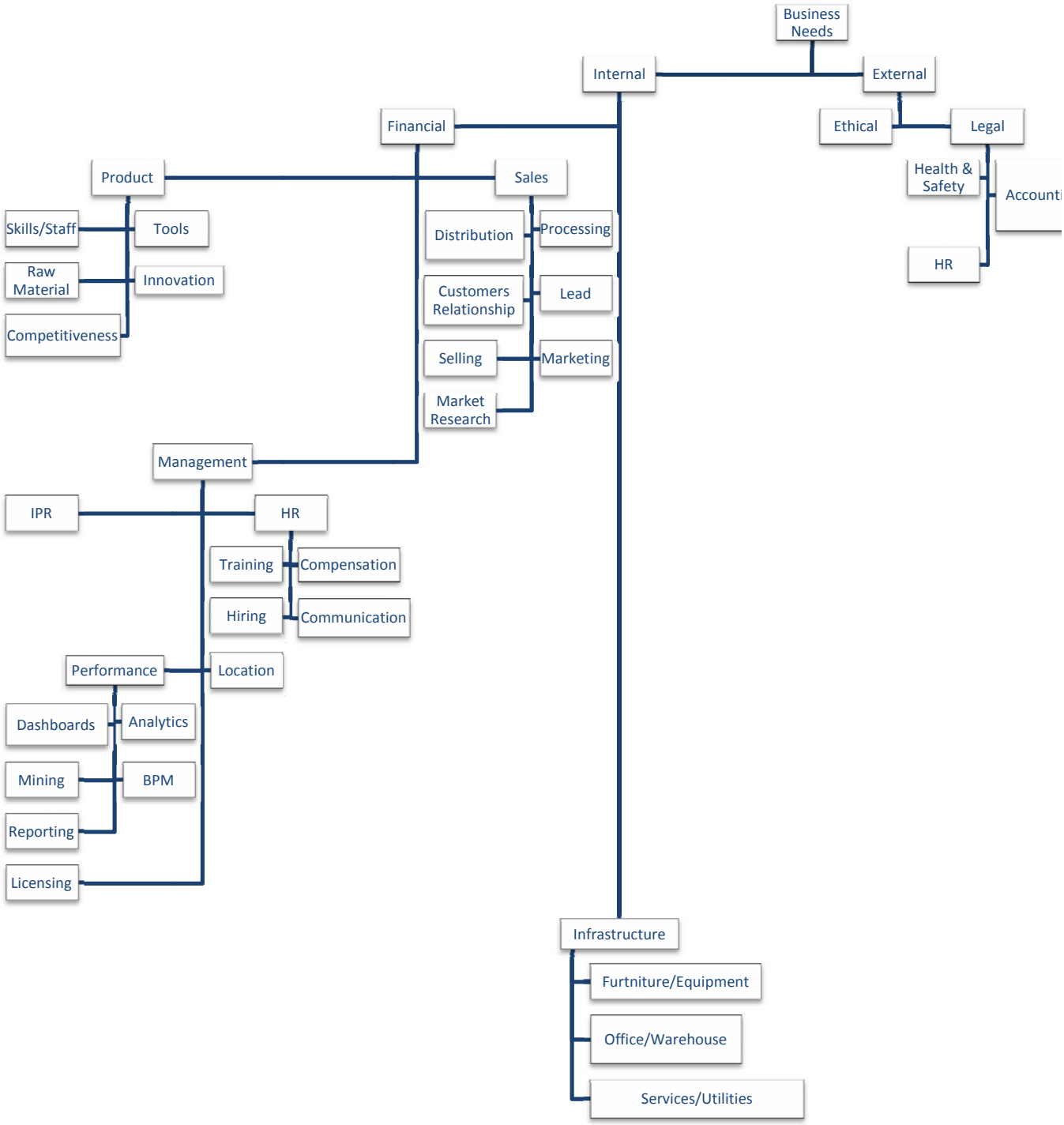


Figure 4.2: Detailed Organization Structure

4.1.3 Internal and External forces

By internal, we mean to group all the functions that the organization is required to run successfully. This includes financials of the company, procurements, resources, supply chain management, production etc. These are common in terms that they all have employees or resources that can handle the situations, occurs at different stages. Also, they would be recurring payment activities, i.e. payroll must be run monthly or after every specific period of time as per terms and conditions. They all will be controlled by the management. Thus management is also a function to manage and control all mentioned functions.

By external, we mean the laws, ethics, welfare work etc. that are influenced from outside. This influence can be imposed either by the country rules and regulations, local laws, transportation laws, global laws, change in market etc. These cannot be controlled and thus because of external forces, we have to control internal resources to coup up with the environmental changes.

4.1.4 Functions

Following are the main functions, which are actually grouped of the similar activities

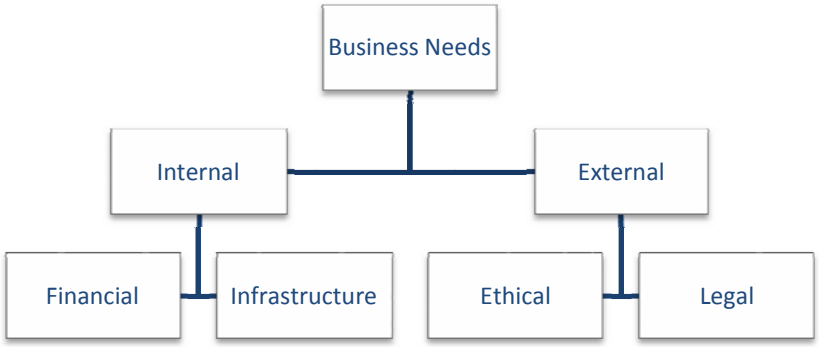


Figure 4.3: Function

Legal

These include all the functions related to follow industry-specific, country-specific, and WTO-specific rules, so no obligation can be imposed that may either liquidate the organization, but also could have bad impact over its customers and suppliers, and could be banned by external uncontrollable forces. It includes the following:

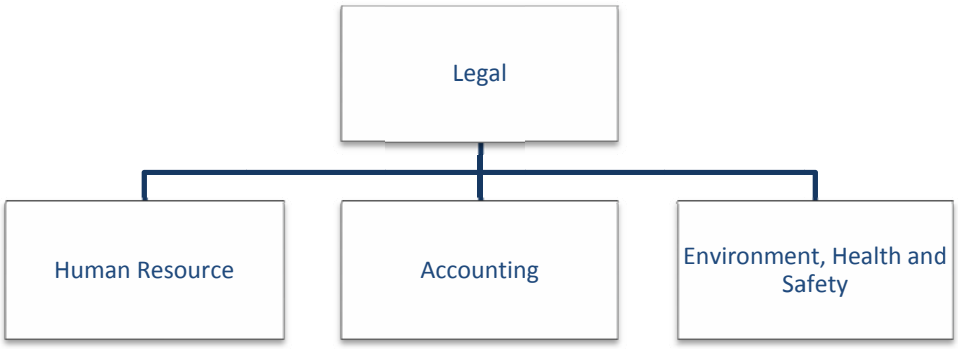


Figure 4.4: Legal

Human Resource

This is related to have relationship with the ministries and other government bodies to relocate people, if sourcing resources from international-pool of talent. This also include following policy to hire, provide compensation and other benefits as per the imposed government rules of the country, being operated in particular area.

Accounting

This include following the rules and regulation of account procedures, so if the organization is listed in financial market, they should expose their financial statements and related profit within financial year, to attract more investors

Environment, Health & Safety

This is about not harming the environment through e.g. wastes of the organization, while equally taking care of health and safety of their staff, within and outside organization by obeying internal rules of EHS or publically published rules through external bodies.

Ethical

These include all the ethical work, following ethical practices to have better management within the resources of organization. This should be influenced as they may portray a better impact to customers. This may include, educating people in remote areas, having medical and other assisting camps for below-average salaried people or if not in a welfare state, or government is inactive in providing benefits to low-level class, and helping poor countries through donation etc.

Infrastructure

These includes the maintenance of existing infrastructure as well as to develop with innovative and new infrastructure for more productivity, to accommodate new resource, while equally making comfort zones or entertainments to re-energize the resources for better outcome, as being done by Google and other competitive organizations.

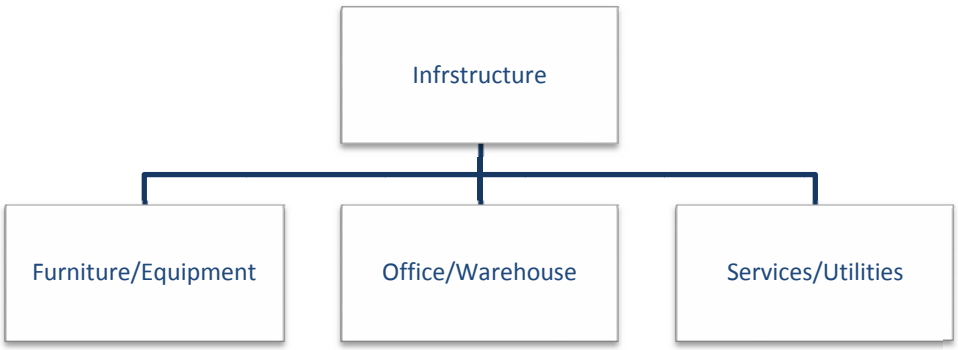


Figure 4.5: Infrastructure Structure

Financials

These include all the activities that involved finances. As everything is dependent on financials, the more you have, the better you are in a position to invest in new technologies. All internal activities are dependent on it. Following are the main functions related to the finance that can be controlled well through financial means.

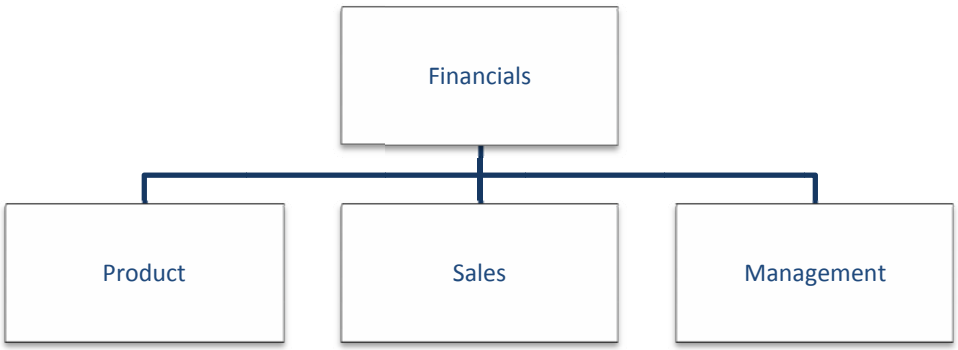


Figure 4: Finance Division

Product

This includes developing the finished product or providing services, if in a service based industry. It should be well researched of what the product cost would be and at what price will it be sold.

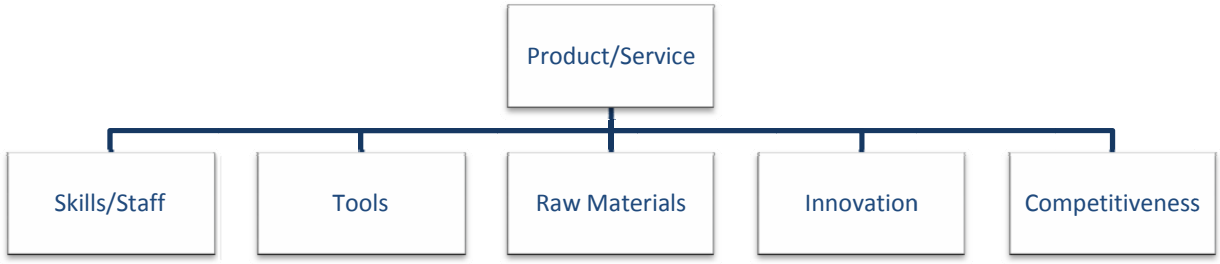


Figure 4.7: Product/Service Structure

Skills/Staff

Product includes skilled staff to produce quality product, while supervising automated systems that may malfunction, and needs to be diagnose to repair to limit breakdown period.

Tools

This include related systems that should be available in order to develop the product, or providing services

Raw Material

This is the main material, which after being processed for value-addition would be made available for its customers.

Innovation

This includes continuous investments on new and improved product or services to let footprint in the market to satisfy their customer, while equally penetrating in the new market to cater customer needs and satisfying the same

Competitiveness

This includes being in a process to present in the market and provide equal or better services to their customers by keeping an eye on the competitors' moves.

Sales

This includes the function to conduct research about what pricing model should be adopted, at what price should the product/service be sold, which new financial benefits can be provided to customers and

making it possible for products to reach the market to make it available at almost everywhere, where being marketed

Figure 5: Sales Structure

Distribution

This involves making distribution channel to get orders from markets and providing the same in a short period.

Processing

This includes making the product available almost everywhere after making the product in presentable form. This involves research of where the product should be placed so specific customer-market can have reach.

Customers Relationship

This involves having feedback of the customers, while making them answered to every query, to feel them the ownership over the products and could rely on the products' after sale service.

Lead

This includes having a research on market norms, understanding the needs of the market and provide equally suited product after providing research and processed information to the production OR innovation departments to come up with the market requirements.

Selling

It involves giving opportunities and offers to run their own business through, say direct selling, by means of franchises. The initial financials is however can be provided by the organization, though not mandatory.

Marketing

This involves penetrating the idea in the market through either advertisement of the product, distributing pamphlets, providing schemes to vendors, direct and cross selling,

Market Research

This involves conducting a research on either the product outcome, doing trend analysis for future production to ensure product availability as per consumption, or researching about the needs of the customers, like availability of Smartphone application for consumer to get consumer-interests-specific updates.

Management

This involves overall manages activities to overlook the organization performance, inducted resources, and consequently taking steps or decisions while revising mission and vision of the organization, as well as taking steps towards mergers, acquisitions or liquidate.

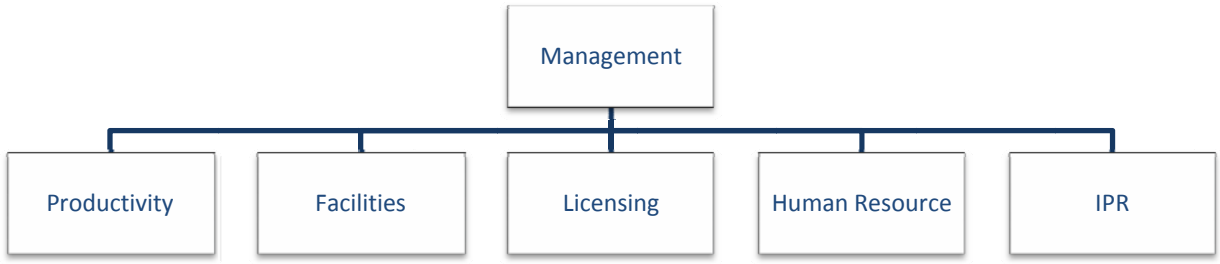


Figure 6: Management Structure

IPR

This includes either intellectual property rights being followed in the organization. This is to enforce policy in the organization while equally monitoring the same.

HR

This includes management of the resources in the following ways

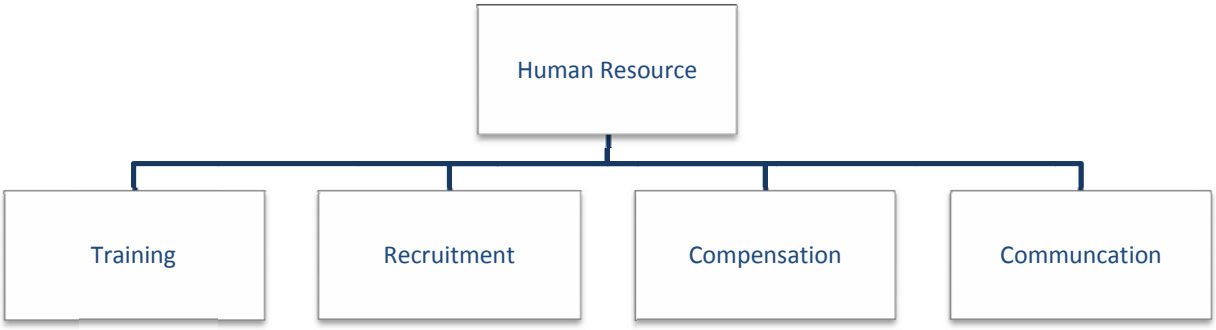


Figure 7: Human Resource Structure

Training

Provide local, in-house and foreign training to the employees of an organization. So they can be more productive by utilizing latest techniques within their areas.

Compensation

This includes providing market competitive compensation for better employee's retention policy while inducting the best resources from the market through attractive compensations.

Hiring

This includes the entire hiring processing of an employee, by publishing specification based vacancy and hiring competent resource as per the requirements, etc.

Communication

This includes effective communication between and within departments for optimized output. If there is any incorrect communication being done, the chances are to either being delayed in fulfilling the requirement, or loss of funds or collapse of the function.

Performance

This includes proper performance monitoring by utilization of following techniques

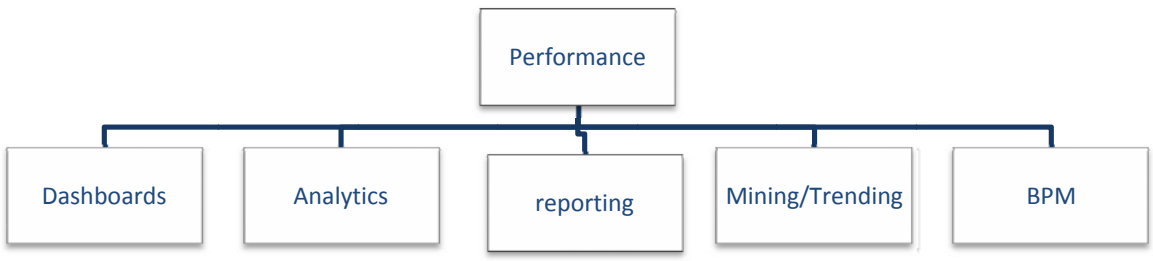


Figure 4.11: Performance

Dashboards

Develop KPIs for business processes to identify the bottlenecks in the process and coming up with the solution to address the same

Analytics

This includes coming up with the statistics of companies' positions in numbers. Thus, which area is potential, which needs to be redefined etc. can be analyzed through analytics

Mining

This includes taking decisions by conducing trend analysis, e.g. customer trends of purchasing and external factors that lead them to go for their decisions.

BPM

This includes continuous optimization of business processes to make the processes for effective by utilizing recent related technologies.

Reporting

This includes summarizing the financial figures for internal use and audit purposes of different dimensions.

Location

This includes management of acquiring locations while enforcing policies and other related laws at the sites to streamline entire business process through-out organization.

Licensing

This includes evaluating software licenses while coming up with the decision to either acquire particular software by conducting cost-benefit analysis of the organization which includes recurring maintenance payments of standard 22% around the world, which is definitely costing.

4.2 Engineering Management Organization

4.2.1 Engineering Information Management

To start with engineering an Information Management functions in an organization, it should be clearly defined of what the information should be circulated, maintained decision-making nature. How fast it should be reached to its destination, its authenticity, reachability and its maintenance for future references and audit purpose.

The second comes the integration of different system, how data will flow, i.e. implementations of workflows, definition of business processes and its adaption from industry's best practice. That also includes the selection of most suitable enterprise resource planning (ERP) systems, which may end-up in automating and integrating business processes of the entire organization.

To achieve automation and subsequently maintaining repository of internal information, different requirements of an organization must have been evaluated. For that, we need to go in depth in each of its functions. This is required to better manage their IT infrastructure, at reduced recurring and maintenance cost, as well as aware of what their functions are, for what are they responsible, and what services will be provided to users. The Service Level Agreement may then be incorporated to automate the workflow process, etc.

The next to evaluate is the infrastructure, that how the message will be communicated to concerns, within and outside of the organization. We will discuss all of them in more details in following sections.

4.2.2 Engineering Information Management Structure

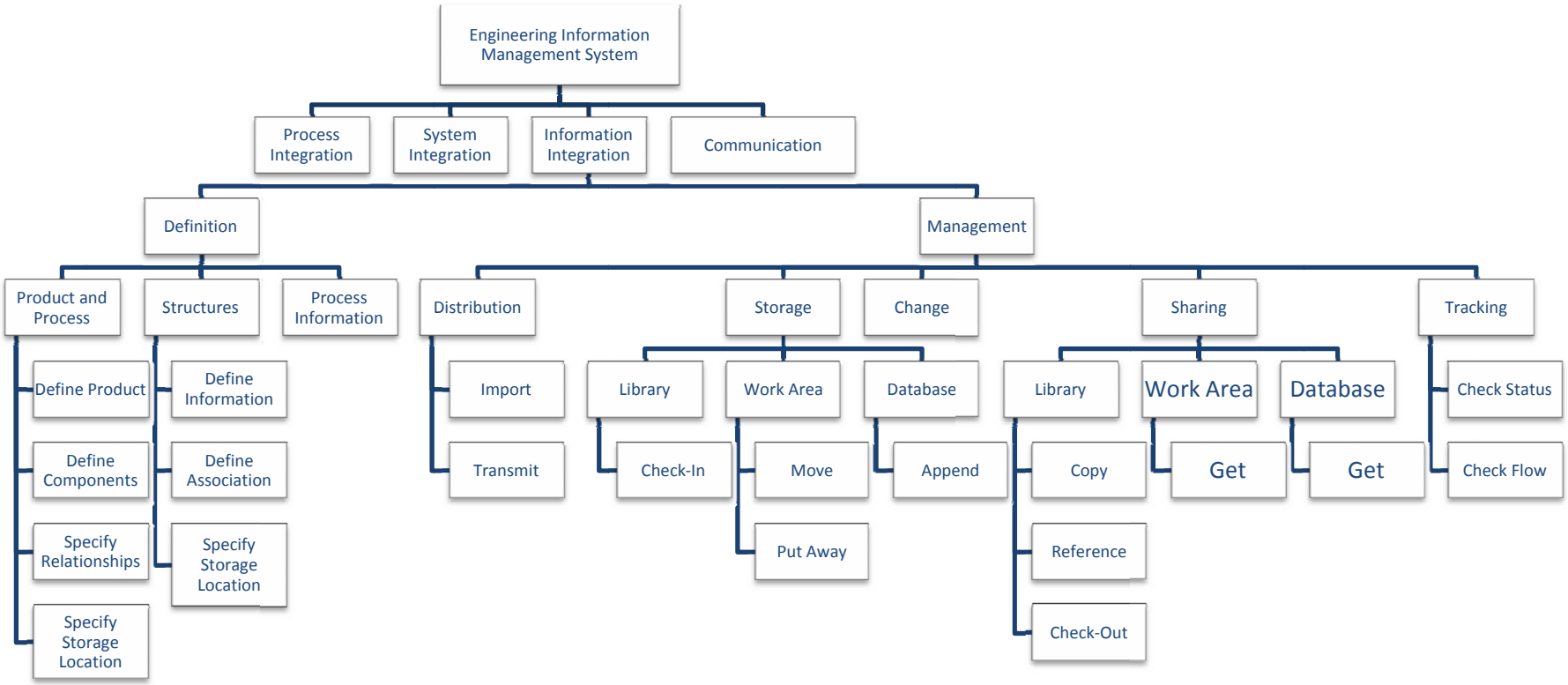


Figure 4.12: Detailed Information Management System for Organization

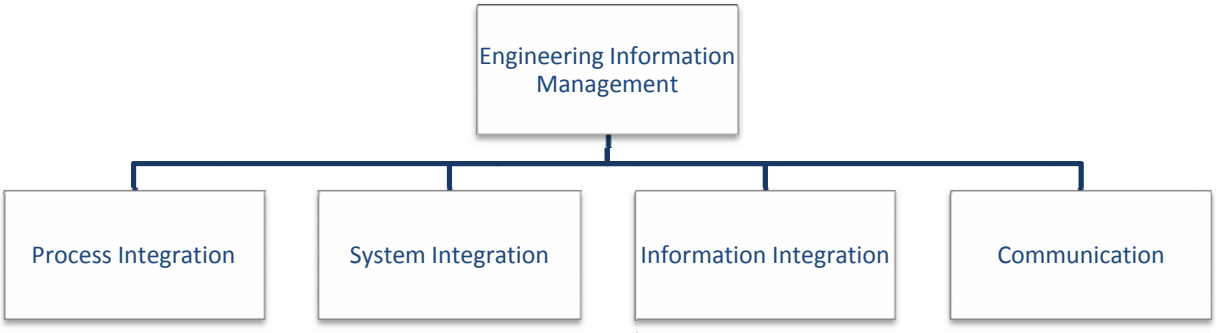


Figure 4.13: Engineering Information Management

4.2.3 Process Integration

By process integration, we mean the interconnectivity within the business processes, exists in the organization. The more they are integrated, the better the chances of miscommunication, the better the route between different actions, and the better the time will be utilized.

4.2.4 System Integration

This defines the interconnected systems, deployed in the organization. If the systems are interconnected, the less the chances are the information gaps, data redundancy, data duplication, and data inconsistency. The generated data can therefore directly flow to the next level.

4.2.5 Information Integration

This addresses the connectivity on the dependent processed data. This includes, e.g. policy being imposed on high-level decision, escalated down to systems to deploy policy and therefore automatically being applied as being tightly integrated. It involves different steps, defined as follows:

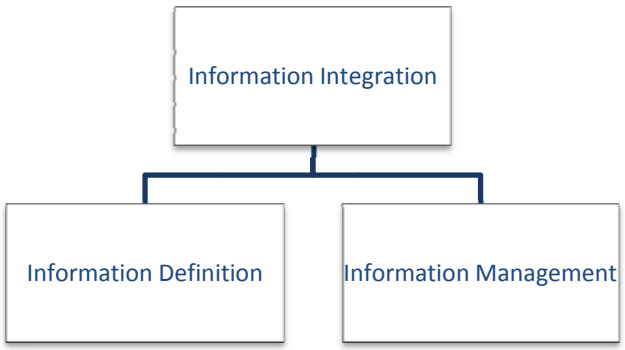


Figure 4.14: Information Integration

Definition

This includes defining the entire information, the components of which are as follows:

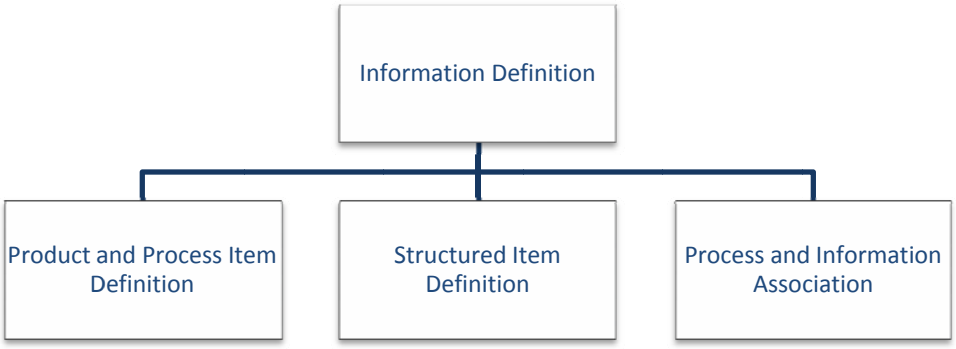


Figure 4.15: Information Definition

Product and Process

This is the sub-level of defining the process definition, which comprises of the following sub-components:

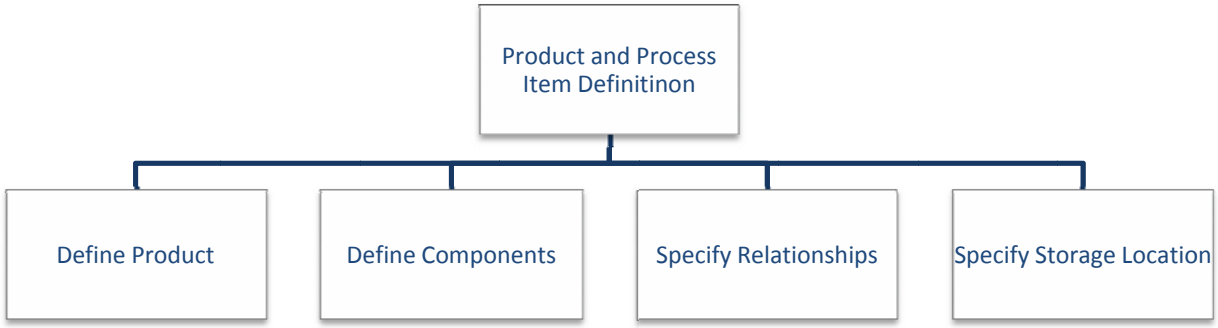


Figure 4.16: Product and Process

Define Product

What actually the product is, what are its attributes, properties, and functions are attributed in this component

Define Components

The interconnected components of the product being defined, are declared here, the definition of which is already mentioned at some other node as product definition. Here, we just reference the products, to make the entire product, productive

Specify Relationships

This includes the relationship of the product with the defined processes. This includes how it is created, what dependent products are, and how it will be utilized in next or previous production.

Specify Storage Location

This includes the storage location where the products will actually being placed.

Structures

Unlike product definition, in this part we define how the information will be structured. This consists of the following information

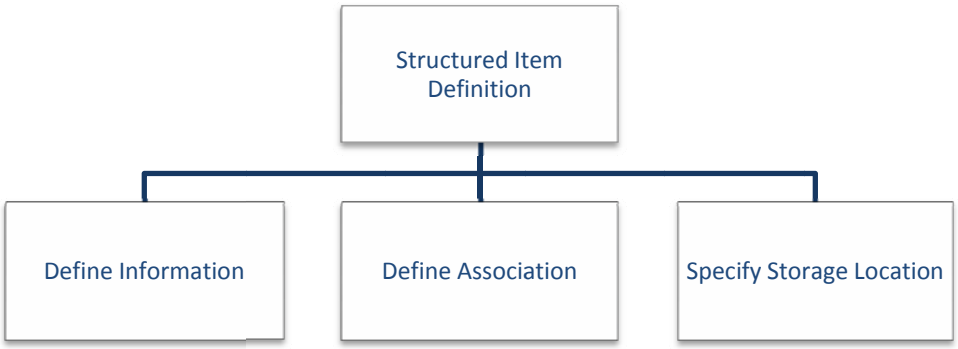


Figure 4.17: Structured Item

Define Information

We first define the information item which actually tells what the structure is about to end-users

Define Association

We then declare its associatively with different other information in the context. Thus an interconnectivity of information can be analyzed.

Specify Storage Location

Where the information will be put after being processed is mentioned in this part of the information item structure definition.

Process Information

The third component of the information is how will it be processed, that is the algorithm through which the data is transformed to information, is mentioned here.

This is how the complete information is defined to re-utilized the same, effectively.

Management

After definition, the other concern is the management of maintained/defined information. It comprises of the following parts:

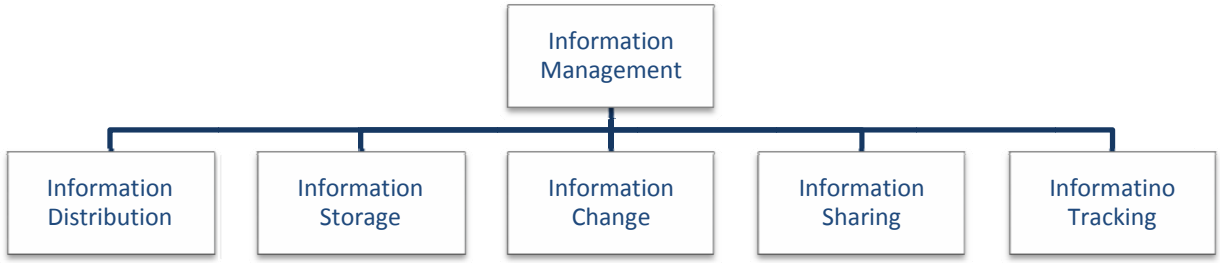


Figure 4.18: Information Management

Distribution

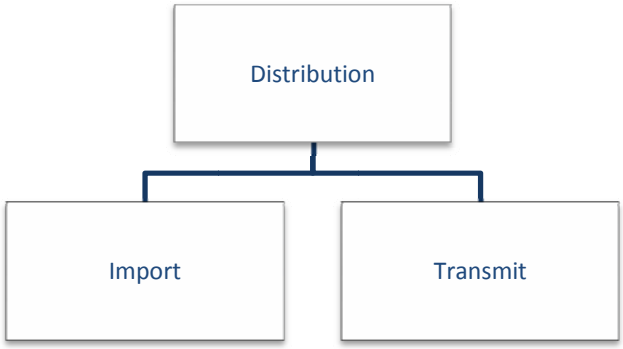


Figure 4.19: Distribution

Import

This component specifies the import function of how and where the information will be distributed.

Transmit

This component specifies the transmission function by which medium the information will be distributed.

Storage

The other management aspect is the storage and retrieval of the information, which comprises of following components

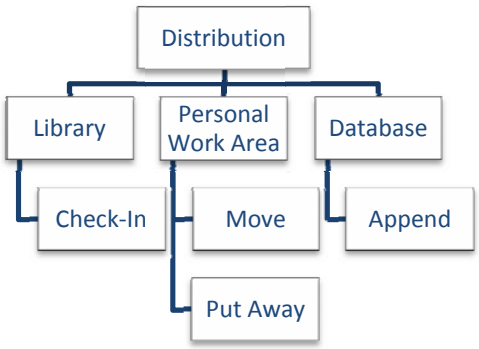


Figure 4.20: Distribution 2

Library

The first location where we keep information is the library, which comprises of the following functions

Check-In

If we retrieve any information from the library, we name it check-in. Thus retrieval of information from library is performed through this function

Work Area

The other area where the information system should allow having repository of information is personal work area

Move

This function allows moving data from personal work area to other location or wise versa

Put Away

This function is equivalent to deleting particular information from personal work area.

Database

The third option is to keep information in central repository, in the form of database

Append

This function allows to add more and more information, to have the warehousing solution for data-mining or trend analysis

Change

The third component in management of information is how we can change particular information. This defines change management policy and procedures, as any alteration may depict incorrect analytic information

Sharing

This component of management information is to how the information, being maintained in repository can be shared. There exists different function of sharing of each area, what we have mentioned in Storage section, earlier.

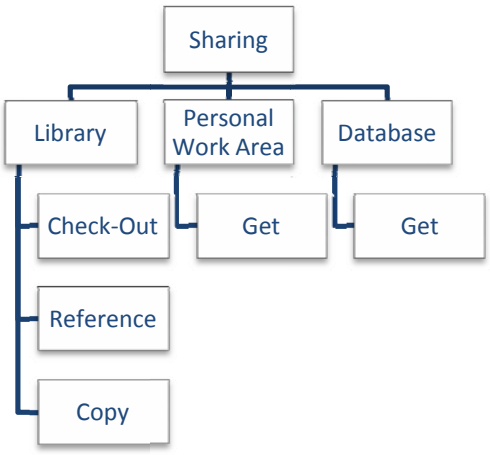


Figure 4.21:Sharing

Library

Copy

We can directly copy the data and share it from library.

Reference

We can refer the information so it cannot be duplicated rather being referenced

Check-Out

There should exists check-out function to let the information pull from library

Work Area

Get

In personal work area, we can share information by authorizing Get function to retrieve the data and therefore, can share.

Database

Get

As mentioned above, the same is applicable on databases

Tracking

The final component is to track certain information, which is also part of the information management

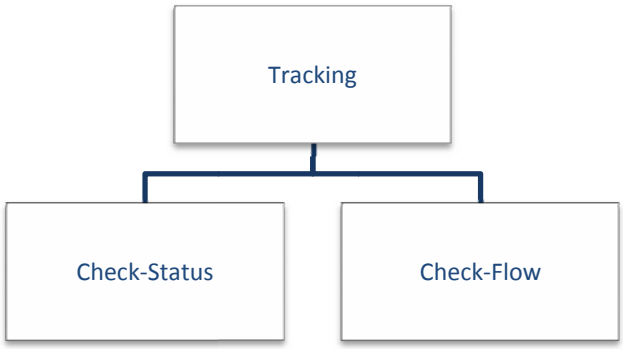


Figure 4.22: Tracking

Check Status

There should be a check and balance system by availability of status of particular information.

Check Flow

At which stage the information is and by when it will reach to certain should be available for streamlined processing of data.

4.2.6 Communication

The last concern of IS is how the Information should be communicated within the integrated process. This includes choosing the medium to transmit information, securing data so unauthorized personnel cannot retrieve, and on what policy the systems are connected to have uninterrupted services while communicating with each other.

4.3 Semantic Modeling

4.3.1 Introduction

Company information system is a complete process and its building and development needs to have certain points to ponder i.e.

- The complete knowledge of the organization
- Implementation of the information within organization
- The detailed information about the evolvement of organization

The important step in business is semantic modeling which can lead to make business successful**.**The purpose of the semantic modeling tends to clarify the specifications in terms of the additions of formalities plus promising the consistencies to prove the business definitions. This condition is clearer by comprehending and defining the business terms in order to make rational relationships with each other by the objective of putting them into certain hierarchy or adding classification in terms of taxonomy and this all is done by adding some rules.

The benefit of semantic modeling tends to clarify the relationship between business entities.Thus it is more obvious that this model tends to give the ability to make a relationship between two different abstracts and provides information about the relations with these elements. Semantic model is a tool to provide the modeling of entities and give the relationship between them. Taxonomy of classes is a tool to represent the real world which is made by the set of entities. The representation of the ideas is put under ontology which seeks to provide a kind of vocabulary through which user can form queries through this semantic model. The semantic model might represent the relationship of entities and defines the constraints regarding their relationships. The process tends to clarify the essence of semantic makeup.

Semantic model: advantages

Semantic model gives a complete picture of representing and informing data different from rational database. Using semantic models can provide an advantage by using large and complex data information through differentiated products and services as well.

Advantages of Semantic databases in relation with Relational databases:

- **The data model gives better flexibility**: the relational database tends to provide a whole changed set up of structure database which includes installation and migration of the structure and data respectively. The relationships are related with semantic database in which just one entry is needed to be added if new relationship is prone to add. Therefore, semantic model is a convenient way to adapt changes or assets and also provides a platform through which new questions are answered that was priory not a part of initial design.

- **Complex information through simple queries:** the relational database tends to end up in complex queries in which "where" or "table joins" clauses are added. The database structure is complex when the object relationships are remote. The semantic model is reasonable in showing if the specified link in semantic model tends to show up, the queries will automatically be deduced from the mechanism of knowledge. This simple procedure can be used by end-user and he can perform the queries himself if basic graphic query formation tools are provided.

- **Improves alignment**: by extending and relating work from the Business Glossary with added semantics, which helps building a tangible bridge between business and IT and business related fields can converge towards developing contextual applications dealing with Master Data Management, Data Integration, Business Intelligence..

After analyzing the different aspects of the hierarchy structure and understanding the depth of each function, This section show how the business hierarchy structure it's modeled into semantic using OWL ,

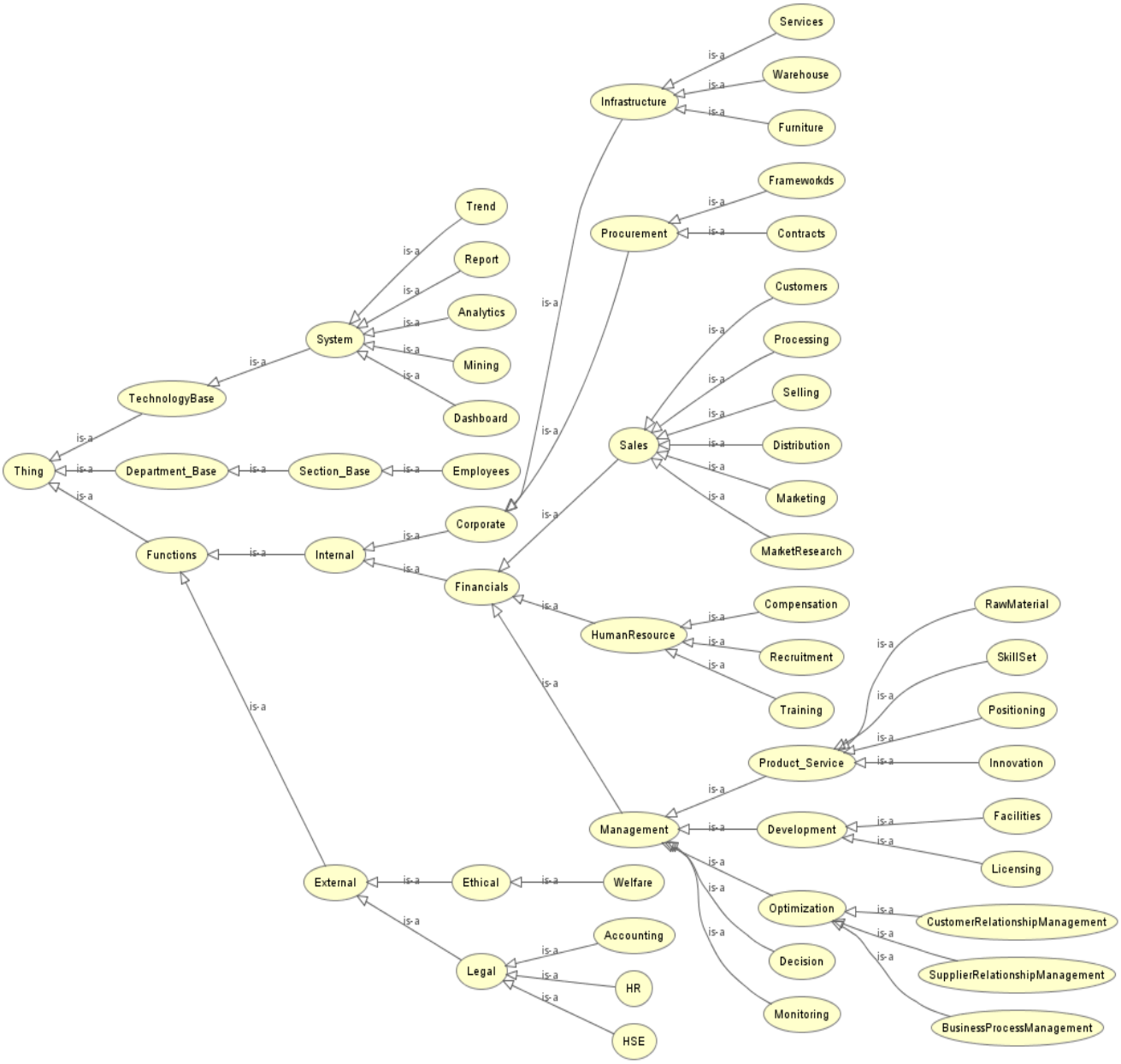


Figure 4.23: EID Ontology

4.3.2 Application of Knowledge

Knowledge is the prerequisite to design any OWL, which is why it is a bit difficult as compared to other ontologies. For business needs, we have to cover different aspects. We have different aspects which includes financials, legal's, resources, maintenance etc. However, it is not a good idea to have a flat hierarchy which is difficult to manage for any business. It should be well structured. With the idea, we can group the similar functions having same attributes. We segregate them as **Internal** and **External**.

4.3.3 Internal and External forces

By internal, we mean to group all the functions that the organization is required to run successfully. This includes financials of the company, procurements, resources, supply chain management, production etc. These are common in terms that they all have employees or resources that can handle the situations, occurs at different stages. Also, they would be recurring payment activities, i.e. payroll must be run monthly or after every specific period of time as per terms and conditions. They all will be controlled by the management. Thus management is also a function to manage and control all mentioned functions.

By external, we mean the laws, ethics, welfare work etc. that are influenced from outside. This influence can be imposed either by the country rules and regulations, local laws, transportation laws, global laws, change in market etc. These cannot be controlled and thus because of external forces, we have to control internal resources to coup up with the environmental changes.

4.3.4 Functions

Following are the main functions, which are actually grouped of the similar activities

Ethical

These include all the ethical work, following ethical practices to have better management within the resources of organization. This should be influenced as they may portray a better impact to customers

Financials

These includes all the activities that involved finances

Infrastructure

These includes the maintenance of existing infrastructure as well as to come up with innovative and new infrastructure for more productivity

Legal

These include following all the rules that are made by external bodies

OWL - Function

```
<owl:Class rdf:ID="Functions"/>

<owl:Class rdf:ID="Ethical">

<rdfs:subClassOf rdf:resource="#Functions"/>

</owl:Class>

<owl:Class rdf:ID="Financials">

<rdfs:subClassOf rdf:resource="#Functions"/>

</owl:Class>

<owl:Class rdf:ID="Infrastructure">

<rdfs:subClassOf rdf:resource="#Functions"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

<owl:Class rdf:ID="Legal">

<rdfs:subClassOf rdf:resource="#Functions"/>

</owl:Class>
```

Explanation

Here, you can clearly see that we have defined base class Function, with its properties, like Unique property of Department and may involve the category of either Internal or External.

We have defined main class "Functions". The rest we have created the sub-classes of Ethical, Financials, Infrastructure and Legal. We have set the Base properties of Departments to Infrastructure so as to directly inherit all the concerning properties to it.

4.3.5 OWL - Departments

Now we will explain each of the department which are classified of sub classes of functions. However, as they are also the super classes of their own functions, segregated among different sections, we will create OWL of these mid-level classes. Also, as we know that all the departments has unique properties of having Department Manager, No. of Employees, and Job Descriptions etc., we will create an abstract class, so that the unique properties can be inherited from this class to other department's classes. Thus, the following is the abstract class, having main properties that must be mandatory for all other classes.

<owl:Class rdf:ID="Department_Base">

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

<rdfs:subClassOf rdf:resource="&owl;Thing"/>

</owl:Class>

You may see that we have also applied restriction, so only SOME departments can be inherited from them. We can declare the departments name here as well to restrict the members of the class.

Also, as it is mandatory to segregate sections of departments, and that each section has a section head as well who monitors and controls the functions, a base class for section is also a good option to develop OWL in the most efficient way. Now that, we have base class, we can directly inherit super classes, i.e. departments

<owl:Class rdf:ID="Section_Base">

<rdfs:subClassOf rdf:resource="#Department_Base"/>

</owl:Class>

We can see that the base class of Section is the inherited from Base of Department class. Logically speaking, Sections are part of Departments, which further are part of Functions. Thus creating base classes of Sections and Departments help us creating their instances for individual instances, having inherited property, which is unique for them.

Coming up with the development of Infrastructure function, we would have following OWL

OWL - Legal Function w/ Explanation

<owl:Class rdf:ID="Accounting">

<rdfs:subClassOf rdf:resource="#Legal"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

From above, we have created the Department "Accounting", which is a sub class of Function class "Legal". We also have created properties, which can be added as desired. We can apply restrictions by maintaining further properties. Further, we have Department's Base class on top of Department, so that the unique properties, defined in respective class, are automatically to its child or sub classes.

<owl:Class rdf:ID="HealthnSafety">

<rdfs:subClassOf rdf:resource="#Legal"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

From above, we have created the Department "Health and Safety", which is a sub class of Function class "Legal". We also have created properties, which can be added as desired. We can apply restrictions by maintaining further properties. Further, we have Department's Base class on top of Department, so that the unique properties, defined irrespective class, are automatically to its child or sub classes.

<owl:Class rdf:ID="HR">

<rdfs:subClassOf rdf:resource="#Legal"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

From above, we have created the Department "Human Resource", which is a sub class of Function class "Legal". We also have created properties, which can be added as desired. We can apply restrictions by maintaining further properties. Further, we have Department's Base class on top of Department, so that the unique properties, defined in respective class, are automatically to its child or sub classes.

OWL - Infrastructure Function w/ Explanation

```
<owl:Class rdf:ID="Furniture_Equipment">

<rdfs:subClassOf rdf:resource="#Infrastructure"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>
```

From above, we have created the Department "Furniture and Equipments", which is a sub class of Function class Infrastructure. We also have created properties, which can be added as desired. We can apply restrictions by maintaining further properties. Further, we have Department's Base class on top of Department, so that the unique properties, defined in respective class, are automatically to its child or sub classes.

```
<owl:Class rdf:ID="Office_Warehouse">

<rdfs:subClassOf rdf:resource="#Infrastructure"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>
```

From above, we have created the Department "Office/Warehouse", which is a sub class of Function class Infrastructure. We also have created properties, which can be added as desired. We can apply restrictions by maintaining further properties. Further, we have Department's Base class on top of Department, so that the unique properties, defined in respective class, are automatically to its child or sub classes.

<owl:Class rdf:ID="Services_Utilities">

<rdfs:subClassOf rdf:resource="#Infrastructure"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

From above, we have created the Department "Service/Utilities", which is a sub class of Function class Infrastructure. We also have created properties, which can be added as desired. We can apply restrictions by maintaining further properties. Further, we have Department's Base class on top of Department, so that the unique properties, defined in respective class, are automatically to its child or sub classes.

OWL - Ethical w/ Explanation

<owl:Class rdf:ID="**Ethical**">

<rdfs:subClassOf rdf:resource="#Functions"/>

</owl:Class>

<owl:Class rdf:ID="Welfare">

<rdfs:subClassOf rdf:resource="#Ethical"/>

</owl:Class>

From above, we have created the Department "Welfare", which is a sub class of Function class Ethical. Here, we haven't created any super classes, because we are not aware of what ethical properties could be. This is because, employers don't know if they will have enough budget to spend in welfare OR they cannot plan prior and is their decision is extempore, i.e. may be dependent on some natural disaster, etc.

OWL - Financials w/ Explanation

This is the major part that any company may play to execute or launch program or product. Thus it is divided into different departments, which are further divided into different sections.

OWL - Sales w/ Explanation

<owl:Class rdf:ID="Sales">

<rdfs:subClassOf rdf:resource="#Financials"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

From above, we have created the Department "Sales", which is a sub class of Function class "Finance". We also have created properties, which can be added as desired. We can apply restrictions by maintaining further properties. Further, we have Department's Base class on top of Department, so that the unique properties, defined in respective class, are automatically to its child or sub classes.

Further sub-classes are as follows:

<owl:Class rdf:ID="Distribution">

<rdfs:subClassOf rdf:resource="#Sales"/>

</owl:Class>

The section, "Distribution", is part of the "Sales" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Sales". Also, being a sub-class of "Sales", it is indirectly linked to function "Financials". So if we select "Distribution", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="LeadManagement">

<rdfs:subClassOf rdf:resource="#Sales"/>

</owl:Class>

The section, "Lead Management", is part of the "Sales" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Sales". Also, being a sub-class of "Sales", it is indirectly linked to function "Financials". So if we select "Lead Management", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="Marketing">

<rdfs:subClassOf rdf:resource="#Sales"/>

</owl:Class>

The section, "Marketing", is part of the "Sales" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Sales". Also, being a sub-class of "Sales", it is indirectly linked to function "Financials". So if we select "Marketing", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="MarketResearch">

<rdfs:subClassOf rdf:resource="#Sales"/>

</owl:Class>

The section, "Market Research", is part of the "Sales" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Sales". Also, being a sub-class of "Sales", it is indirectly linked to function "Financials". So if we select "Market Research", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="Processing">

<rdfs:subClassOf rdf:resource="#Sales"/>

</owl:Class>

The section, "Processing or Production", is part of the "Sales" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Sales". Also, being a sub-class of "Sales", it is indirectly linked to function "Financials". So if we select "Processing or Production", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="Selling">

<rdfs:subClassOf rdf:resource="#Sales"/>

</owl:Class>

The section, "Selling", is part of the "Sales" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Sales". Also, being a sub-class of "Sales", it is indirectly linked to function "Financials". So if we select "Selling", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="ServicingCustomers">

<rdfs:subClassOf rdf:resource="#Sales"/>

</owl:Class>

The section, "Customer Relationship", is part of the "Sales" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Sales". Also, being a sub-class of "Sales", it is indirectly linked to function "Financials". So if we select "Customer Relationship", we can directly have information of "Financials" or vice versa.

OWL - Product or Service w/ Explanation

<owl:Class rdf:ID="Product_Service">

<rdfs:subClassOf rdf:resource="#Financials"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

From above, we have created the Department "Product or Service", which is a sub class of Function class "Finance". We also have created properties, which can be added as desired. We can apply restrictions by maintaining further properties. Further, we have Department's Base class on top of Department, so that the unique properties, defined in respective class, are automatically to its child or sub classes.

Further sub-classes are as follows:

<owl:Class rdf:ID="CompetitivePostion">

<rdfs:subClassOf rdf:resource="#Product_Service"/>

</owl:Class>

The section, "Positioning", is part of the "Product or Service" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Product or Service". Also, being a sub-class of "Product or Service", it is indirectly linked to function "Financials". So if we select "Positioning", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="Innovation">

<rdfs:subClassOf rdf:resource="#Product_Service"/>

</owl:Class>

The section, "Innovation", is part of the "Product or Service" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Product or Service". Also, being a sub-class of "Product or Service", it is indirectly linked to function "Financials". So if we select "Innovation", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="RawMaterial">

<rdfs:subClassOf rdf:resource="#Product_Service"/>

</owl:Class>

The section, "Raw Material", is part of the "Product or Service" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Product or Service". Also, being a sub-class of "Product or Service", it is indirectly linked to function "Financials". So if we select "Raw Material", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="SkillSet">

<rdfs:subClassOf rdf:resource="#Product_Service"/>

</owl:Class>

The section, "Skill Set", is part of the "Product or Service" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Product or Service". Also, being a sub-class of "Product or Service", it is indirectly linked to function "Financials". So if we select "Skill Set", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="Tools">

<rdfs:subClassOf rdf:resource="#Product_Service"/>

</owl:Class>

The section, "Tools", is part of the "Product or Service" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Product or Service". Also, being a sub-class of "Product or Service", it is indirectly linked to function "Financials". So if we select "Tools", we can directly have information of "Financials" or vice versa.

OWL - Management w/ Explanation

```
<owl:Class rdf:ID="Management">

<rdfs:subClassOf rdf:resource="#Financials"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>
```

From above, we have created the Department "Management", which is a sub class of Function class "Finance". We also have created properties, which can be added as desired. We can apply restrictions by maintaining further properties. Further, we have Department's Base class on top of Department, so that the unique properties, defined in respective class, are automatically to its child or sub classes.

Further sub-classes are as follows:

```
<owl:Class rdf:ID="Facilities_Locations">

<rdfs:subClassOf rdf:resource="#Management"/>

</owl:Class>
```

The section, "Locations and Facilities", is part of the "Management" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Management". Also, being a sub-class of "Management", it is indirectly linked to function "Financials". So if we select "Locations and Facilities", we can directly have information of "Financials" or vice versa.

```
<owl:Class rdf:ID="HumanResource">

<rdfs:subClassOf rdf:resource="#Management"/>
```

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

The section, "Human Resource Management", is part of the "Management" department, having all the properties that its base class possesses. It is therefore classified as sub- class of the department "Management". Also, being a sub-class of "Management", it is indirectly linked to function "Financials". So if we select "Human Resource Management", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="Communication">

<rdfs:subClassOf rdf:resource="#HumanResource"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

Communication is the spinal cord of the organization. The more it is efficient, the quickly the messages can be delivered, and the more the efficient decision can be taken.

<owl:Class rdf:ID="Recruitment">

<rdfs:subClassOf rdf:resource="#HumanResource"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

Recruitment is what inducting resources that can help increase productivity. Thus plays major part in the organization.

This section is part of the "Human Resource Management" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Resource Management". Also, being a sub-class of "Resource Management", it is indirectly linked to function "Financials". So if we select "Human Resource Management", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="Training">

<rdfs:subClassOf rdf:resource="#HumanResource"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#hasProperties"/>

<owl:someValuesFrom rdf:resource="#Department_Base"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

The section is responsible to train resources so that they are up to date with the current technology, and after learning can provide their input to their work, for more efficient processing.

<owl:Class rdf:ID="IPR">

<rdfs:subClassOf rdf:resource="#Management"/>

</owl:Class>

The section, "IPR", is part of the "Management" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Management". Also, being a sub-class of "Management", it is indirectly linked to function "Financials". So if we select "IPR", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="Licensing_Software">

<rdfs:subClassOf rdf:resource="#Management"/>

</owl:Class>

The section, "Software Licensing", is part of the "Management" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Management". Also, being a sub-class of "Management", it is indirectly linked to function "Financials". So if we select "Software Licensing", we can directly have information of "Financials" or vice versa.

<owl:Class rdf:ID="Productivity">

<rdfs:subClassOf rdf:resource="#Management"/>

</owl:Class>

The section, "Productivity", is part of the "Management" department, having all the properties that its base class possesses. It is therefore classified as sub-class of the department "Management". Also, being

a sub-class of "Management", it is indirectly linked to function "Financials". So if we select "Productivity", we can directly have information of "Financials" or vice versa.

Under productivity section, we have different ways to analyze, monitor and control all over the functions through different techniques. That is why, we have to create sub- classes, as each individual is having different function and instances that may or may not be for different user as different entity. Thus members of such classes can also be developed, and can be restricted to say, higher management only

<owl:Class rdf:ID="Analytics">

<rdfs:subClassOf rdf:resource="#Productivity"/>

</owl:Class>

Analytics, is a major part in monitoring and controlling the organization efficiently by having all the information on their finger tips. Records to each resource is available, with the traceability. This helps in quickly identifying the bottleneck and resolve it quickly to have smooth productivity.

<owl:Class rdf:ID="BPM">

<rdfs:subClassOf rdf:resource="#Productivity"/>

</owl:Class>

Business Process Management is where the management decides what the processes should be, and how technology can be adopted to further improve or simply the business process to make it more efficient.

<owl:Class rdf:ID="Dashboards">

<rdfs:subClassOf rdf:resource="#Productivity"/>

</owl:Class>

To process each business processes, KPIs are introduced in the form of dashboard so that the alerts can be generated for inefficient processes.

<owl:Class rdf:ID="Mining_Trends">

<rdfs:subClassOf rdf:resource="#Productivity"/>

</owl:Class>

Through trending, the organization can focus more their productivity, that is can predict either more customers or less would be interested in coming season. So to gear the organization's resources accordingly.

<owl:Class rdf:ID="Operational_Logistics">

<rdfs:subClassOf rdf:resource="#Productivity"/>

</owl:Class>

This includes the management of all the operational logistics work, to help making the suppliers available to provide required logistics at required time. Thus, all of the above can be categorized under Productivity, which is being managed and controlled by management, and for which finance is responsible to take care of. If there exists monetary benefits, the outcome would surely be better.

4.4 Engineering Domain

4.4.1 Introduction

This section explains how the engineering information management is deployed. With its representation in OWL Ontology, it will surely be a great input to semantic web, making it accessible to everyone, to further work over the semantic and to come up with the optimized version. This will surely a step towards more improvement in how the structure is actually designed for any Engineering information management system.

4.4.2 Overview

The initial step is to first analyze the different aspects of the requirements. Further, we need to go in depth in each of its feature. This is possible if you are familiar with the core IT, familiar with its functions or by outsourcing to knowledge engineer, who is actually the domain expert and can better provide services. We worked and came up with the following high-level OWL for any Engineering Information Management. The same structure can be adopted by the organizations, to better manage their IT infrastructure, at reduced recurring and maintenance cost, as well as aware of what their functions are,

for what are they responsible, and what services will be provided to users. The Service Level Agreement may then be incorporated to automate the workflow process, etc.

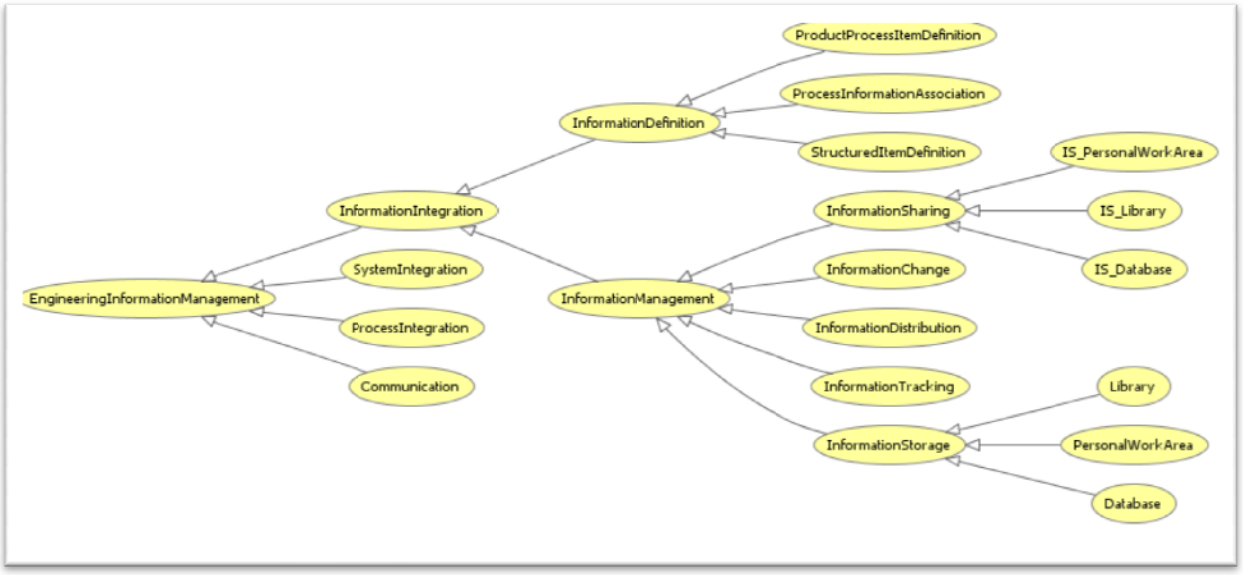


Figure 4.23 Engineering Ontology

4.4.3 Semantic Model of Engineering Ogranization

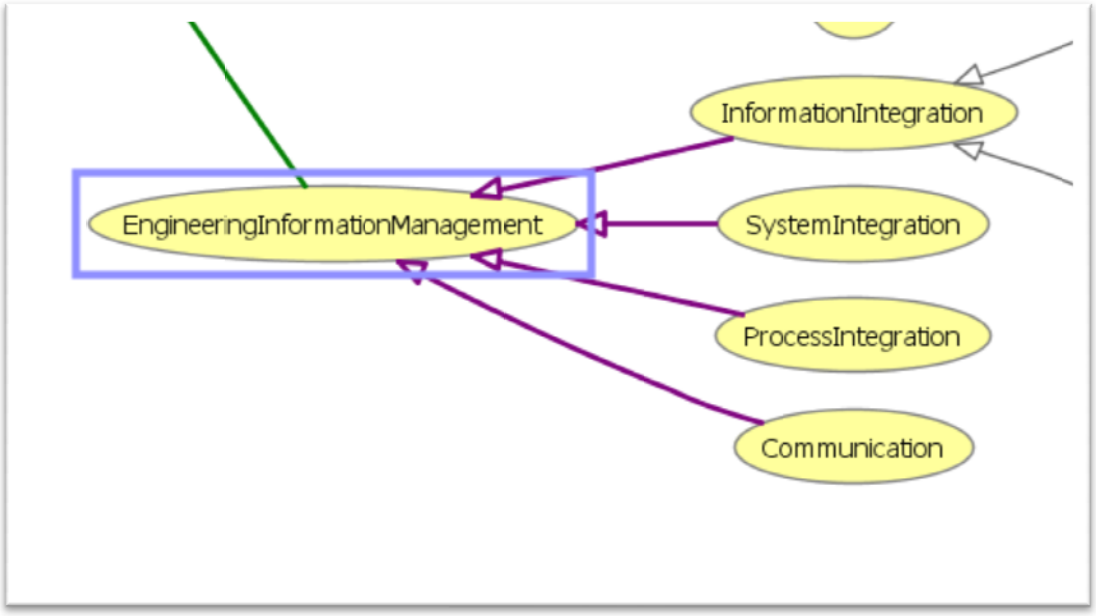


Figure 4.24 Engineering information management

The integration of information, its maintenance within the specified repository, its dependent functions and what information should remain for longer period and which should be discarded or archived.

The second comes the deployment of systems that includes which server architecture are we going to follow, which systems would be integrated and which runs in isolation, what would be the basis of 'Disaster Recovery' site, which are the critical systems, what monitoring systems should be deployed, which hardware firewall are to be installed, etc. It is cumbersome with the fact that the companies are getting their steps globally. The infrastructure gets more complicated with the fact that the data may or may not reach to its actual destination. This includes availability of different hops in between. Further, to secure important data is of course a big challenge.

The most important part is the automation of processes, and its integration with other processes, to not only help doing the task more effectively and efficiently, but also to do in less time with complete availability of information at each step. The same does help in monitoring the entire business process and develop Key Performance Indicator, to take actions against any bottlenecks within the processes.

The final comes the Communication, which plays important role in aforementioned feature. The better the communication mechanism exist, the more the effective decisions can be taken promptly.

Information Integration

The integration of information is a critical part. It involves inter-connectivity of data, how they are architected or modeled. Where they will be maintained, how they are going to maintain, which repository is to be used, its dependent functions and what information should remain for longer period and which should be discarded or archived. It comprises of **Definition** and how they are being **Managed:**

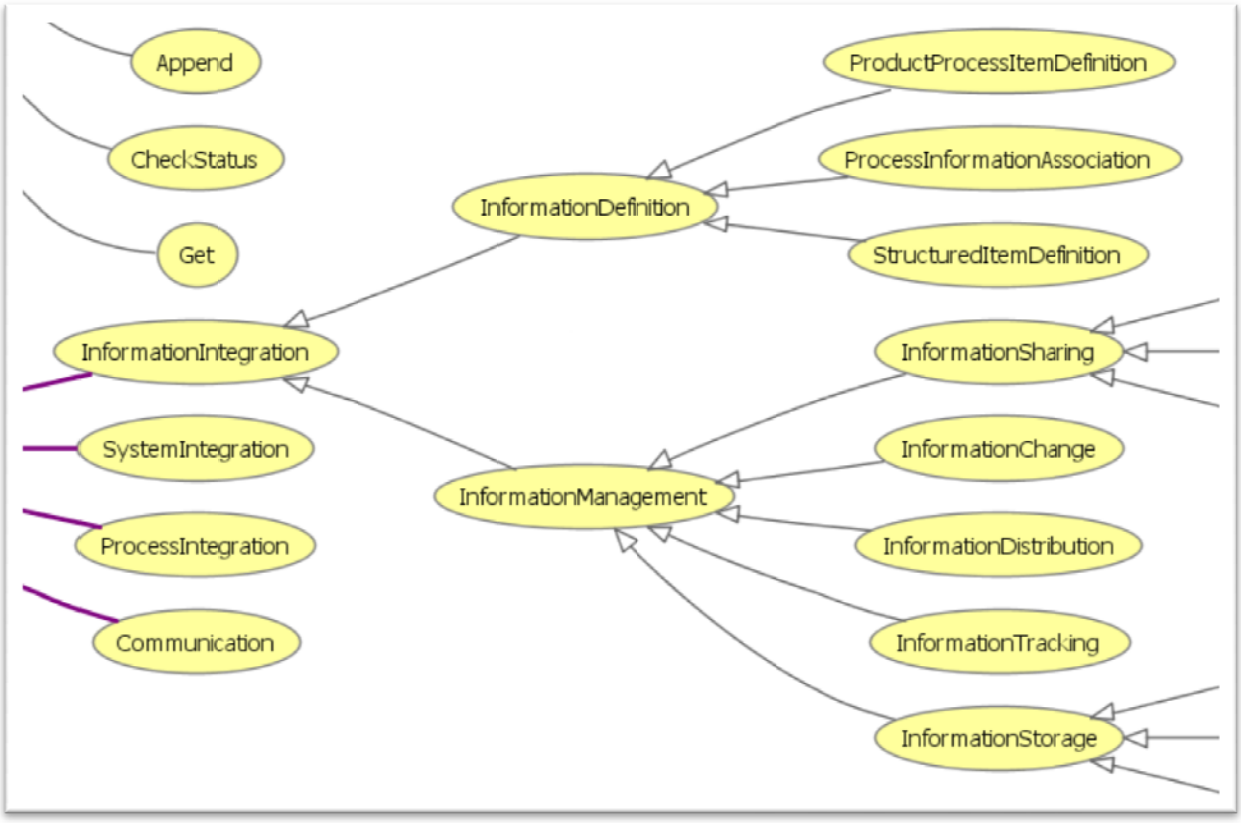


Figure 4.25 Information Integration

OWL Function

```
<owl:Class rdf:ID="InformationIntegration">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="EngineeringInformationManagement"/>
```

```
  </rdfs:subClassOf>
 </owl:Class>
```

Explanation

We can see that the base class of Information Integration is the inherited from Engineering Information Management class. Logically speaking, Information Integration is part of Engineering Information Management. Thus creating base classes of Information Integration and Engineering Information Management help us creating their instances for individual instances, having inherited property, which is unique for them.

Information Definition

OWL Function

```
<owl:Class rdf:about="#InformationDefinition">
  <rdfs:subClassOf rdf:resource="#InformationIntegration"/>
 </owl:Class>
```

Explanation

We can see that the base class of InformationDefinition is the inherited from InformationIntegration class. Logically speaking, InformationDefinition is part of Engineering Information Management. Thus creating base classes of InformationDefinition and InformationIntegration help us creating their instances for individual instances, having inherited property, which is unique for them.

As shown above, it involved defining the information. It comprises of three different parts for properly defining the same:

Product and Process Item Definition

This includes itemizing the information, and consequently defining the same for developing a proper understanding for semantics

OWL Function

```
<owl:Class rdf:ID="ProductProcessItemDefinition">
```

```
<rdfs:subClassOf>
 <owl:Restriction>
  <owl:allValuesFrom rdf:resource="#Specify_Storage_Location"/>
  <owl:onProperty>
   <owl:ObjectProperty rdf:ID="hasFunction"/>
  </owl:onProperty>
 </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
 <owl:Restriction>
  <owl:allValuesFrom>
   <owl:Class rdf:ID="Specify_Relationship"/>
  </owl:allValuesFrom>
  <owl:onProperty rdf:resource="#hasFunction"/>
 </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
 <owl:Restriction>
  <owl:onProperty rdf:resource="#hasFunction"/>
  <owl:allValuesFrom>
   <owl:Class rdf:ID="Define_Component"/>
```

```
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasFunction"/>
      <owl:allValuesFrom rdf:resource="#Define_Product"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="InformationDefinition"/>
  </rdfs:subClassOf>
 </owl:Class>
```

Explanation

We can see that the base class of ProductProcessItemDefinition is the inherited from InformationDefinition class. Logically speaking, ProductProcessItemDefinition is part of Engineering Information Management. Thus creating base classes of ProductProcessItemDefinition and InformationDefinition help us creating their instances for individual instances, having inherited property, which is unique for them.

Secondly, you may have noted that there exists following restrictions, which means, this class can only have following functions:

Specify Storage Location
Specify Relationship
Define Component
Define Product

Process Information Association

This includes how the information will be processed, and after being processed, what will be the output and how will it be utilized in other processes for interconnectivity between them for developing a proper understanding for semantics

OWL Function

```
<owl:Class rdf:ID="ProcessInformationAssociation">
<rdfs:subClassOf rdf:resource="#InformationDefinition"/>
</owl:Class>
```

Explanation

We can see that the base class of ProductProcessItemDefinition is the inherited from InformationDefinition class. Logically speaking, ProductProcessItemDefinition is part of Engineering Information Management. Thus creating base classes of ProductProcessItemDefinition and InformationDefinition help us creating their instances for individual instances, having inherited property, which is unique for them.

Secondly, you may have noted that there exists following restrictions, which means, this class can only have following functions:

- Specify Storage Location
- Specify Relationship
- Define Component
- Define Product

Structured Item Definition

This includes in which structure will the item be placed, and what actually defines the structure.

OWL Function

```
<owl:Class rdf:ID="StructuredItemDefinition">
<rdfs:subClassOf>
```

```
  <owl:Restriction>
    <owl:allValuesFrom>
      <owl:Class rdf:ID="DefineAssociation"/>
    </owl:allValuesFrom>
    <owl:onProperty rdf:resource="#hasFunction"/>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasFunction"/>
    <owl:allValuesFrom>
      <owl:Class rdf:ID="DefineInfoItem"/>
    </owl:allValuesFrom>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasFunction"/>
    <owl:allValuesFrom rdf:resource="#Specify_Storage_Location"/>
  </owl:Restriction>
</rdfs:subClassOf>
```

```
  <rdfs:subClassOf>
    <owl:Class rdf:about="#InformationDefinition"/>
  </rdfs:subClassOf>
 </owl:Class>
```

Explanation

We can see that the base class of ProductProcessItemDefinition is inherited from the InformationDefinition class. Logically speaking, ProductProcessItemDefinition is part of Engineering Information Management. Thus creatr the base classes of ProductProcessItemDefinition and InformationDefinition help us creating their instances for individual instances, having inherited property, which is unique for them.

Secondly, you may have noted that there exists following restrictions, which means, this class can only have following functions:

Specify Storage Location
Specify Relationship
Define Component
Define Product

Information Management

OWL Function

```
<owl:Class rdf:about="#InformationManagement">
  <rdfs:subClassOf rdf:resource="#InformationIntegration"/>
 </owl:Class>
```

Explanation

We can see that the base class of InformationManagement is the inherited from InformationIntegration class. Logically speaking, InformationManagement is part of Engineering Information Management. Thus

creating base classes of InformationIntegration and InformationManagement help us creating their instances for individual instances, having inherited property, which is unique for them.

As shown above, it involved managing and structures the entire information. It comprises of five different parts for properly defining the same:

Information Sharing

Sharing is the most critical factor in the sense that if the unconcerned information is shared to unauthorized people, it may harm the entire organization. For instance, sharing information of new product to the people, who might convey the same to competitors. The same can market the same before their actual inventor etc. This includes 3 other sub-classes.

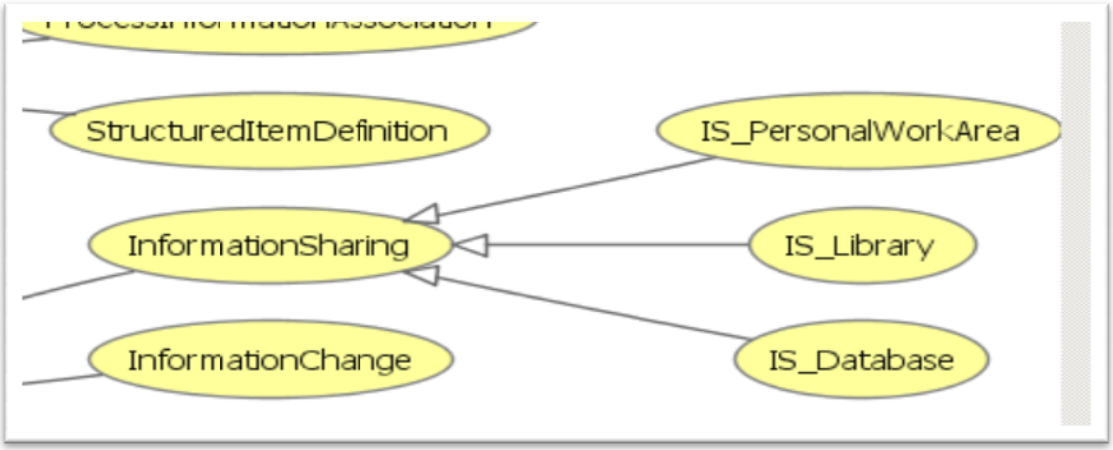


Figure 4.26: Information Sharing

OWL Function

```
<owl:Class rdf:about="#InformationSharing">

 <rdfs:subClassOf rdf:resource="#InformationManagement"/>

</owl:Class>
```

Explanation

We can see that the base class of Information Sharing is inherited from Information Management class. Logically speaking, Information Sharing is part of Engineering Information Management. Thus creating

base classes of Information Sharing and Information Management help us creating their instances for individual instances, having inherited property, which is unique for them.

Personal Work Area

This involves sharing of personal work to only limited people, say team.

OWL Function

```
<owl:Class rdf:ID="IS_PersonalWorkArea">
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:allValuesFrom>
     <owl:Class rdf:ID="Get"/>
    </owl:allValuesFrom>
    <owl:onProperty rdf:resource="#hasFunction"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Class rdf:ID="InformationSharing"/>
  </rdfs:subClassOf>
 </owl:Class>
```

Explanation

We can see that the base class of IS_Personal Work Area is the inherited from Information Sharing class. Logically speaking, IS_Personal Work Area is part of Engineering Information Management. Thus creating base classes of IS_Personal Work Area and Information Sharing help us creating their instances for individual instances, having inherited property, which is unique for them.

Secondly, you may have noted that there exists following restrictions, which means, this class can only have following functions:

Get function to retrieve information only

Library

This involves maintaining the information or sharing the same within the limited group of period, say department

OWL Function

```
<owl:Class rdf:ID="IS_Library">
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:allValuesFrom rdf:resource="#CheckOut"/>
    <owl:onProperty rdf:resource="#hasFunction"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#hasFunction"/>
    <owl:allValuesFrom>
     <owl:Class rdf:ID="Reference"/>
    </owl:allValuesFrom>
   </owl:Restriction>
  </rdfs:subClassOf>
```

```
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:allValuesFrom rdf:resource="#Copy"/>
    <owl:onProperty rdf:resource="#hasFunction"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Class rdf:about="#InformationSharing"/>
  </rdfs:subClassOf>
 </owl:Class>
```

Explanation

We can see that the base class of IS_Library is the inherited from Information Sharing class. Logically speaking, IS_Library is part of Engineering Information Management. Thus creating base classes of IS_Library and Information Sharing help us creating their instances for individual instances, having inherited property, which is unique for them.Secondly, you may have noted that there exists following restrictions, which means, this class can only have following functions:

Check Out
Reference
Copy

Database

The information can be shared and dump to database, which can be made available to everyone. This also includes, performing the Analytics, or analyzing Trend for Predictive Analysis

OWL Function

```
<owl:Class rdf:ID="IS_Database">
```

```
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:allValuesFrom>
     <owl:Class rdf:about="#Get"/>
    </owl:allValuesFrom>
    <owl:onProperty rdf:resource="#hasFunction"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#InformationSharing"/>
 </owl:Class>
```

Explanation

We can see that the base class of IS_Database is the inherited from Information Sharing class. Logically speaking, IS_Database is part of Engineering Information Management. Thus creating base classes of IS_Database and Information Sharing help us creating their instances for individual instances, having inherited property, which is unique for them.

Secondly, you may have noted that there exists following restrictions, which means, this class can only have following functions:

Get

Information Change

This involves the changes, involved to update or reform already maintained information. This may include the upgrade of information according to the utilization of latest technologies for better accessibility, etc.

OWL Function

```
<owl:Class rdf:ID="InformationChange">
```

```
  <rdfs:subClassOf rdf:resource="#InformationManagement"/>
 </owl:Class>
```

Explanation

We can see that the base class of InformationChange is the inherited from InformationManagement class. Logically speaking, InformationChange is part of Engineering Information Management. Thus creating base classes of InformationChange and InformationManagement help us creating their instances for individual instances, having inherited property, which is unique for them.

Information Distribution

This involves developing the communication channel and distribution of information to entire or limited group of organization. The channel should be secured enough so that no intervention is possible, and whatever should be conveyed, should convey.

OWL Function

```
<owl:Class rdf:ID="InformationDistribution">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Transmit"/>
      <owl:onProperty rdf:resource="#hasFunction"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Import"/>
      <owl:onProperty rdf:resource="#hasFunction"/>
```

```
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#InformationManagement"/>
 </owl:Class>
```

Explanation

We can see that the base class of InformationDistribution is the inherited from InformationManagement class. Logically speaking, InformationDistribution is part of Engineering Information Management. Thus creating base classes of InformationDistribution and InformationManagement help us creating their instances for individual instances, having inherited property, which is unique for them.

Secondly, you may have noted that there exists following restrictions, which means, this class can only have following functions:

Transmit
Import

Information Tracking

Where actually the information exists, what is the current status of information, if information is following some workflow, what actually it should be by now etc. comes under tracking.

OWL Function

```
<owl:Class rdf:ID="InformationTracking">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasFunction"/>
      <owl:allValuesFrom rdf:resource="#CheckFlow"/>
    </owl:Restriction>
  </rdfs:subClassOf>
```

```
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#hasFunction"/>
    <owl:allValuesFrom rdf:resource="#CheckStatus"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#InformationManagement"/>
 </owl:Class>
```

Explanation

We can see that the base class of InformationTracking is the inherited from InformationManagement class. Logically speaking, InformationTracking is part of Engineering Information Management. Thus creating base classes of InformationTracking and InformationManagement help us creating their instances for individual instances, having inherited property, which is unique for them.

Secondly, you may have noted that there exists following restrictions, which means, this class can only have following functions:

Check Flow
Check Status

Information Storage

Storage plays major part, which actually tells, how the data is to be stored within the hardware, should it be public or private, what actually should be done if hardware fails, where should the backup be done, how frequently the related process should be run, etc.

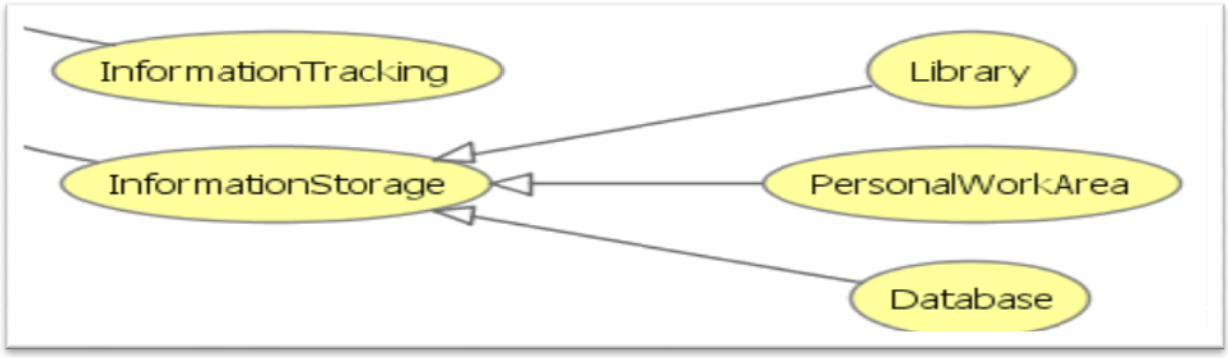


Figure 4.27 Information Storage

OWL Function

```
<owl:Class rdf:ID="InformationStorage">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="InformationManagement"/>
  </rdfs:subClassOf>
 </owl:Class>
```

Explanation

We can see that the base class of Information Storage is the inherited from Information Management class. Logically speaking, Information Storage is part of Engineering Information Management. Thus creating base classes of Information Storage and Information Management help us creating their instances for individual instances, having inherited property, which is unique for them

Personal Work Area

This involves sharing of personal work to only limited people, say team.

OWL Function

```
<owl:Class rdf:ID="PersonalWorkArea">
  <rdfs:subClassOf>
   <owl:Restriction>
```

```
      <owl:allValuesFrom>
        <owl:Class rdf:ID="PutAway"/>
      </owl:allValuesFrom>
      <owl:onProperty rdf:resource="#hasFunction"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Move"/>
      <owl:onProperty rdf:resource="#hasFunction"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#InformationStorage"/>
 </owl:Class>
```

Explanation

We can see that the base class of Personal Work Area is the inherited from Information Storage class. Logically speaking, Personal Work Area is part of Engineering Information Management. Thus creating base classes of Personal Work Area and Information Storage helps us create their instances for individual instances, having inherited property, which is unique for them.

Secondly, you may have noted that there exists following restrictions, which means, this class can only have following functions:

Put Away
Move

Library

This involves maintaining the information or sharing the same within the limited group of period, say department

OWL Function

```
<owl:Class rdf:ID="Library">
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#hasFunction"/>
    <owl:allValuesFrom rdf:resource="#CheckIn"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#InformationStorage"/>
 </owl:Class>
```

Explanation

We can see that the base class of Library is the inherited from Information Storage class. Logically speaking, Library is part of Engineering Information Management. Thus creating base classes of Library and Information Storage help us creating their instances for individual instances, having inherited property, which is unique for them.

Secondly, you may have noted that there exists following restrictions, which means, this class can only have following functions:

Check In

Database

The information can be shared and dump to database, which can be made available to everyone. This also includes, performing the Analytics, or analyzing Trend for Predictive Analysis

OWL Function

```
<owl:Class rdf:ID="Database">
  <rdfs:subClassOf>
```

```
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasFunction"/>
      <owl:allValuesFrom rdf:resource="#Append"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#InformationStorage"/>
 </owl:Class>
```

Explanation

We can see that the base class of Database is the inherited from Information Storage class. Logically speaking, Database is part of Engineering Information Management. Thus creating base classes of Database and Information Storage help us creating their instances for individual instances, having inherited property, which is unique for them.

Secondly, you may have noted that there exists following restrictions, which means, this class can only have following functions:

Append

System Integration

It is worth to mention that systems integration plays major role in setting up an infrastructure. As mentioned earlier, this includes integration of homogeneous and heterogeneous system, the monitoring system to check consistency between systems' integration and an application of how the structure is maintained for troubleshooting.

OWL Function

```
<owl:Class rdf:ID="SystemIntegration">
  <rdfs:subClassOf rdf:resource="#EngineeringInformationManagement"/>
 </owl:Class>
```

Explanation

We can see that the base class of System Integration is the inherited from System Integration class. Thus creating base classes of System Integration and Information Definition help us creating their instances for individual instances, having inherited property, which is unique for them.

Process Integration

This includes implementation of enterprise resource planning system and identifying gaps within business process. Consequently to develop processes to address the gaps, thus integrating the entire end-to-end process

OWL Function

```
<owl:Class rdf:ID="ProcessIntegration">
  <rdfs:subClassOf rdf:resource="#EngineeringInformationManagement"/>
 </owl:Class>
```

Explanation

We can see that the base class of Process Integration is the inherited from Process Integration class. Thus creating base classes of Process Integration and Information Definition help us creating their instances for individual instances, having inherited property, which is unique for them.

Communication

This includes coming up with the channel to build smooth, error-free, quick and reliable communication between different parties, within or outside organization

OWL Function

```
<owl:Class rdf:ID="Communication">
  <rdfs:subClassOf rdf:resource="#EngineeringInformationManagement"/>
 </owl:Class>
```

Explanation

We can see that the base class of Communication is the inherited from Communication class. Thus creating base classes of Communication and InformationDefinition help us creating their instances for individual instances, having inherited property, which is unique for them.

4.5 Summary

Semantic systems are now been accepted by many companies and they are of the view that the systems could communicate with each other (and people). The companies are also eager to recognize the inevitable addition of ontologies or glossaries in their systems to carry out different activities in order to describe languages. Also Avoids misunderstanding: by providing a clear, accessible and agreed set of terms and definitions as the trusted source, discussions and misunderstandings can easily be resolved, long before theses errors and their corresponding costs, are hard coded in an IT implementation.

Chapter 5 Evaluation of Alignment (Traditional Methods)

5.1 Introduction

In earlier documents, we have discussed how product/service introducers gain much more capital by operating their organizations through web-services or by utilizing new innovations or improved technologies to up-lifting the market for better living and luxurious life. Today, it is though easy to get into the market; however problems lie within their sustainability. For that we earlier proposed Ontologies of structure of Organization and engineering its information management in semantic world.

However, to have interoperability between the defined objects / instances, attributes or properties in OWL ontologies, we have to have ontologies that we can reutilize, either for exchange of messages between two different ontologies, or making unique dictionary to reutilize attributes and properties of same object, thus garbage classes can be declared obsolete.

We will be discussing how two proposed ontologies are developed, efforts for designing the ontologies, different traditional methods to align two ontologies that are proposed by different researchers and the possible align-able and un-align-able classes exist in the structure. Further, we will propose our way of deploying ontologies for better heterogeneous objects to make the effective interoperability of classes.

5.2 Development of Ontologies

5.2.1 Organization Structure

Industry-Knowledge

Industry-Knowledge is prerequisite to design any structure of an organization. For business needs, we have to cover different aspects. We have different functions which include Financials, Legal, Resources, Maintenance, and Production etc. However, it is not a good idea to have a flat hierarchy which is difficult to manage for any business. It should be well structured. Though the communication will get slow, but due to utilization of recent technologies, this is not an issue. The other drawback is there are a high-dependency on Jack-of-All people, being flat hierarchy, while in structural; there are specialized people for each function.

With the aforementioned idea of having a structural organization, we can group the similar functions having same attributes. We segregate them as **Internal** and **External**, being top level node, further divide the concept to main category, include Financial, Ethics, Legal and Infrastructure.

This research define organization as follows:

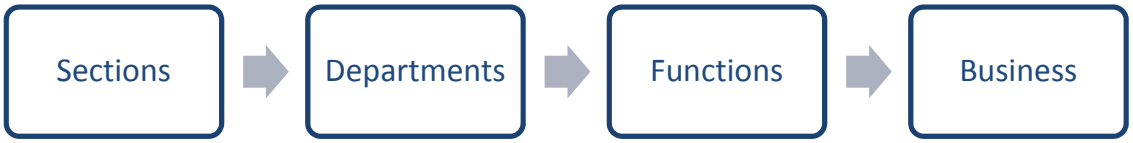


Figure 5.1 Organization defined

Derivation

We earlier defined concepts of individual classes of what their functions are, how they are interrelated and what benefit it will be having from by being utilized within the ontology. However, we have not defined of the concept behind how the ontology is derived. Here we will discuss the same in detail, while aligning the broad-level hierarchy with its commonly published ontology in following sections.

We know, with the industry's best practice that

> Broadly, the organization deals with either internal communication or external communication. Internally, unlike traditional hierarchy, we have functions of procurement, logistics, operations, financials, human resource and information technology. However, we have proposed hierarchy by utilizing the cost management for effective internal controlling and monitoring. Thus, only Financials, Management, and Information Technology should be the main functions as all communication within the organization is done by means of utilizing latest communication model, Financials provide funding to each and every department, and thus the departments can well be controlled through financing or proper cost management. Management is to overlook the IT and Financials and take relevant decision for more productivityExternally, the organization has to deal with their vendors, suppliers and customers. While it should comply with EHS policies, Welfare system, Legal affairs and Human Resource Management etc.

That why organizations should be defined in an aforementioned way that will not only help them in effectively operating, monitoring and controlling but also ethical issues, problems related to satisfaction of employees and other moral values will be addressed by not providing the leading chair to the core

business, rather core should concentrate on productivity being advised by management, with provided funding of Financials and state-of-the-art technology by IT.

5.2.2 Engineering Information Management System

Introduction

To achieve automation and subsequently maintaining repository of internal information, different requirements of an organization must have been evaluated. For that, we need to go in depth in each of its functions. This is possible if you are familiar with the business functions while equally competent with core IT, familiar with its functions. The same can be outsourced to knowledge engineers, who are actually the domain expert and can better provide services. This is required to better manage their IT infrastructure, at reduced recurring and maintenance cost, as well as aware of what their functions are, for what are they responsible, and what services will be provided to users. The Service Level Agreement may then be incorporated to automate the workflow process, etc.

Engineering IMS

As mention before to start with engineering an Information Management functions in an organization, it should be clearly defined of what the information should be circulated, maintained and are of decision-making nature. How fast it should be reached to its destination, its authenticity, reach ability and its maintenance for future references and audit purpose. For that, knowledge engineering is a must.

The second comes the integration of different system, how data will flow, i.e. implementations of workflows, definition of business processes and its adaption from industry's best practice. That also includes the selection of most suitable enterprise resource planning (ERP) systems, which may end-up in automating and integrating business processes of entire organization.

Derivation

We earlier defined concepts of individual classes of what their functions are, how they are interrelated and what benefit it will be having from by being utilized within the ontology. However, we have not defined of the concept behind how the ontology is derived. Here we will discuss the same in detail, while aligning the broad-level hierarchy with its commonly published ontology in following sections.

We know, with the industry's best practice that

Conceptually, the Information Management systems are based on the functions of how the data will be maintained, how they will communicate, which systems will be utilized and what integration procedures and system will be functional. Thus, these broad categories in combination make an environment for better information transfers. We therefore defined the aforementioned points as being the top layer, when Information Management is to be deployed in an organization

In an organization, the most important that should be authentic, reliable and part of decision-making is communicating message. IT plays important role for that, being a top-level function.

To have effective communication, there is a need for Integration of different deployed Systems, so that an error-free message can be communicated and translated.

The other factor is the medium through which the message is to be delivered, which should be a secured-one. Any leakage may have negative impact on the organization or it could be as severe as collapse of the entire structure

The other top-level factor is definition of repository to maintain entire communicated data so have to better perform analysis, or providing functionality to be referred anytime. The same will help organizations to get to know of their past information, the decisions they took and the outcome of what they get etc.

Thus, in combination, these functions can make an effective information management so as to assist in different operations of an organization and take valuable decisions for better outcome.

5.3 Evaluating Ontologies

There are numerous ways to assign efforts for each one, however we can assign effort = 1 to every leaf-level node and sum-up the top node to calculate effort of ontology

5.3.1 Organization Structure

Class-Level-1	Class-Level-2	Class-Level-3	Effort
Financials			26
	Management		14
		Productivity	7
		Facilities	1
		Licensing	1

	HR	4
	IPR	1
Sales		7
Product/Service		5
Infrastructural		3
Furniture		1
Office		1
Utilities		1
Ethical		1
Legal		3
HSE		1
Accounting		1
HR		1

Table 2.1: Ontology Evaluation

From the sub-level defined hierarchies, we can go with the effort-evaluation method of assigning equal weight to each of the node and apply bottom-up approach to sum the weights to its parent node, thus evaluating over-all effort of the ontology. By summing each leaf-level node and assigned the summed-value to its branch, and adding each weight of branches to evaluate the entire hierarchy, we have total worth of the Ontology equals 33.

5.3.2 Engineering Information Management System

Class-Level-1	Class-Level-2	Class-Level-3	Effort
Process Integration			1
System Integration			1
Information Integration			30
	Definition		8
		Product	4
		Structure	3
		Association	1
	Management		22
		Tracking	3
		Sharing	9
		Change	1
		Storage	7
		Distribution	2
Communication			1

Table 5.2: Ontology Evaluation II

From the sub-level defined hierarchies, we can go with the effort-evaluation method of assigning equal weight to each of the node and apply bottom-up approach to sum the weights to its parent node, thus evaluating over-all effort of the ontology. By summing each leaf-level node and assigned the summed-value to its branch, and adding each weight of branches to evaluate the entire hierarchy, we have total worth of the Ontology equals 33.

From the above two ontologies, they are same as per the derived efforts which are 33 in total.

5.4 Alignment of Ontologies

5.4.1 Organization Structure

Introduction

Alignment is done to come up with the ontologies that can be reutilized by others so the interoperability of the classes, objects, instances and attributes can be achieved. This helps not only the development of

referential architecture so one change can change the entire class at all instances at once, but also tight integration between the different ontologies so they can interconnect with each other for effective communication between the objects.

As provided in the report of life-cycle of aligning ontology, the details of which are as follows:

Apply algorithm to match ontologies
Save it to storage
Fetch through its ID
Fetch metadata of the same
Sort out applicable alignment
Search for different alignments from repository
Format and review alignment
Apply transformation, translation and generate code to implement the same
Search for similar ontology

There is no direct way to simply apply algorithm on ontologies and come up with the ontology alignment. Following are the traditional techniques for alignment

Terminology Technique

This technique caters to the alignment through the terminology or syntax of the class. The entire ontology is traversed, and each class is checked for its recurrence. The runtime is therefore O (n^2). It's a time consuming and might not align the ontology, as not all syntax can simply be mapped

Structural Technique

This technique deals with matching structure of each class in Ontology. This means that attributes of each class are matched, that may result in the similarity of 1 or more than 1 attributes to be matched. Thus finding similarities and coming up with the ideal similar class generalizes the class. At the end, it results in aligned ontology.

Extensional Technique

This technique deals in aligning the classes in ontologies, depending on their instances. So, the similar instances can be compared and if found with the same value, their classes are made similar. This results in aligned ontology, which is better than the structural technique.

Semantic Technique

It can further be elaborated in a way that the first stage should be testing the class on syntax basis and then comes to the local semantics of particular class. If it doesn't match, try to find the class in global semantics. This will apply interoperability on the defined classes. For being effective, reference of each syntax and its semantics should be followed by same repository (5). Thus, proper Descriptive Language is mandatory for such technique.

Definition

Let us start with the developing of different classes, while declaring the attributes so as to apply traditional methods to align the ontology. We will cover few classes which are Productivity and its sub-classes, Human Resource and its sub-classes, and Sales with its sub-classes. They all are internal classes and comes under Financial function. Further, we have classified each node as follows

Class

Sub-Class

Attributes

Productivity

Dashboards

Key Performance Indicators

Graphs

Charts

Tables

Futuristic Values

Analytics

Bar Charts

Line Charts

Tables

Reporting

Headers/Footers

Tables

Trend

Line Charts

Futuristic Values

BPM

Key Performance Indicators

Process Definition

Linkages

Operations

Materials

Human Resource

Training

Nature: Training, Seminar, Business Visit

Type: Foreign, Local, In-House

Process Definition

Key Performance Indicators

Reports

Recruitment

Job Specification

Position

Package (Budget)

Interviewers

Domain

Process Definition

Contracts

Key Performance Indicators

Reports

Compensation

Budget

Allowances: Traveling, Rent, Utilities, Accommodation

Process Definition

Key Performance Indicators

Reports

Communication

System

Route Through: List if Authorities

Process Definition

Key Performance Indicators

Reports

Distribution

Partners

No of Partners

Name of Partner

Rate

Availability

Reach: List of Areas

Workers

No of Available Mediums

No. of Workers

Cost of Distribution

Reach: List of Areas

Processing

Raw Material

Name

Quantity

Total Cost

Value Addition

Name

Quantity

Total Cost

	Advertising
Servicing Customers	Cost for Publicity
Feedback	Cost for Ad Production
Medium	Cost for Banners
Message	Cost Benefit Analysis
Customer	Market Response
Assigned Resource	Customer Base
	Market Requirement
	Need Analytics
Selling	Customer Response
Research	Purchasing Trend
Market Reach	
Cost of Selling	
Pricing	Market Research
Demand Analysis	Needs
Customers	Identify Needs
Potential Area	Check Customer Base
Customer Demand Price	Conduct Survey
Customer Needs	Conclude
Discounts	Provide Valuation
Marketing	

Alignment

From existing research method in literature we can assign customers, suppliers, employees, sellers, purchasers, partners, vendors and position defined above in different classes as an adopted framework, either FOAF (Friend of a Friend) or vCard. Though, MAFRA can be used as well, being a mapping framework.

We cannot directly apply Ontology Mapping Enhancer or OMIEN, as there isn't a differences in between the classes to utilize probabilistic values. Thus we can avoid the same.

Following are the applied methods of our ontology

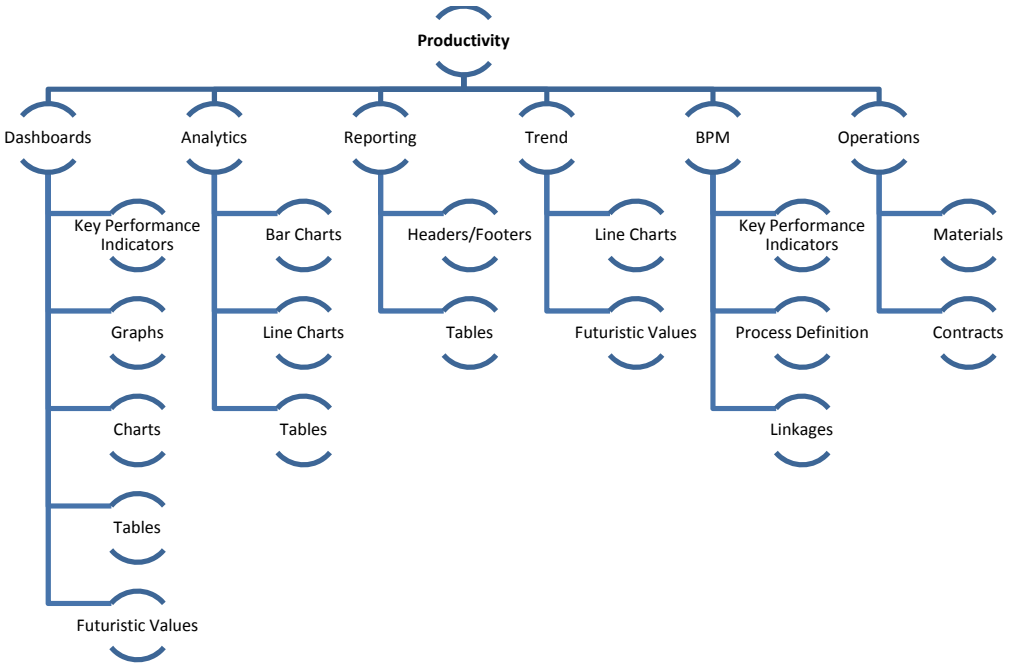


Figure 5.2: Productivity Class

Attributes	Terminological	Structural	Extensional	Semantically
Key Performance Indicator	Dashboards, BPM	Dashboards, BPM		Dashboards, BPM
Graphs		Bar Charts, Line Charts, Graphs and Charts all have same structure. Therefore; Trends, Analytics, Dashboards		Bar Charts, Line Charts, Graphs, Tables and Charts all have same semantics Therefore; Analytics, Reporting, Trend, Dashboards
Charts		VOID		VOID

Tables	Reporting, Analytics, Dashboards			VOID
Futuristic Value	Dashboards, Trends			VOID
Bar Charts		VOID		VOID
Line Charts	Analytics, Trends	VOID		VOID
Process Definition				Process Definition itself has Process Gaps, therefore Linkages can be VOID
Header/Footer				
Linkages				VOID
Material				
Contracts				

Table 5.3: Performance Class Alignment through Traditional Method

From above, we can clearly see that

Terminologically, following are the same in the mentioned classes:

KPIs are in Dashboards and BPM classes

Tables are within the Reporting, Analytics and Dashboards classes,

Futuristic Values are in Dashboards and Trends classes

Line Charts are in Analytics and Trends classes

Therefore, classes will get limited to “Analytics” only, while all can be classified as attribute “Type”. We can segregate the classifications through defined attribute of single class.

Structure-wise, Dashboads and BPM classes have the same attribute of KPI, while Graph have same structure as of Bar Charts, Line Charts, Graphs and Charts. Therefore, Trends, Analytics and Dashboards are such classes that have the same attribute. As structurally they are same, Therefore we can declare single attribute of Graphs, while assign it to each mentioned class, to make it as interoperable class.

The same goes for Semantic method, which not only make Dashboard, Trend, and Analytics and Reporting as one type of Analytics, while making VOID semantically same attributes. Each class can have attribute “Type”, to classify the nature of Graph.

Further, we have Process Definition as an attribute in Process Gaps and Linkages, these all can be made single class, while declaring attribute of “Classification”, to segregate classes as per their nature.

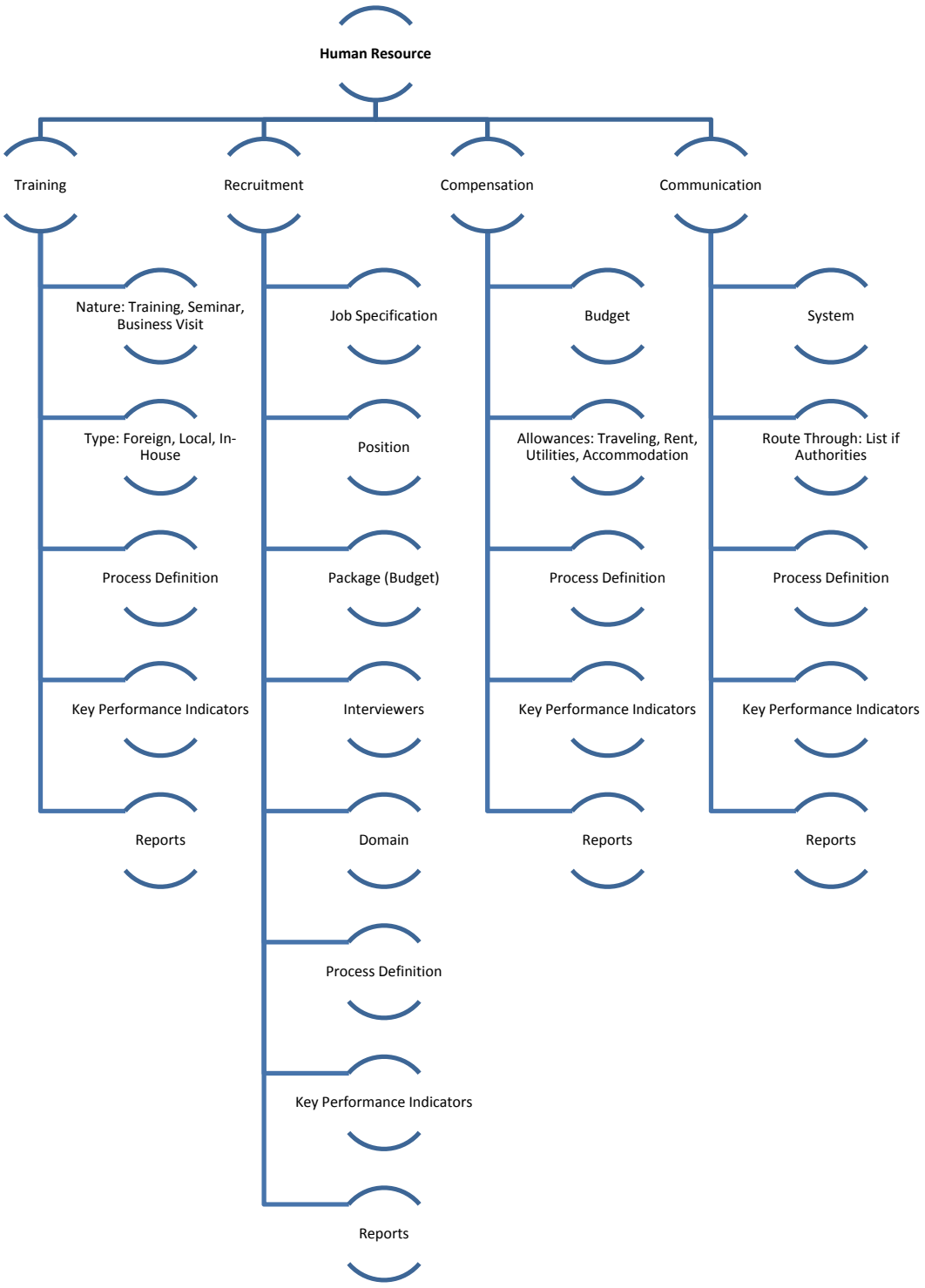


Figure 5.3 : Human Resource Class

Attributes	Terminological	Structural	Extensional	Semantically
Nature				
Type				
Process Definition	Training, Compensation, Communication, Recruitment	Job Specification derives from Process Definition, therefore according to structure, they are same. Therefore; Recruitment can have this class with two instances of Process Definition		
KPI	Training, Compensation, Communication, Recruitment			Training, Compensation, Communication, Recruitment
Reports	Training, Compensation, Communication, Recruitment			Training, Compensation, Communication, Recruitment
Job Specification		VOID		
Position				Position, Routers, Interviewer, Domain Expert are semantically same Therefore; Recruitment and Training can have instances of same class
Package				Package, Budget and Allowances are of same semantics, therefore; Compensation and Recruitment can have

				instances of same class with additional attribute of classification
Interviewer				VOID
Domain Expert				VOID
Process				
Allowance				
System				
Routers				VOID

Table 5.4: HR Class Alignment through Traditional Method

Process Definition is part of every process, which is provided in Training, Compensation, Communication and Recruitment classes. As terminologically Process Definition is same, we can Use single class of Process and in process definition, we can define the entire process, while organization-unit can be assigned at the same level.

Structural-wise, as Job Specification is derived from Process Definition itself, they have the same structure. Thus it can be replaced with Process Definition, keeping Job Specification as VOID as its attribute.

Semantically, Reports, KPIs and Dashboards are semantically same. Therefore these can be replaced with single keyword Analytics. Position, Routers, Domain Expert and Interviewer are semantically same. Therefore, They can be assigned as one attribute Person, while the same will make the unique attribute of each class, making it interoperable.

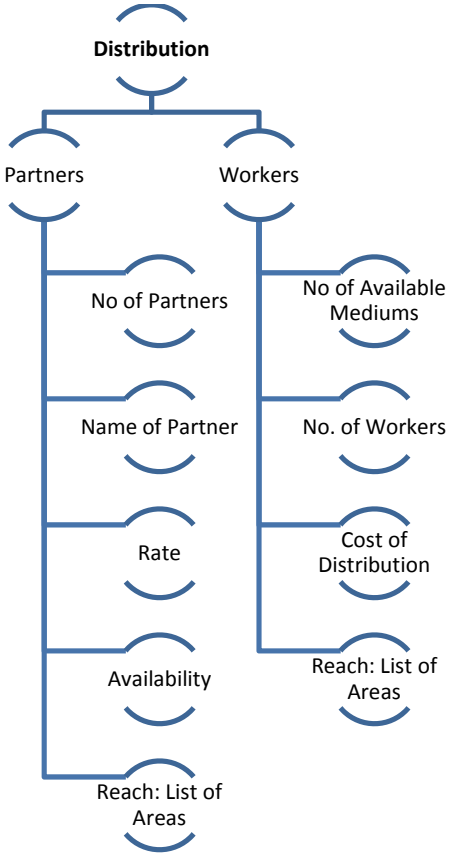


Figure 5.4 Distribution Class

Attributes	Terminological	Structural	Extensional	Semantically
No. of Partners		No. of Workers and Partners are based on numerical value (integer), therefore; Workers, Partners		Rate and Cost of Distribution; No. of Workers and Partners; Reach and Availability; Therefore; Partners and Workers
Partner Name				
Rate			Partners and Workers	Rate and Cost of Distribution; No. of Workers and Partners; Reach and Availability; Therefore; Partners and Workers
Availability				
Reach		VOID		Rate and Cost of Distribution; No. of Workers and Partners; Reach and Availability; Therefore; Partners and Workers
Mediums				
No. of Workers		VOID		VOID
Cost of Distribution				VOID
Reach	Partners, Workers	VOID		

Table 5.5: Distribution Class Alignment through Traditional Method

Through Structural-wise method of alignment, no. of workers as well as no. of partners have same numeric value. Hence they are structurally same. Therefore, we can have single attribute of Workers and Partners as Stakeholders so as to make them unique.

Further, if we semantically analyze above table, we have attributes of No. of Partners and Workers as of same semantics, while Reach and Availability in Area have same semantics. Cost of distribution and Rate, which includes cost of distribution are having the relationship of "Cost being Hold". Therefore, semantically, Cost of Distribution is part of Rate as well. Therefore, they can be replaced with additional attribute to classify the cost. Therefore classes can be limited to Rate, Area and Nos. only while making them unique for better interoperability

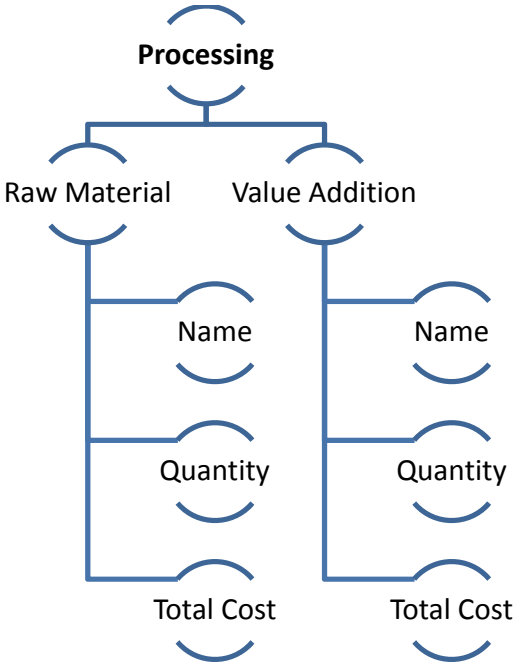


Figure 5.5 Processing Class

Attributes	Terminological	Structural	Extensional	Semantically
Name	Raw Material and Value Addition has same attributes	Same	N/A	Same
Quantity	Raw Material and Value Addition has same attributes	Same		Same
Total Cost	Raw Material and Value Addition has same attributes	Same		Same

Table 3: Processing Class Alignment through Traditional Method

We can see from above that all traditional methods are applied to above classes under Processing. Terminologically, in both the hierarchies, while semantically, as their structure is same, the classes are same as well. By, semantic method, we can see that Name, Quantity and Total Cost are all semantically same in Raw Material as well as Value Addition. Therefore, Raw Material and Value Addition classes can be incorporated with additional field, ”Material Type”, then they can be the instances of same class. Thus, they both can be combine into one to make the class interoperable in the semantic world.

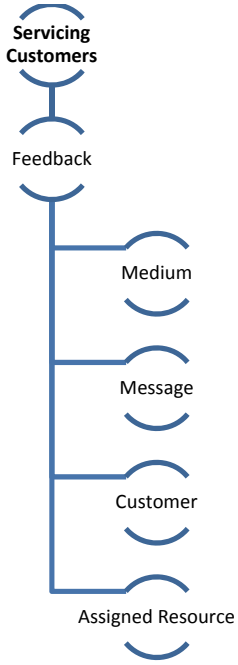


Figure 5.6 : Serving Customers Class

The above can be compared with Global class, as no similar class is available in the currently defined classed

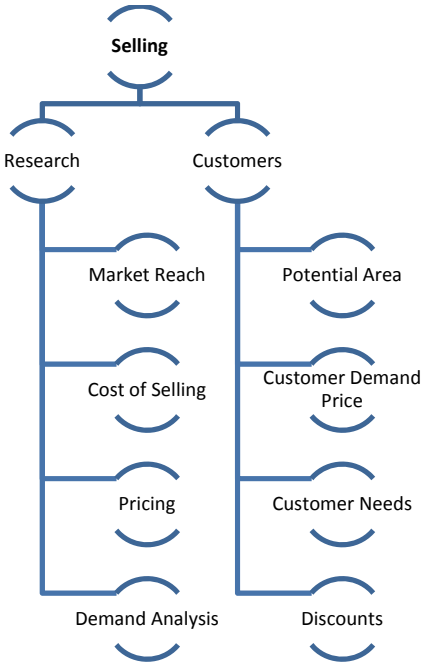


Figure 5.7 Selling Class

Attributes	Terminological	Structural	Extensional	Semantically
Market Reach				Potential Area has same semantics, therefore; Customer Research
Potential Area				VOID
Cost of Selling		Price, Discount and Cost have same structure, therefore: Customer and Research		Price, Cost and Discount have same semantics therefore; Research Customer
Price		VOID		VOID
Demand Analysis		Demand Analysis have same structure as of Customer Needs, therefore; Customer Research		Demand Analysis is semantically same as Customer Research, therefore; Customer Research
Customer Demand Price				VOID
Customer Needs		VOID		VOID
Discounts		VOID		VOID

Table 5.7: Selling Class Alignment through Traditional Method

Terminologically, we can clearly see that there isn't any class or attribute that are similar. Therefore the method is inapplicable. However, semantically, Price, cost and discount are the attributes having Amount type as the only field, therefore all these attributes can be made a single attribute while its

classification attribute can segregate it. Demand Analysis and Customer Needs are having the same structure, therefore they can be one attributed of Product Valuable. This makes the Customer and Research classes more simplified.

Semantically, they have attributes of Cost, Price and Discount as same, while Discount therefore the same can be added in both Customer and Research classes to make them a simplified and interoperable. The same goes for Customer Analysis and Demand Analysis, have same semantics. Therefore, they can also be classified as one attribute Product Analysis, to make the Customer and Research classes interoperable, making them integrated with each other with same attribute so no ETL processing is required to tightly integrate them.

Un-aligned Classes

From above, we can clearly see that except sub-classes of Processing, none can be directly aligned with each other. However, instances of many classes can have additional attribute of "Type" to make a same data-model for interoperability. However, following main classes are not even partially be aligned or aforementioned method cannot directly be applied

Productivity: Operations, Mining

Operations:

Productivity operations are referred to as monitoring the performance of the organization by keeping an eye of the operations. This process include not only monitoring the processes but also taking prompt decisions as to gain more and more capital. For operations, we have defined links to external bodies, acquisition of lands and financial assets etc. This, therefore include Business Development as well as Process Engineering as an attribute.

Mining:

Mining is to actually apply statistical algorithm on the past data, resided in data warehouse to extract meaningful pattern of information to take decisions of where to invest further according to the predicted revenue or net income. This includes the attributes of characteristic, key-figures and dimension, while it requires data-warehouse as a database repository as well:

Traditional Methods

Terminologically, Process Engineering, Business Development Techniques, Data Warehouse, Data Characteristics, Data Attributes, Statistical Algorithms and Data Dimensions are not similar or same to any attribute of any of the other class. Therefore we can say that they are not terminologically able to align.

Structurally, as Process Engineering includes development of the process, Business Development Techniques include techniques to expand organization horizontally, Data Warehouse which is the archive data of many years, Data Characteristics, Data Attributes, Statistical Algorithms and Data Dimensions are characteristics of data analysis but are not similar or same to any attribute of any of the other class. Therefore we can say that they are not structurally able to align.

Extensionally, Process Engineering, Business Development Techniques, Data Warehouse, Data Characteristics, Data Attributes, Statistical Algorithms and Data Dimensions have not instances of which upper and lower nodes are same to any attribute of any of the other class. Therefore we can say that they are not extensionally able to align.

Semantically, Process Engineering, Business Development Techniques, Data Warehouse, Data Characteristics, Data Attributes, Statistical Algorithms and Data Dimensions are not in any way having same meaning with that of any other attributes of class. Therefore we can say that they are not semantically able to align.

Servicing Customer: Advertising, Market Response

Advertising

Advertising is actually to make the customer aware of the product or service and let it be marketed in the specific area, by targeting specific category of groups. Therefore, it has the attributes of Clients, Method of Advertising (i.e. Printed, Online, TV, Radio, Banners etc.).

Market Response

Market Response is actually to check how the market responded after the roll-out of service/product in particular area. This means to check the customers intention to buy next time. This helps in identifying the pattern of why the people have bought, what is the demand rate and how much should the

production be for next time, while if any value-addition is required for the next release. Therefore, it has the attributes of Area, Value of Product/Service, Response and Demand

Traditional Methods:

Terminologically, we can see that the there isn't any attribute that are similar or close to similar. Area, Value of Product, Response and Demand, none is equivalent to any attribute in Ontology. Similarly, Clients and Method of Advertising are also not available in entire Ontology. Therefore, they are terminologically exclusive attributes.

In terms of structural comparison, we have different structure of Area, Value of Product/Service, Response and Demand, while we have different structures of Clients and Method of Advertising. Therefore, structurally, they are different with each other and therefore the method cannot be applied to align the ontology.

Extensionally, as none of the class has instances, that is no parent and child nodes exists of particular nodes of Area, Value of Product/Service, Response and Demand, Clients and Method of Advertising. We can conclude that this method to align the classes in ontology is inapplicable.

Semantically, as Area, Value of Product/Service, Response and Demand, Clients and Method of Advertising which are attributes of two classes, are all different in their meanings, and have different purpose for different objects, we can state that the method is inappropriate to align the classes in ontology.

5.4.2 Engineering Information Management System

Definition

Let us start with the developing of different classes, while declaring the attributes so as to apply traditional methods to align the ontology. We will cover few classes which are Process Integration, System Integration, and Information Integration with its sub-classes. Further, we have classified each node as follows

Class

Sub-Class

Attributes

Process Integration
Process Definition
Activities
Actors
Systems

Process Monitoring
Key Performance Indicators
Process Gaps

Process Controlling
Documentation
Audit

System Integration
System
Database
Application

Network
Integration Medium
Security Policy
Router
Protocol

Information Definition
Product
Name
Component
Relationship
Storage Location
Structure

Information
Association
Storage

Process
Steps
Activities
Actors

Information Management
Distribution
Import
Transmit

Storage
Check In
Move
Put Away
Append

Change
Update

Sharing
Copy
Reference
Check Out
Get

Tracking
Check Status
Check Flow

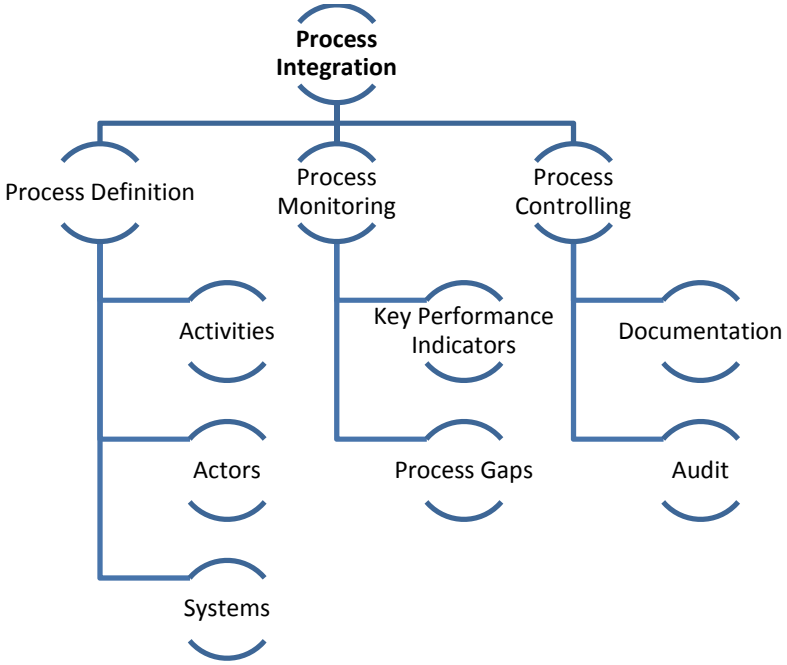


Figure 5.8 Process Integration Class

Attributes	Terminological	Structural	Extensional	Semantically
Activities		Activities, Process Gaps and Documentation are of same structure, therefore we can create instances of their classes	Activities will have instance of Process Gaps, Systems and Actors, therefore, they can become sub-classes	Systems and Activities have same semantics. Actors can be Systems, semantically

Actors		Systems will also have the same attribute	VOID	VOID
Systems		VOID	VOID	VOID
Process Gaps		VOID		
KPIs				
Documentation		Documentation and Audit Report will have same structure		
Audit		VOID		

Table 4: Process Integration Class Alignment through Traditional Method

Structurally, Activities, Process Gaps and Documentation are the attributes that have the same structure. Therefore we can create instances of these classes as same. This makes the class interoperable for better communication between each other. Also, we can see that Audit Report is a type of documentation that have the same semantics, therefore, it can be replaced with the same attribute as well, that makes unique, the junk class and attributes can therefore be void.

Extensionally, As Activities will have instance of Process Gaps, Systems and Actors, therefore, they can become sub-classes of the mentioned classes. This makes the entire hierarchy more simplified and interoperable in a way to let them integrate with each other for effective communication.

Semantically, Systems and Activities have same meanings. Systems are the combination of processes while activities are also the combination of processes. Therefore, we can have same attribute of class Process Definition, thus making it more simplified.

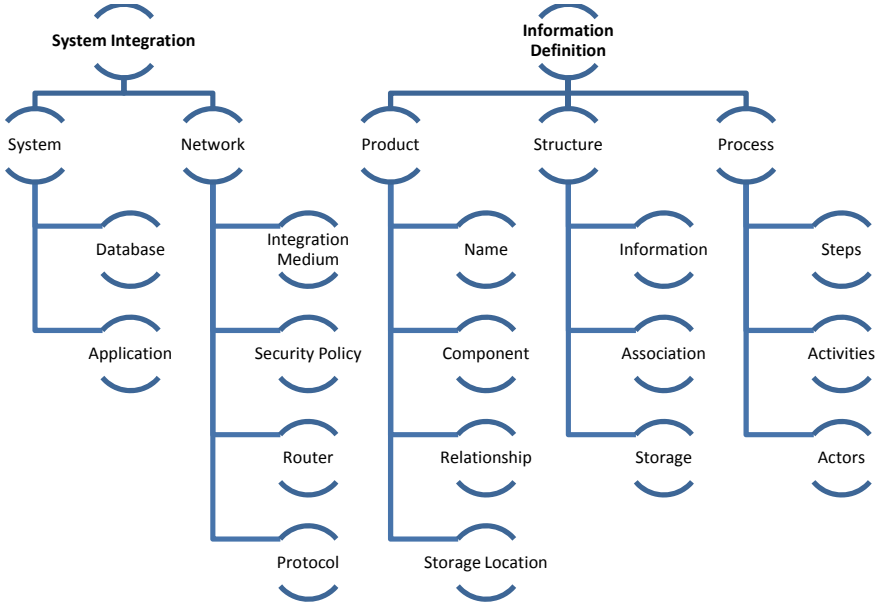


Figure 5.9 System Integration Class and Information Definition Class

Attributes	Terminological	Structural	Extensional	Semantically
Database			Instance of System and Product will be same	Storage, Association and Information are sub-components of any database, therefore Structure class can be an instance.
Application		Structure is same as of Component	SAME	Product would have same attributes semantically.
Router				
Protocol		VOID		Security Policy are embedded
Security Policy				VOID
Integration Medium		Protocol have same structure		
Name			VOID	
Component		VOID	VOID	
Relationship		Association has	VOID	

		same structure	
Storage Location		Storage and Storage Location have same structure	VOID
Information		VOID	
Association		VOID	
Storage		VOID	
Steps			
Activities			
Actors			

Table 5.9: System Integration Class Alignment through Traditional Method

Terminologically, all are different; therefore we cannot effectively get aligned ontology through this method. However, if we apply semantic method on ontology, we get the Structure and Component are of same structure, Protocol and Integration medium are of same structure, while Relationship is same as Structure, thus Component is also same. Storage Location and Storage are same as points to the memory attribute. Thus, unique attributes i.e. Component, Protocol and Memory can be utilized to make related attributes same (i.e. 8 attributes), This will make the class more simplified.

Extensionally, we have same instances of Product and System, therefore we can classify these as single, i.e. Product only. However, we can clearly see that it is not the effective method to align ontology in our case as very few instances are defined.

Semantically, Storage, Association and Information are all of the types of functions, that should be defined under the class Database. Unlike what we mentioned above in structural alignment method, Storage and Storage Location are same, here we can see that these are two different things. Similarly, Application and Product Attributes are of same semantics in Information System domain, therefore they can be replaced with Application, while Security Policy and Protocol are of same semantics, Therefore, only Protocol can be declared to simplify the ontology.

Hence, we can clearly see that in combination of applying semantic and structural methods of alignment on our ontology, we can come up with the aligned ontology to approximate maximum level. However,

we can see that there exists the classes which are not being aligned through traditional methods, mentioned in next section.

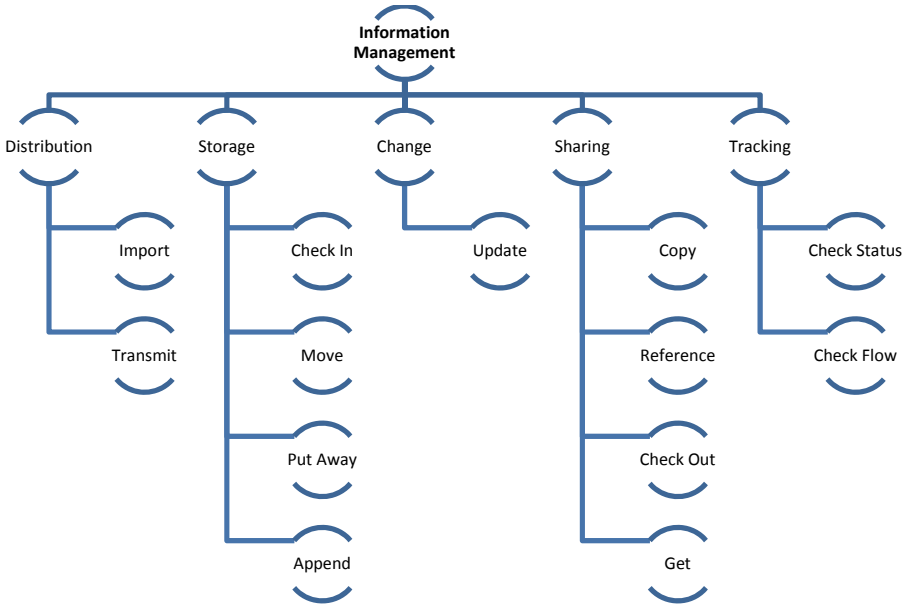


Figure 5.10 Information Management Class

Attributes	Terminological	Structural	Extensional	Semantically
Import		Import and Copy are of same structure		Get, Check In, Copy and Append are of same nature, there can be created as instances of single class with additional

			property
Transmit		Transmit and Move are of same structure	Move, Put Away, Update, Reference and Flow are of same semantics
CheckIn			VOID
Move		VOID	VOID
Put Away			VOID
Append		VOID	VOID
Update			VOID
CheckOut			VOID
Reference			VOID
Get			VOID
Check Flow			VOID
Check Status			VOID

Table 5.10: Information Management Class Alignment through Traditional Method

Terminologically and Extensionally, we cannot align this sub-ontology of class as no attributes are of same nature while no instances are of same hierarchy.

Structurally, Import and Copy attributes are of same structure therefore can be replaced with Copy only. While Transmit and Move are of same structure, therefore can be termed as Move, Thus making the class a bit classified by eliminating/replacing two attributes.

Semantically, Get, Check In, Copy and Append are the attributes that have the same function. They therefore can be used as single attribute to make 3 attributes as 1 to make the ontology simplified. Further, Move, Put Away, Update, Reference and Flow are also the attributes of same semantics, therefore can be replaced with Update attribute to simply class ontology by replacing 4 attributes with 1 attribute.

Un-aligned Classes

Numerous classes can be mapped with their globally published classes, however within the same, there are very few which can directly be mapped to align the ontology. However, the completely unmapped classes are as follows:

System Integration

System

By System, we mean to have instances of different applications, software and hardware in place, so as to integrate them and make the data interoperable. The attributes include Application, Software, Hardware and Databases

Network

It's the class to represent connectivity in between systems. This enables developing communication channel in between different systems so as to create smooth transition of the messages, that can be signals from the interconnected systems or messages sent through systems by means of application. The attribute thus include Wires, Protocol, Servers and Infrastructure.

Traditional Methods

As terminologically, we don't have any term within attributes in Application, Software, Hardware and Databases from System and Wires, Protocol, Servers and Infrastructure attributes from Network that are similar or same, the method is inappropriate for aligning the ontology

Structurally, we have different architecture of Application, Software, Hardware and Databases from System and Wires, Protocol, Servers and Infrastructure attributes from Network. Therefore we cannot relate the classes and make them unique. The method is therefore not applicable for aligning the ontology

Extensionally, as there aren't any instances in parent and child node exists of any of the attribute in Application, Software, Hardware and Databases from System and Wires, Protocol, Servers and Infrastructure attributes from Network, the method is inapplicable.

Also, as semantics of the attributes Application, Software, Hardware and Databases from System and Wires, Protocol, Servers and Infrastructure attributes from Network are all different, as each is

representing different meaning, the method is therefore not applicable in this case to align the ontology.

Information Management

Storage

To save data in a persistent area, while retrieving it for future utilization, we have the class of storage, that have different attributes of where to store the data, in which format the data should be stored, what encryption method is to be used and in which type it should be saved. Therefore, having the attributes of Location, Format, Encryption, Data Type

Sharing

By sharing we mean to retrieve data and send or provide access to currently unassigned stakeholder that can either be any user, application or system. The attributes of which are Source, Destination, Access Rights, and Ability to Re-share.

Traditional Methods

Terminologically, the attributes of unaligned classes in our ontology have different attributes including Location, Format, Encryption, Data Type from Storage and Source, Destination, Access Rights, and Ability to Re-share from Sharing. As it can be seen that no one attribute is similar to any other attribute in entire ontology, therefore we can say that traditional methods are inapplicable.

Also, as attributes including Location, Format, Encryption, Data Type from Storage and Source, Destination, Access Rights, and Ability to Re-share from Sharing are different in structure, and that not a single is equivalent or similar to other attribute, we can conclude that structurally, they are different

Extensionally, as none of the class has instance of parent and child node, we cannot have any class exist to compare the instances. We can therefore conclude that extensional method is inapplicable.

Semantically, as attributes including Location, Format, Encryption, Data Type from Storage and Source, Destination, Access Rights, and Ability to Re-share from Sharing are having different meaning in the same context, while they are used for different purposes, we can say that the class remains unaligned through semantic method.

5.5 Conclusion

Today, for every organization, information management plays an important role in keeping the data to perform different analysis, perform operations in relatively less time, external stakeholders can easily integrate to provide services in limited time, and communication within organization can be made effective. Earlier, we had defined both ontologies as separate entities and tried to explain in detail. However, we have seen that there isn't any class that can assists or enable direct integration within them. We found that the traditional methods did not able to make any sense in integrating Organization with Information System. Although Storage and Sharing in Information System are integral part of Organization, however with traditional, semantic, structural and extensional methods are inappropriate to align the ontology. We have covered the same in detail in earlier sections. We can conclude with the statement that there are traditional methods which can help us in aligning the ontology; however, there isn't a direct algorithm that can directly align ontologies. Also, extensive dictionary is required to be published for global accessibility with proper algorithm to manage, so alignment by any means can be possible. Major role is to develop ontologies carefully, that is to first check if the class exists and if not, then creating our own. However, a body can be made to let the customized class be accessible globally, so there could be a check-n-balance in adding new classes to library. Traditional methods are not much effective as they could result in transforming the meaningful ontology to non-meaningful one. Thus, could have a greater problem when achieving interoperability.

Chapter 6 Developing Compositional Alignment Method

6.1 Introduction

In this thesis has discussed the cumbersome of the process is, to apply all the traditional methods to come-up with the alignment. This may either lead to providing aligned ontology, or could lead to a situation where alignment may not be possible. This is because of unavailability of potential classes or objects that may not be aligned, from either any of the method.

In this Chapter, a new method is introduced ,and applying its different techniques methods on few of the classes, declared non-potential for alignment through traditional methods.

6.2 Background

There exist many systems which operates on Input->Process->Output. This could be in cycle in a way that Output of one system would be an Input for another system. The Process part transforms data into more appropriate information that either could be final result, or act like a simple Input for an effective Output. Many scientists worked and derived different algorithm on the basis of this simple fact.

While aligning single Ontology, it gets difficult to apply traditional methods to align the complex ontologies. There exists very few ontologies that are having very few classes. Majority, being published over the web, are documented extensively. As only the in-depth defined ontology can result in a heterogonous of the objects over the web, do tightly integrate different other systems. With the concept, we can design entire ontology, and bifurcate it according to small units or systems with the objective to make output of one system would act as a feeder class for other system, and provide input to let it get processed.

6.3 Rely/Guarantee

In the past, it was easy to make an output of a system as an input of another system. This is because all the programs being generated were actually based on sequential methods. That is, the systems always rely on the output of its dependent system. However, when it comes to parallel processing, more research was performed which results in providing solution as an integral way, which was previously being done independently.

Rely-Guarantee, or in other words Assumption/Commitment is a method which assumes that not only a component is verified by just satisfying all of its commitments, but also verifies all the assumptions, being exposed from the internal or external environment.

This can better be explained with the real-life example that if we want to have dinner, we can either go out to some restaurant, can cook at home, or can place order to get it delivered at the door-step. Now, the process of having dinner is dependent on either or the commitments from the mentioned processes. For first option, we need to drive way to the restaurant to reach the destination, so “Traveling” is the process which needs to be committed. Similarly, if we want to cook at home, we first need to go for “Shopping” to buy ingredients. The process commitment will enable us to cook or Initiate cooking process, so that dinner system can be practiced. The last option is to let the dish be delivered at home. For that, two commitments, a phone call and book and order while another process by the delivery-boy to bring it to your doorstep, should be committed so have dinner.

So, if all the assumptions from the environment are verified, while the commitments are fulfilled, the system can have its input to ignite the system. In other words, all pre-requisite systems should have commitments in order to provide feed to dependent system to let the process flow.

6.4 Rule of Thumb

Rely guarantee works on the rules that if the first state satisfies the initiating process and all the state being changed are relying (R), then every final state will satisfy the final process and every next state within the processing is guaranteed (G)

$R,G \vdash \{P\}\ C\ \{Q\}$

6.5 Architecting Rely/Guarantee

Any analyst can better come up with the systems and apply the assumption/guarantee method to align the ontology of that particular system. For that, the analyst first have to come up with the no. of systems exists within entire system. This means to, bifurcate the entire process into small systems which are dependent on each other, if related. The focus while deriving the sub-systems should be on how the system verifies the initial process, and all sub-systems do rely on prior system to come-up with the consequence.

The technique can be adopted to align the ontologies. That creates the possibility of aligning such ontologies that cannot be aligned with the traditional methods. We will further discuss different methods to align the ontologies in different ways. Further, with the methods, we can also automate the alignment process by developing a program that can align each ontology accordingly.

6.6 Methods

There exists different method for the Assumption/Commitment process which we are going to propose for ontology alignment. However, we will concentrate on following two methods only.

6.6.1 Parallel Rule

With parallel rule method, we mean to have different processes being run at the same time, i.e. simultaneously. We can better explain it with the following example.

Suppose we have to come up with the minimum no. within a provided list of nos., say 100. If we go with the sequential processing to identify minimum no. though an environment for parallel execution is available. We need to traverse entire hierarchy, the complexity of which will become O(n). However, with parallel rule, we can execute matching algorithm by splitting the number range in even and odd values. Each list will have 50 nos. Then we can execute our matching algorithm on both the list in parallel which results in half of the time as was done earlier. At final, we have two values of the two lists. We can therefore conclude that the smallest no. among two identified nos. is the lowest in the range.

So, IF

$R \cup G2, G1 \vdash \{P1\}\ C1\ \{Q1\}$

$R \cup G1, G2 \vdash \{P2\}\ C2\ \{Q2\}$

THEN

$R, G1 \cup G2 \vdash \{P1 \wedge P2\}\ C1 \parallel C2\ \{Q1 \wedge Q2\}$

Later, we will apply the Parallel Method to our modeled class to get its outcome and compare it with the other proposed method for Rely/Guarantee.

6.6.2 Rule of Consequences

In rule of consequences, we assume that if the dependent state satisfies with all the environment assumptions and the next state satisfies the same assumptions, the outcome will also guarantees to satisfy the all the assumptions.

Therefore, if Initial Process (P) satisfies all the assumptions (R), while the final process (Q) satisfies all the assumptions (R), which was earlier satisfied by P as well, and if the effect is guaranteed and is contained in (G), then

$R,G \vdash \{P\}\ c\ \{Q\}$

In other words, if all the processes are relying, and feeding input to other process which also satisfies the assumptions, than within the two processes, guarantee exists. The change states will be maintained in Guaranteed state (G)

With a real-life example, we can better explain the method. For instance, if we have a process to spend holidays (Process 0), we need to book the tickets (Process 1), need to purchase clothes (Process 2), and need to purchase fresh-food (Process 3). For Process 0, we first need to proceed for Process 1 as its outcome will inform either to trigger Process 2 and Process 3. This is the consequence-rule way of Assumption/Commitment. However, Process 3 and Process 2 can run in parallel as Process 3 can be triggered by ordering, while Process 2 is what we our-self have to do. So, in a same time, both the processes will be executed, thus being followed in parallel. However, this parallel execution is a consequence of Process 1. This means that Process 1 satisfies the assumptions, while Process 2 and 3 satisfies assumptions only if Process 1 satisfies, i.e. If tickets are booked, than the shopping and food purchase is executed, while if tickets are not booked, both the dependent processes will not be executed. This endorses applicability of Rely-Guarantee.

In other words, IF

$R \subseteq R'\ R',G' \vdash \{P\}\ C\ \{Q\}\ G' \subseteq G$

THEN

$R,G \vdash \{P\}\ C\ \{Q\}$

6.7 Algorithms of Ontology Alignment through Rely/Guarantee

6.7.1 Parallel Rules

We will run the algorithm of Ontology Alignment through Parallel Rule of Rely/Guarantee, where the hierarchy level will run in parallel

1) Process In-Parallel

a) Check the Input Parameters

b) Check the Output Parameters

2) On successful execution of (1a) and (1b), or simply (1), traverse each node and reconcile in parallel

as follows

a) Reconcile Input Parameter with (1a)

b) Reconcile Output Parameter with (1b)

3) IF (2) is successful, do the following in parallel

a) Save the Class as this is the Unique class, and therefore will be replaced by all similar classes,

thus aligning the Ontology effectively.

b) Goto (1)

ELSE STOP

6.7.2 Rule of Consequence

We will run the algorithm of Ontology Alignment through Consequence Rule of Rely/Guarantee, where the hierarchy level will run in parallel

1) Test if All Input Parameters are not Checked, If not GOTO (2) ELSE STOP

2) Check the Input Parameters

3) Traverse each node and reconcile with the current Input Parameter

4) IF reconciles any node

a) Check the Output Parameters

b) Traverse each node and reconcile with the current Output Parameter

c) IF reconciles any node

i) Save Class, as this can be replaced by similar classes, thus aligning the Ontology effectively.

ELSE GOTO (2)

ELSE (1)

6.8 Alignment of Ontologies

6.8.1 Organization Structure

Introduction

We earlier discussed the ontology in detail of what the organization structure should be. We then proposed the ontology to make in an entire interoperable on the web. We then made some alignments to discard the replicated classes and came up with a single class, while its instances should be utilized. We earlier did the alignment through traditional methods where attributes matters extensively. However, we had many of the classes which remained unaligned. For that, we can utilize our proposed way to align ontology using rely/guarantee.

Definition

Let us revisit already developed classes , while recalling the attributes so as to apply Rely/Guarantee method to align the ontology. We will cover few classes which are Productivity and its sub-classes, Human Resource and its sub-classes, and Sales with its sub-classes. They all are internal classes and comes under Financial function. We earlier aligned different classes through traditional methods, however following are the unaligned classes that we will be applying rely/guarantee to. Further, we have classified each node as follows

Background

We aligned ontology through traditional methods and able to streamline few of the classes. However there were numerous classes that we didn't able to align. Among them we will discuss the high level alignment at class-level while we will discuss attribute level alignment of classes at second level.

For that, we first need to identify the business processes within an organization to capture the data flow between different processes, sections and departments, as well as function.

Un-aligned Classes through Traditional Method

From traditional methods, we had seen that except sub-classes of Processing, none can be directly aligned with each other. However, instances of many classes can have additional attribute of "Type" to make a same data-model for interoperability. However, following main classes are not even partially be aligned or aforementioned method cannot directly be applied

Productivity: Operations, Mining

Servicing Customer: Advertising, Market Response

Alignment through Rely/Guarantee

Let's focus on the Productivity class, while restricting its access to other dependent classes, Marketing, Human Resource and Sales. Following are the hierarchies of our ontology

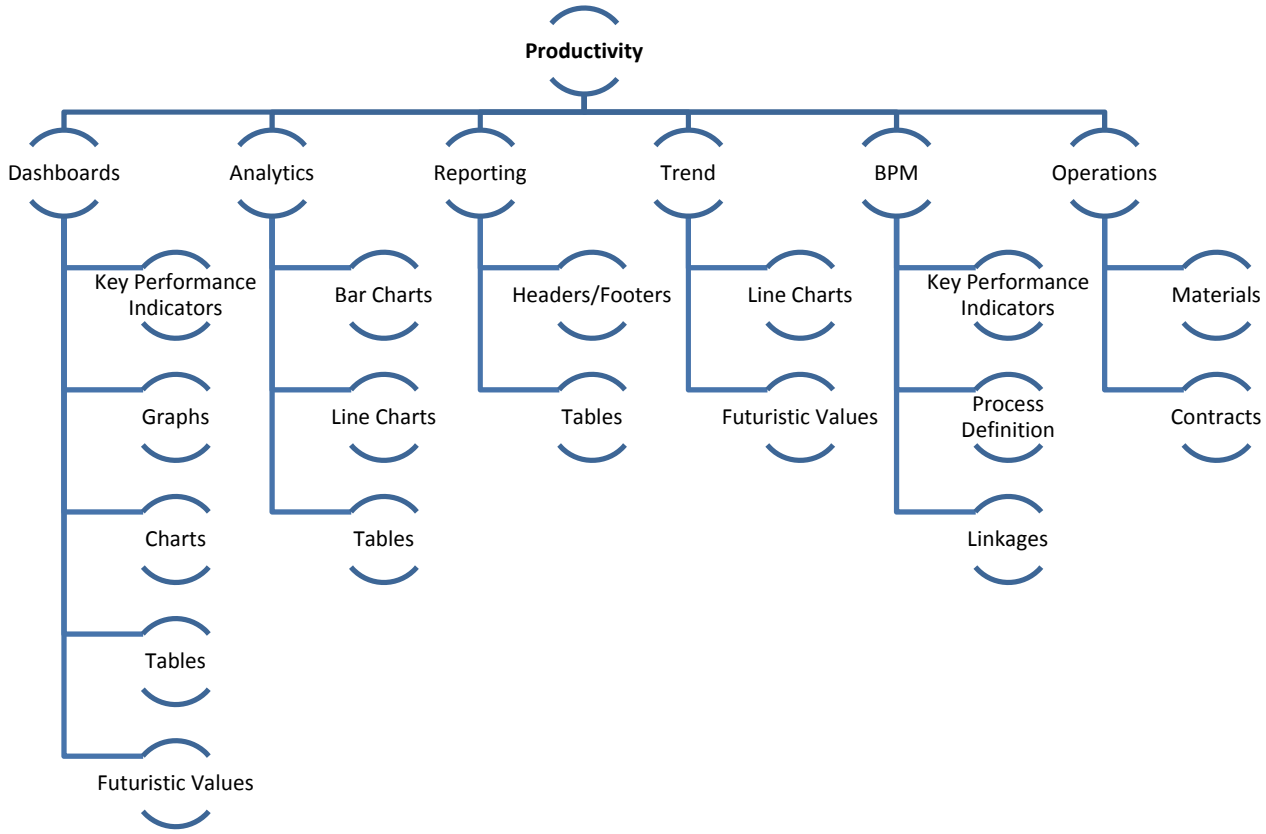


Figure 6.1 Productivity Class

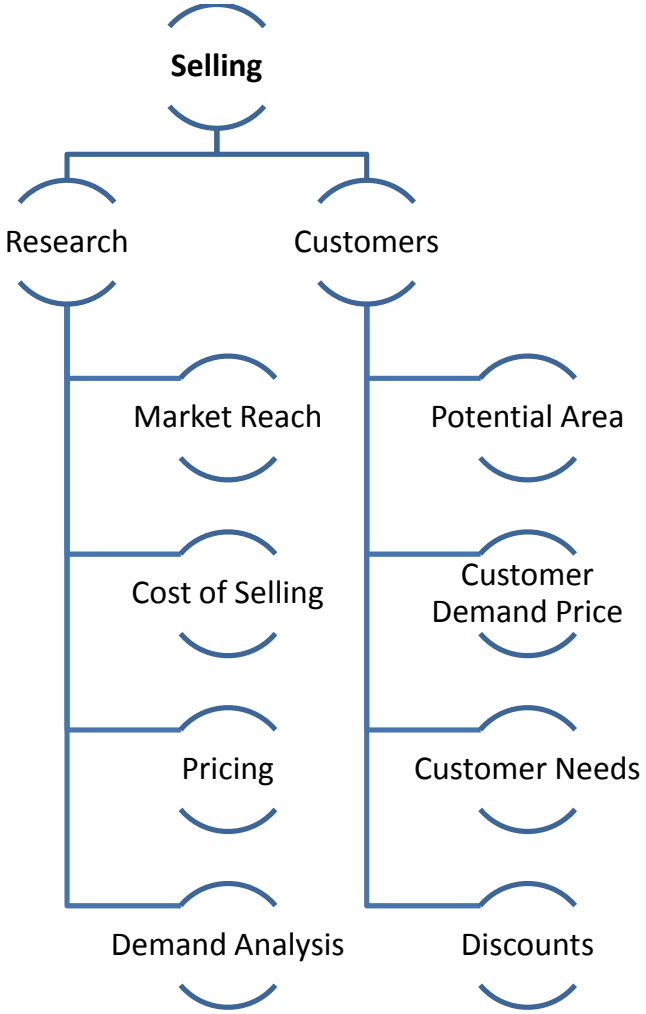


Figure 6.2 Selling Class

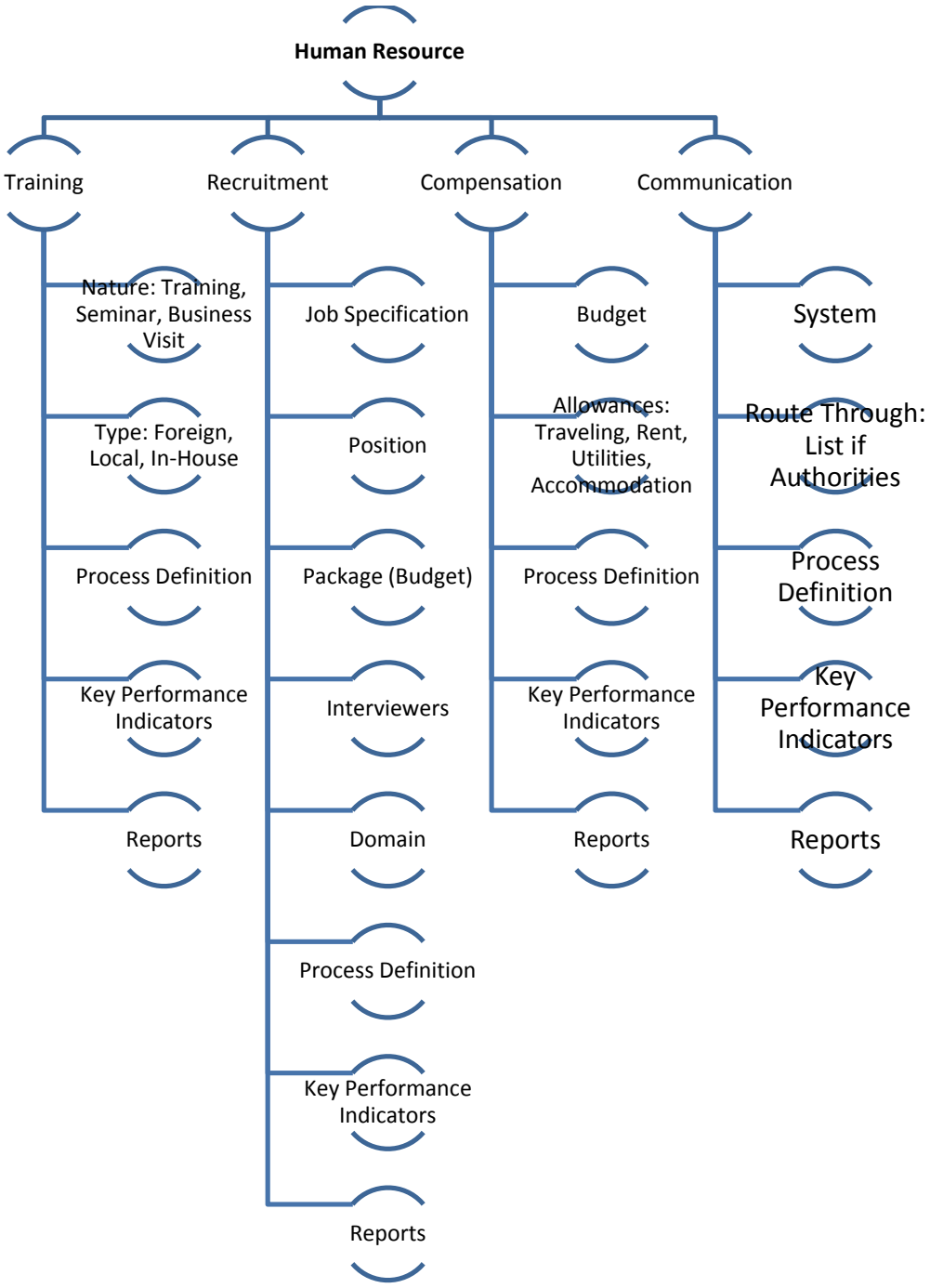


Figure 6.3 Human Resource class

Analyzing the Productivity class, we come to know that its sub-classes are dashboards, analytics, reporting, trending and operations, are all type tools to monitor the entire operational process. These are the tools that need to be utilized for effective monitoring. However, looking at the other side, they

are the Monitoring Processes. Thus, as per Rely/Guarantee, the data must be feeding by different classes.

Further, analyzing the Marketing class, we have analysis of customers, customer needs, market trends and competitor's analysis. On analyzing Sales, we come to know that market price analysis and regions where the customer could have access to the product or potential market analysis are critical. While analyzing Human Resource, we have trainings, communication and compensation.

Thus, applying Rely/Guarantee, we get that the Business Goals should be the main class, while Sales has to focus on Business Goals, e.g. Increase Sales by 20%. For that, they need to market that product and there is where Market class becomes functional. After getting functional, Human Resource would be either required to hire, or mobile them to achieve particular goal. The entire system feed data to Productivity class, where management can monitor their goals if at any instance, its achievable or not and can take prompt decisions/actions for effective functioning of the organization.

Figure .6.4 organization Functions

Communication class under Human Resource can be aligned by relying on its prior class, job specification of human resources. As we can see that if the external environment which in our case is goal of the management satisfies assumption that each position is classified as per their specification, then next class will surely be assuming the same properties. Thus, any communication be transmitted through technical channel will be understood as per their background knowledge. Therefore, the satisfaction of specification based communication will be the same as being in the relevant context. Therefore we can assume of the guarantee that correct message will be conveyed to stakeholder

Trend Analysis and Mining under production also can be aligned in a way that if all the sales information is correctly maintained within the data warehouse, i.e. satisfying the environmental factors, then mining through different statistical algorithm can be performed, maintaining the same environmental factors. This is through the consequence rule. However, internally, the mining can be done in parallel, in the way to apply algorithm on different regions simultaneously. So, if we want to find maximum sales as per

region, we can perform max function on information, executed in parallel to display the required output.

Similarly, advertising is initiated when an outcome from market research department is provided. This contains information of where and of what category of people the target should be, on the basis of which the Advertising department works out on different medium and comes up with the effective one, to propagate the message to masses.

Market response is therefore analyzed, which is the outcome of the advertising to feed information to sales department to start their function. Thus, at every stage, it all depends if each process is being satisfied with the external environment and other variables. If so, the other stage is then evaluated on the same factors and if it satisfies as well, this means they have rely. Thus can achieve their goals in combination.

Application of Rely/Guarantee

Parallel Method

As discussed earlier, in parallel method, few of the alignment activites are done in parallel, while through FORK, they can rejoin at some state, so output of more than one activities can combine at this point, and can be utilized for single activity.

Here, identifies that there are two of the classes that were not able to get aligned through traditional method. However, with our proposed algorithm being applied the it demonstrate the alignment process

Run in Parallel

Input Parameters of Productivity class is the data resides in data warehouse. Therefore existence of data warehouse should exists

Output Parameters, which includes graph, trends and predicted values/patterns after processing will results in Mining.

Check (1a) and (1b) in the entire Ontology. Shows that Analytics, Dashboards and Reporting, all have these parameters.

Replace class Analytics to Dashboards, Reporting as well as our Input Class Productivity.

Goto (1) and check Input Class Servicing Customer

Run in Parallel

Input Parameters is the Sales Data, which helps us analyzing the market

Output Parameter would be the analysis, which is Analytics class, which we already aligned above.

Check entire Ontology;

Market Response is actually the output of Sales, which in turns is the output of Advertising. Therefore, the Output Parameter of prior class is actually the Input Parameter of Market Response

As (2) finds Market Response similar to Customer Feedback class, as having same Input and Output of the class, therefore replace all found classes to Market Response. Thus aligning the Ontology.

Rule of Consequences

Input Parameters of Productivity class is the data resides in data warehouse. Therefore existence of data warehouse should exists

Check (1) in the entire Ontology. Show that Analytics, Dashboards and Reporting, all have these parameters

Check Output Parameters, which includes graph, trends and predicted values/patterns after processing will results in Mining

Replace class Analytics to Dashboards, Reporting as well as our Input Class Productivity.

Goto (1) below and check Input Class Servicing Customer

Input Parameters is the Sales Data, which helps us analyzing the market

Check entire Ontology;

Market Response is actually the output of Sales, which in turns is the output of Advertising. Therefore, we can see that Output Parameter of prior class is actually the Input Parameter of Market Response

Check Output Parameter would be the analysis, which is Analytics class, which we already aligned above.

As (2) finds Market Response similar to Customer Feedback class, as having same Input and Output of the class, therefore replace all found classes to Market Response. Thus aligning the Ontology.

6.8.2 Engineering Information Management System

Definition

Let revisit the development of different classes, where we declared attributes and applied traditional methods to align the ontology. Few classes have been covered which are Process Integration, System Integration, and Information Integration with its sub-classes. However, very few classes were there of which we were able to achieve alignment. Here, we will focus on few of those classes and apply Rely/Guarantee method to check if the alignment is possible with the method.

Introduction

Again aligned ontology through traditional methods and able to streamline few of the classes. However there were numerous classes that we didn't able to align. Among them we will discuss the high level alignment at class-level while we will discuss attribute level alignment of classes at second level.

For that, we first need to identify the business processes within the information system to capture the data flow between different processes.

Un-aligned Classes

Numerous classes can be mapped with their globally published classes, however within the same, there were very few which can directly be mapped to align the ontology through traditional methods. However, the completely unmapped classes are as follows:

System Integration: System, Network
Information Management: Storage, Sharing

Processes are defined before they are monitored. All the information of the process is stored at single repository, which is then be monitored to check its effectiveness. This can only be done if proper storage and sharing is deployed. Sharing and deployment is dependent on networks and systems. Therefore, each process is interrelated, and follows Rely/Guarantee. The outcome of networks enables system to interconnect with storage, the storage is shared among the monitoring systems, while entire process definitions are interconnected for smooth flow of information.

Alignment through Rely/Guarantee

Let's focus on the Information Integration class, while restricting its access to other dependent classes, Process Integration and System Integration. Following are the hierarchies of our ontology

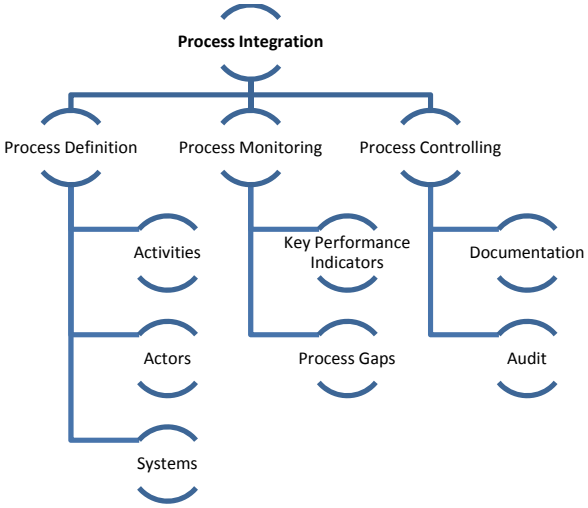


Figure 6.5 Process Integration Class

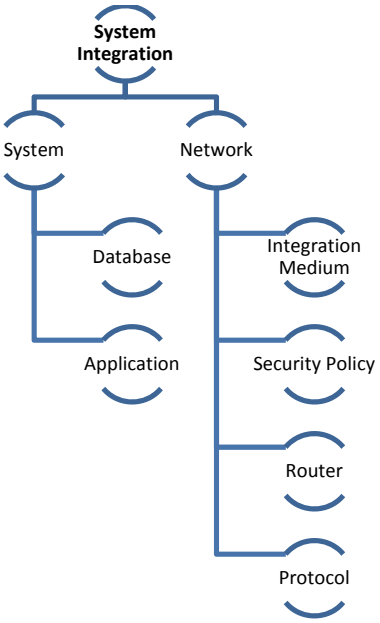


Figure 6.6: System Integration Class

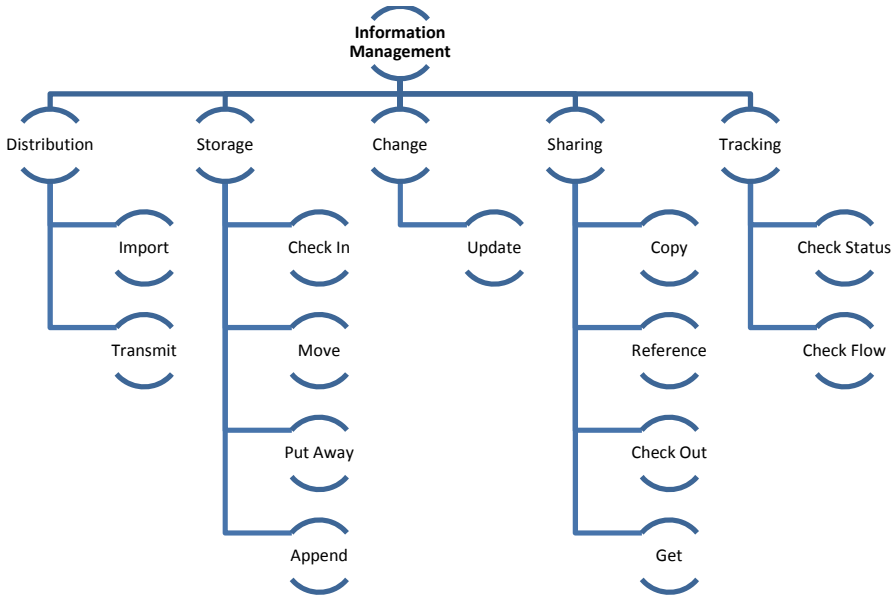


Figure 6.7 Information Management Class

Lets start by with explaining process integration. All business processes should be well-defined to capture each and every event within the process. This is only possible with the tight integration with the processes. Thus, within the sub-processes, we have Rely/Guarantee approach that one of the process transfers data to other sub-process, thus completing the entire process cycle. Every sub-process satisfies the same environment factors, for the reason their results contain in a guarantee (G). For example, obeying company, corporate and legal law the purchase department procures by going through the procurement process, which gives outcome to finance department to pay for the procurement. Thus, two different departments rely to finally get guarantee. As each department satisfies common environment, they can be made as a single class, the attributes would be same, though they cannot be aligned through traditional method as none of their attributes matches with each other.

Thus, the process integration with Rely/Guarantee can be achieved. However, for that, system integration also comes into play. Without integration of systems, process integration is not possible, as data can get corrupted or incorrect data can flow in the process. For that, Process Integration, having same environmental factors, guarantees the flow of information, and as an outcome, system integration takes inducts it into their channel to let it flow to other process. Thus, it follows Rely/Guarantee

method. This can become sub-class of the Process Integration class, as an instance can be created within it. Managing Information is another class which seems independent of system and process integration. However, if Rely/Guarantee is applied , we can find that through this class, the processes are being relied, while systems are just the hardware part of it. Thus, with different sub-classes, the process integration can be achieved. For example, Sharing class of Information management actually lets outcome of the process A to feed to Input of process B.

Figure 6.8 Managing Information – Broad Aspect

In following section, discusses the classes that were unaligned through traditional methods , demonstrate how we can have it aligned through Rely/Guarantee

Application of Rely/Guarantee

Parallel Method

As discussed earlier, in parallel method, few of the alignment activities are done in parallel, while through FORK, they can rejoin at some state, so output of more than one activities can combine at this point, and can be utilized for single activity.

- System Integration: System, Network
- Information Management: Storage, Sharing

Run in Parallel

Input Parameters of System Integration class is the data resides in application, software which itself in some hardware. Therefore existence of application persist

Output Parameters, which includes providing functionality to run data, while with proper authentication and network channel

Check (1a) and (1b) exist in the Ontology by traversing each node and testing its input and output parameters

Input of Database Application is same as of System Integration

Output of Database Application is same as of System Integration

IF found similar, DO

Replace the class with name of later identified class

IF Input Paramater was of last node, STOP loop ELSE

Goto (1) to fetch new Input Parameter

Run in Parallel

Input Parameters of Information Management class is the data resides in application as Storage attribute

Output Parameters, which includes providing functionality to share data on common channel with proper authentication

Check (1a) and (1b) exist in the Ontology by traversing each node and testing its input and output parameters

Input of Database Application is same as of System Integration

Output of Database Application is same as of System Integration

IF found similar, DO

Replace the class with name of later identified class

IF Input Paramater was of last node, STOP loop ELSE

Goto (1) to fetch new Input Parameter

Rule of Consequences

Input Parameters of System Integration class is the data resides in application, software which itself in some hardware. Therefore existence of application persist.

Check (1a) exist in the Ontology by traversing each node and testing its input and output parameters

Input of Database Application is same as of System Integration

IF Found, Check Output Parameters, which includes providing functionality to run data, while with proper authentication and network channel

Check (3) exist in the Ontology by traversing each node and testing its input and output parameters

Output of Database Application is same as of System Integration

IF found similar, GOTO (6) ELSE GOTO (1) for Next Input Parameter

Replace the class with name of later identified class

IF Input Parameter was of last node, GOTO (1) below loop ELSE

Input Parameters of Information Management class is the data resides in application as Storage attribute.

Check (1a) exist in the Ontology by traversing each node and testing its input and output parameters

Input of Database Application is same as of Information Management

IF Found, Check Output Parameters, which includes providing functionality to share data on common channel with proper authentication

Check (3) exist in the Ontology by traversing each node and testing its input and output parameters

Output of Database Application is same as of Information Management

IF found similar, GOTO (6) ELSE GOTO (1) for Next Input Parameter

Replace the class with name of later identified class

IF Input Parameter was of last node, STOP loop ELSE

6.9 Reasons to Incorporate Solution

There exist two levels of mismatches in Ontology

Ontology Level or Model Level

Ontology Language or MetaModel

The above mismatches include the syntax based mismatches, semantics differences in language and structural or expressivity of the language. The same is highlighted in Michel Klein (2001)

As earlier said, the languages can be different in semantic way, syntactically, and structurally – having different meaning in different ontologies. For the reason, to align them, Ontologies are to be brought in same language first. These all comes under normalization process. All the aforementioned techniques are applied before ontology-mapping.

Though, they can be matches as per the different alignment techniques, however, because of having different linguistic meanings for the same term, mismatching may result. For instance, either for same concept, different terms are used OR same term can be referred in different concepts. The mismatching can also be caused through different modelling paradigm, convention and level of granularity. The same is referred by (Noy ,2004).

There exist different types of mismatches; however it is always difficult to uniquely identify the kind of the mismatch that is currently being occurred. As according to (Predoiu et al 2006), classification is also important to find the likely mismatches and the kind of mismatch that can be resolved with some formulism. Further, the detection of mismatches through some matching algorithm can also be done.

According to (Chungoora et al 2008), mismatches of concepts occurs when two or more concepts of certain domain are used in a single ontology. Further, following are few of the mismatches that can conceptually occur from certain domain while making the ontology unaligned:

Class Mismatch: Different classes and subclasses are used in Ontologies
Categorization Mismatch: Different structure of same class, having different associations exist in Ontology
Aggregation-Level Mismatch: When classes are same but one has additional attributes or different level of abstraction, this mismatch occurs
Relation Mismatch: This type of mismatch is because of different properties of same class in different context
Structure Mismatch: This occurs when the relations of the attributes are different in one class while relations are different in another class
Attribute-Assignment Mismatch: This type of mismatch occurs when same relations exist in two different Ontologies, however the assignment of the attributes are different as being utilized in different concepts.
Attribute-Type Mismatch: This type of mismatch occurs when same value type are defined in different way, i.e. a value is defined under classification A in particular class but is defined under classification B in another class
Syntax mismatching can be resolved by transforming it to likely representable in both the ontologies. There could like be the condition that syntactically the classes are different however, logically there can be like the classes are same. To avoid this condition, one should always do normalization before going deep into mismatching. Semantically, if there exist different classes being mismatched, they can be handled by utilizing same semantically published classes in both the Ontologies rather deriving new terms or representing by using new terminologies. However, this can be done while designing the Ontology. This again can lead to inconsistency when all terms are utilized from common repository, semantics can be changed of concept in one class while utilizing the same concept in ontology's class, where semantics might be different.

6.10 Applying Compositional Ontology Alignment

As Information Management is an integral part of any Organization. We can apply alignment method to make it interoperable so that the ontologies can tightly be integrated. Earlier proven that through traditional method, are unable to integrate Information Management Ontology with Organization Structure Ontology. Although it should be tightly integrated as all the data flow through the channel. System Integration and Information Management are the core classes that can actually integrate two of the Ontologies. Also earlier proved that no traditional method is applicable to make the two mentioned classes aligned so as to make them interoperable.

With our proposed method, in the above section, we were able to align the Ontology, thus making them integrated with Organization.

Through Rely/Guarantee, it is possible to integrate the Ontologies as follows, (applying parallel rule)

6.10.1 Customer Analysis

Run in Parallel

Input Parameters is the Sales Data, which helps us analyzing the market

Output Parameter would be the analysis, which is Analytics class, which we already aligned above.

Check entire Ontology;

Market Response is actually the output of Sales, which in turns is the output of Advertising. Therefore, we can see that Output Parameter of prior class is actually the Input Parameter of Market Response

As (2) finds Market Response similar to Customer Feedback class, as having same Input and Output of the class, therefore replace all found classes to Market Response. Thus aligning the Ontology.

6.10.2 System Integration

Run in Parallel

Input Parameters of System Integration class is the data resides in application, software which itself in some hardware. Therefore existence of application persist

Output Parameters, which includes providing functionality to run data, while with proper authentication and network channel

Check (1a) and (1b) exist in the Ontology by traversing each node and testing its input and output parameters

Input of Database Application is same as of System Integration

Output of Database Application is same as of System Integration

IF found similar, DO

Replace the class with name of later identified class

IF Input Paramater was of last node, STOP loop ELSE

Goto (1) to fetch new Input Parameter

6.10.3 Information Management

Run in Parallel

Input Parameters of Information Management class is the data resides in application as Storage attribute

Output Parameters, which includes providing functionality to share data on common channel with proper authentication

Check (1a) and (1b) exist in the Ontology by traversing each node and testing its input and output parameters

Input of Database Application is same as of System Integration

Output of Database Application is same as of System Integration

IF found similar, DO

Replace the class with name of later identified class

IF Input Paramater was of last node, STOP loop ELSE

Goto (1) to fetch new Input Parameter

6.10.4 Productivity

Run in Parallel

Input Parameters of Productivity class is the data resides in data warehouse. Therefore existence of data warehouse should exists

Output Parameters, which includes graph, trends and predicted values/patterns after processing will results in Mining.

Check (1a) and (1b) in the entire Ontology. We can see that Analytics, Dashboards and Reporting, all have these parameters.

Replace class Analytics to Dashboards, Reporting as well as our Input Class Productivity.

GOTO (1) and check Input Class Servicing Customer

In summary, the output of the one sub-system is being provided as an information being input in later system so as to make the ontology aligned and integrated to make it interoperable in semantic world. The summary can better be depicted through the following chart

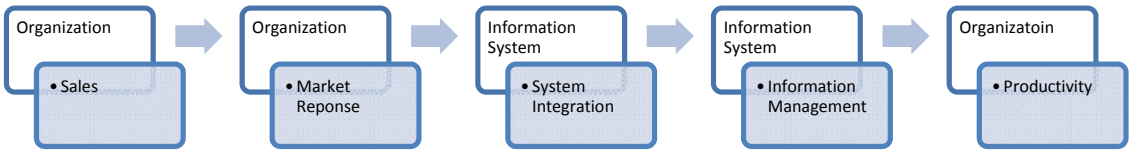


Figure 6.9 Rely/Guarantee alignment process

With the above process flow, this shows how the two Ontologies are aligned through Rely/Guarantee way. In the process flow, as the Market Response in an Organization from its customers is dependent on the Sales data. Now both Sales and Market Response are two classes, associated with the Organization Ontology. The attributes of them are dependent as one is feeding information to another. To analyze market trends by means of market response, we have to connect Organization Ontology with Engineering Information Ontology, in order to have information for Data Mining. With System Integration, the relevant data is fetched of the Market Response, while Information Management is fed with data. The data which resides in data-warehouse will now be evaluated according to which predictive analysis will be performed. The same then leads to better Productivity. This is how the complete flow will work.

6.11 Explanation

Storage	Move	Append
Check In	Put Away	

Change	Reference	Distribution
Update		Import
Sharing	Check Out	Transmit
Copy	Get	

Let's analyze identifying and explaining differentiation factors on two of the classes. Lets cater one scenario to actually analyze how the classes can be aligned, while how the same can be better aligned by means of rely/guarantee method. In Information System Engineering module, of which we developed the entire ontology in earlier sections, has three main classes. They are Storage, Sharing and Distribution. In relating, we can see that Check-In is the attribute that is not find in any of the other class. The same is for the attribute Move, Put Away and Append, which exists in Storage class while not in other classes. Similarly, Copy and Reference are two of the attributes which are not found in any class to let it relate to them. Check-Out and Get are also an attribute of the same class which are not available as per the matching algorithm. Also, Import and Transmit are two of the attributes which are related to distribution class, while they are not available in any other class.

Now let's relate the attribute to match the classes in a traditional way. For each of the class, it's addressed how it will get aligned through our proposed method, i.e. in rely/guarantee way

In Traditional Methods, we have the following classes while accordingly the comparison if they can be mapped through traditional ways to achieve the ontology alignment. The final column will show how our proposed methods actually come up with the ontology alignment technique

Class	Terminology	Semantic	Expansion	Structural	Rely/Guarantee
Storage	X	X	X	X	Processing
Sharing	X	X	X	X	Publishing
Distribution	X	X	X	X	Sharing

Table 6.1 Summary of explanation

From the above summary in table, we can have

Check-In, Move, Away, Append, Change and Update are the attributes of storage class

Copy, Reference, Check-Out and Get are attributes of Sharing

Import and Transmit are of class Distribution

Terminologically, none of the attribute is similar to attribute of any other class

Semantically, none of the attribute is similar to attribute of any other class

Expansion, as none of the class has similar instances, utilized in any other, therefore it can be said that it cannot relate it to get the aligned ontology

The same goes in Structural way, as every class is differently structured and therefore cannot be mapped

However, in terms of our proposed method, i.e. Rely/Guarantee the same scenario as follows:

The *Processing*, after getting completed will provide input to the *Storage* that can process and provide output to some other class. This is the main idea that when so ever *Storage* ignites, it is relying on *Processing* class for a guaranteed input for generating some output. The same dependency goes for *Sharing*, which is depending on *Publishing*, which itself is depending on the class *Development*. Thus, it's a chain of dependency, and thus could be aligned through our proposed way. Now, going into depth of the rely/guarantee, here we can align *parallel method* of rely/guarantee as there are two of the separate dependent classes that can mutually start and stop at certain point, which includes *Processing* class, while the other is *Distribution* class. This occurs on *Update* attribute as whenever the class is updated / modified, it is being processed, while its earlier version may be utilized by means of *Distribution* class. The same goes for *Distribution* class, which will be given the input from the *Sharing* class.

6.12 Advantages of Assumption /Commitment

For our proposed algorithm of Rely-Guarantee, there are many advantages to align the entire Ontology, while making them integrated to reutilize current ontologies. However, with the advantages, there are different drawbacks of using it.

6.13 Advantages over Traditional Methods

With our analysis over different algorithms in different proposed white-papers, we identified following advantages of rely-guarantee over traditional methods

6.14 Object-Based Matching

In traditional methods, the algorithms are based on matching either Syntax or Semantics of attributes of different classes. With our proposed solution in which single attribute is available, it will be based on matching the outcome of the class, so to make it dependent to other class. This will automatically keep the ontology aligned, even when new class is added.

It can better be explained with an example. For instance, if a class named Juice which has different ingredients including Water, Flavor, Salt etc. And have another class of Human, which comprises of 71% Water. If is needed to align Human Life ontology with Juice Production ontology, they can be aligned classes by putting water and salt ingredients and making instances into each class. For that, it's a must to know all the attributes. This is what all is required for traditional method. When it comes to apply rely-guarantee approach to align ontology, the Juice Production is dependent on Human Life ontology, if there is no Human Life, there is no Juice. With said that, its possible to make class Water and Salt of Ontology Juice production dependent on the intake of Human Life. Thus, the process is well-aligned while Ontology itself gets aligned. Therefore, cumbersome activity can automatically be omitted of matching each attribute of each class of two different ontologies. Also, the dependency over having a set of attributes in the form of dictionary is also omitted. Therefore, our proposed ontology is having an edge over traditional methods of Ontology alignment.

6.15 Semantic Matching

For semantic matching through traditional methods, we do need to have comprehensive dictionary which maintains meaning of each term pertaining domain of its utilization. The same is required to align ontology on the basis of domain, being applied to test the current ontology in the dictionary and maintaining the same term, which we associated as Attribute, in the target ontology in order to align it.

With rely-guarantee approach, we can void the semantic matching of the single attribute, making the ontology complex so as to come up with the synchronized attribute. As one ontology is feeding information to another, the concept of semantic doesn't remain at attribute level, but at Ontology level. Therefore, two similar attributes of same semantics can be associated automatically when two of the Ontologies are integrated and being fed by source Ontology to Target one.

For instance, let's say we have two ontologies of Computer Development and Printing Press. The packaging in both of the ontologies have same semantics, but for that each attribute is first checked manually or semi-automatically through traditional method in order to utilize the same attributes in both of the Ontologies. Now, with Rely-Guarantee, we know that computers, after getting packed are put in the cartons, which Printing Press supplies. At business process level, we checked the semantics, and with Feeding of class "Installed" in Computer Development ontology to "Pack" of class Packaging of Printing Press, we can easily integrate two of the ontologies, which its class is automatically aligned, as Packaging from Computer Development class is automatically voided.

Thus, rely-guarantee has advantage of semantically integrating ontologies, rather semantically matching each attribute of a class through traditional methods.

6.16 Algorithm Efficiency

Our proposed approach is also efficient in terms of algorithmic runtime. This is because of few reasons as follows

- It does run in parallel
- It does not require to go through each attribute in order to align the ontology
- It can be make quicker by increasing no. of CPUs, as works on Threading basis

In traditional methods, the algorithm works like for the given ontology, it runs through each class, while for each class it goes for each attribute. The same attribute is matched with all other attributes of all classes. Therefore, If O is no of Ontologies, N is no. of Classes, and M is no. of Attributes, the algorithm will execute (OxNxM)^2

From the above, it is obvious that if we want to align two or more ontologies, that may take days, while its result do not guarantee to be accurate, as at current, only manual or semi-automatic algorithms exist to align the ontology. With the proposed ontology, this process is overruled by the fact that the

semantics of entire ontology is mapped to the other ontology in order to make it aligned. Thus, going through each attribute of each class won't be required, while classes being fed from one ontology to another, in order to make it the ontology aligned by mapping their semantics.

6.17 Runtime Analysis

As said in the preceding section, the traditional method not only takes too much time to run, but also requires more hardware and other resources in order to make entire ontology aligned. This also includes the no. of steps requires to be processed comparatively much more than that of applying rely-guarantee approach.

With that said, by doing runtime analysis, we can come up with the conclusion that no. of steps involved in aligning ontology through traditional approach are much more, as well as are manual, while in our approach, the steps are drastically reduced. With our approach, we can also conclude that not only the steps are reduced but also only requires manual intervention when semantically mapping the ontology, while the rest will be done by the plug-in through automated approach, thus a semi-automatic approach with few no. of steps.

6.18 Protégé Development

The current plug-in for protégé is applicable to align two ontologies traditionally. However, that does require lots of metadata, that should be incorporated to each attribute as well as class and ontology, in order to let the component match the attributes and classes. Also, with semantic matching from traditional algorithms, it requires dictionary with extensive terminologies and corresponding semantics to enable the plug-in work in the best way.

With our proposed method, the semantic approach for attributes is irrelevant, therefore there isn't any need of comprehensive dictionary to let it be run. Therefore, there is not dependency on external content. Also, our method can be run on the basis of Threading approach, we may say that our approach of doing it in protégé can be done effectively with utilization of full resources of protégé development environment as well as hardware utilization as most of the computers having multiple cores. The earlier methods doesn't run in parallel and therefore doesn't comply with multi core processors, while our solution works best with utilization of multicores.

6.19 Automatic and Semi-Automatic

Traditional methods are mostly dependent on manual work. This include incorporating metadata to each attribute and class so as to apply methods efficiently. The traditional algorithm can therefore work on manual and semi-automatic while none of the algorithm is fully automated, unless the model is designed in the particular way.

With the rely-guarantee approach, which is currently semi-automatic as planning is to be done while matching the whole ontology semantically, it can be make it automated which requires time. This is because when all the ontologies are made efficient and published globally, it can maintain semantics of each of the ontology. So, whenever a new ontology is designed, it will automatically go through the matching of classes of each publically available classes and can find its semantics, while can be aligned in the same way. Here, we need to focus with the suggestion of introducing Ontology Body, similar to PMBOK, to devise standards and initially go through the Ontologies. When the process is gets into cycle, it can then become mature and can be adopted globally to make the semantically aligned ontology.

6.20 Alignment Turn-Around

In traditional methods, not enough ways to match each attribute from one Ontology to another. This is because of the different semantics of same attribute in different domains. Therefore, the turnaround of aligning ontology through traditional ways is not 100%. With our proposed methodology, though it is not 100% in the beginning to get aligned Ontology, but with the concepts of Business Process Reengineering, we can have integrated system, the AI agents of which can learn themselves. By time, we can have trained agents which lets the Ontology semantically aligned in order to have 100% results. Therefore, there is a lot of work that can be done in our proposed method, while earlier methods don't have much room for enhancement.

6.21 Integration

Integration is one of the main aspect in semantic world. We intend to align ontology in order to have proper integration within the processes or systems. With said that, traditional methods works in silos in order make them effectively run. Traditional method only measures ontologies that are currently in scope. Therefore, limiting the matching technique - syntactically as well as semantically and structural approach.

With our proposed method, the entire system works on the basis of integration. It does align on the basis of feeding information from one system to another, therefore it gets automatically integrates two or more of the systems. The biggest advantage is, the ontologies which are out of scope (i.e. currently not considered) are also come into integration aspect so user, designing ontology, can better have idea of how the ontology will be impacting and how much the ontology will get dependent. Therefore, our approach offers tight integration.

6.22 Global Semantic

With the aforementioned topic, as traditional method doesn't counter ontologies which are currently out of scope, while our approach does cater them as well, in order to make the Ontologies tightly integrated, therefore with the globally publishing of the Ontology, it can make the entire ontology interconnected so as to have presence in semantic world, globally. This idea doesn't cater by the traditional approach, but by our idea of semantically align different systems so as to connect different systems. This gives the platform of connecting systems from one corner of the world to another, therefore creating new chances for the business to expand more, and having presence globally, with optimized way of processing information.

6.23 Example Ontology

Let's have a real-life example of Ontology, which we later align through traditional methods and then through our Assumption/Commitment method. After explaining it through the examples, we will show how the algorithm is designed to implement the same through some programming language, e.g. JAVA

We have FINANCE department, which is part of ORGANIZATION. We also have INFORMATION TECHNOLOGY and DATA MANAGEMENT departments which are also part of ORGANIZATION. FINANCE department has REPORTS requirements, therefore REPORTS are utilized by FINANCE. These REPORTS are pre-formatted by INFORMATION TECHNOLOGY department. As DATAWAREHOUSE is part of DATA MANAGEMENT, the STATUARY e-FILES are developed by the department. The same is required by FINANCE department for their reporting. KPI Dashboards are required for monitoring performance of the ORGANIZATION; therefore the same is required by MANAGEMENT, as their key element for MONITORING purpose. MANAGEMENT also requires ANALYTICS to perform What-IF ANALYSYS as an example. The same is developed by DATA MANAGEMENT department to provide the same through their internally maintained data-island DATAWAREHOUSE. The Ontology of this scenario will be as follows:

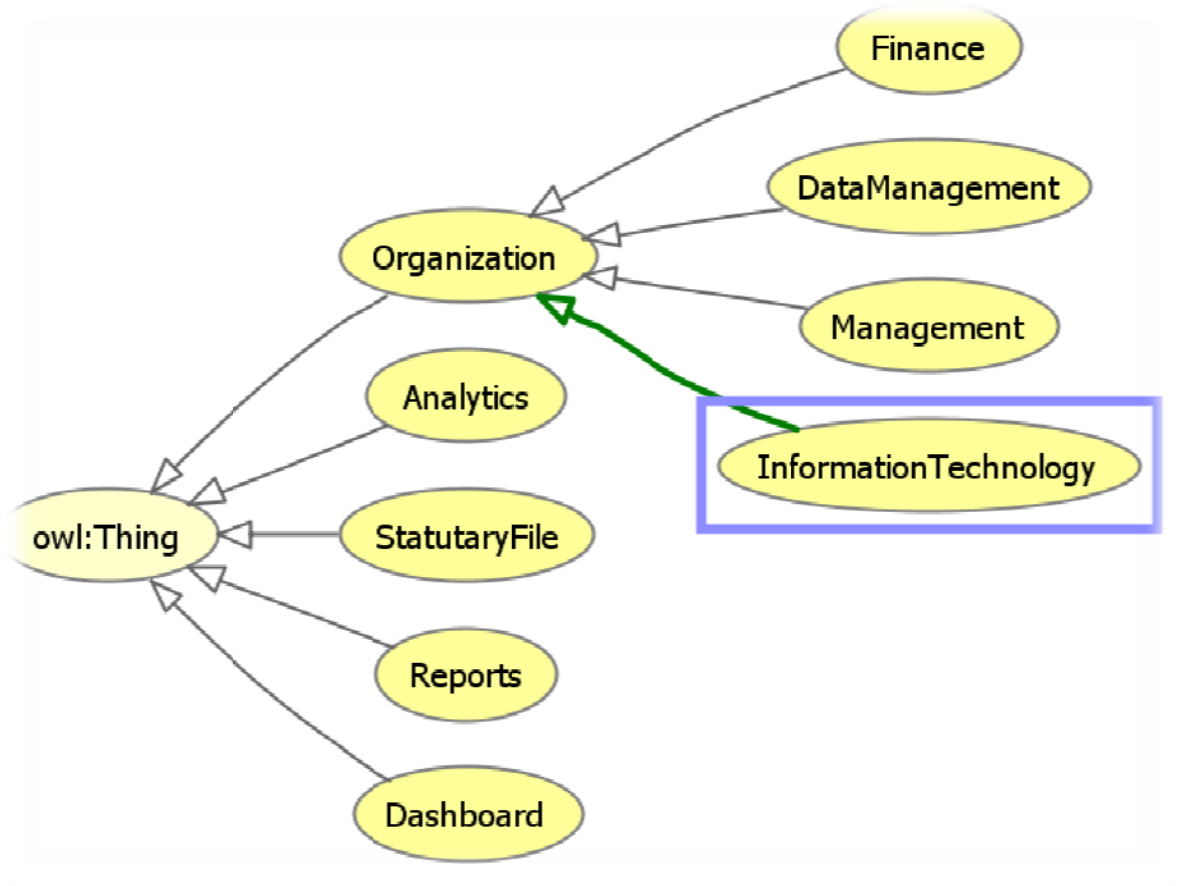


Figure 6.10: real-life example of Ontology

6.24 Traditional Method

If we apply traditional methods, following are the outputs that we will have

6.24.1 Terminology Technique

This technique caters the alignment through the terminology or syntax of the class. The entire ontology is traversed, and each class is checked for its recurrence. The runtime is therefore O (n^2). It's a time consuming and might not align the ontology, as not all syntax can simply be mapped

Results:

There isn't any class that have similar name. Therefore, we can conclude that the terminology technique is ineffective on our example Ontology.

6.24.2 Structural Technique

This technique deals in matching structure of each class in Ontology. This means that attributes of each class are matched, that may result in the similarity of 1 or more than 1 attributes to be matched. Thus finding similarities and coming up with the ideal similar class generalizes the class. At the end, it results in aligned ontology.

Results:

There are few classes, i.e. Analytics and Dashboards, which have same structure which includes Graphs, Charts and Tables. Further, Statuary Reports and Reports have same structure, i.e. Balance Sheet and Profit and Loss Statements. Thus they can be assigned as one. The aligned Ontology would become as follows

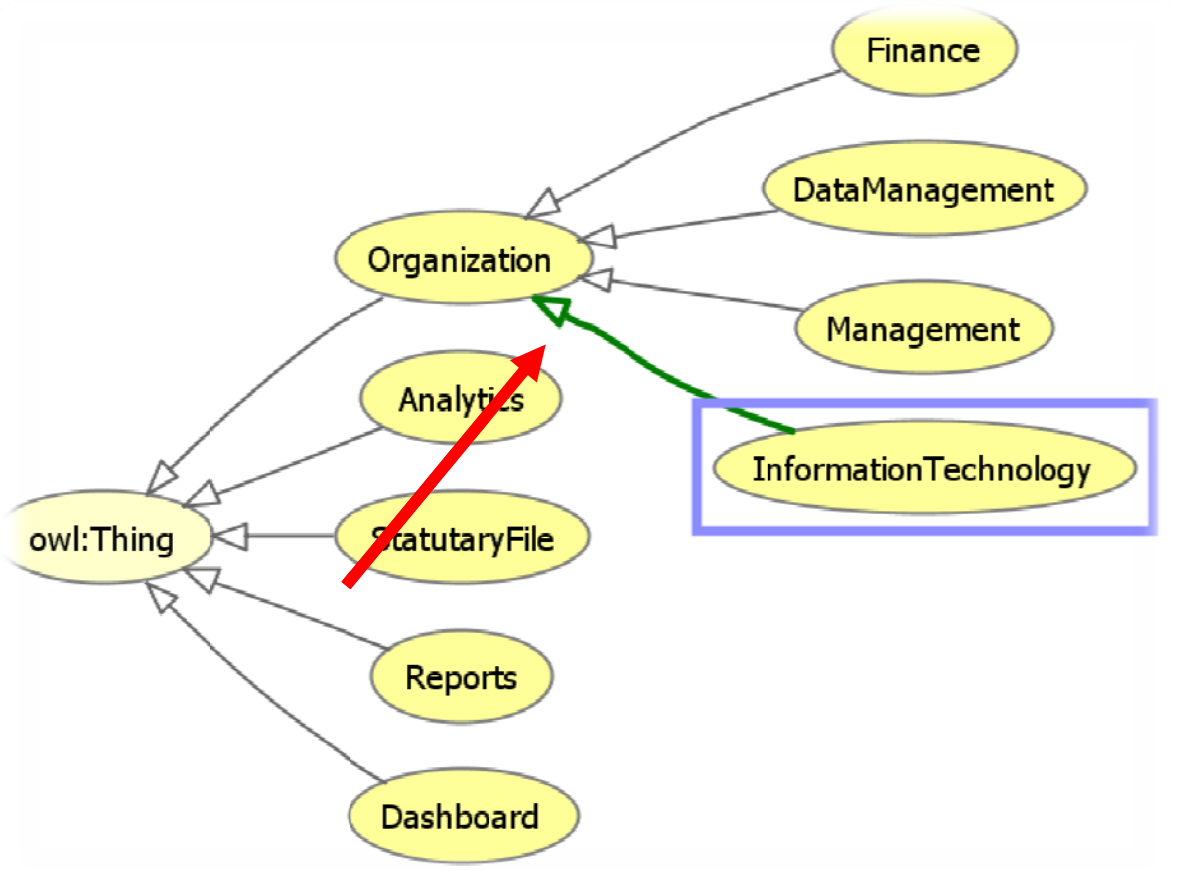


Figure 6.11: Structurally Aligned Ontology

From OWL language, we can see that these are assigned to Information Technology, Data Management as well as to Finance. However, we cannot perform a recursive function to check their parent level in this technique. Therefore, only two classes are omitted.

6.24.3 Extensional Technique

This technique deals in aligning the classes in ontologies, depending on their instances. So, the similar instances can be compared and if found with the same value, their classes are made similar. This results in aligned ontology, which is better than the structural technique.

Results:

There isn't any class that has more than one instance. Therefore, we can conclude that the extensional technique is inapplicable in our example Ontology.

6.24.4 Semantic Technique

It can further be elaborated in a way that the first stage should be testing the class on syntax basis and then comes to the local semantics of particular class. If it doesn't match, try to find the class in global semantics. This will apply interoperability on the defined classes. For being effective, reference of each syntax and its semantics should be followed by same repository. Thus, proper Descriptive Language is mandatory for such technique.

Results:

We can see that the Management class requires Dashboards and Analytics, while Finance requires Internal and External reports. The same is developed and provided by Information Technology and Data Management respectively. There isn't any class that have similar name. Therefore, we can conclude that the syntax is not matching in example Ontology. We can seek the definition of each class on globally published classes, and can relate if it can be aligned. It's a cumbersome process. However, following is the result of application of semantic technique

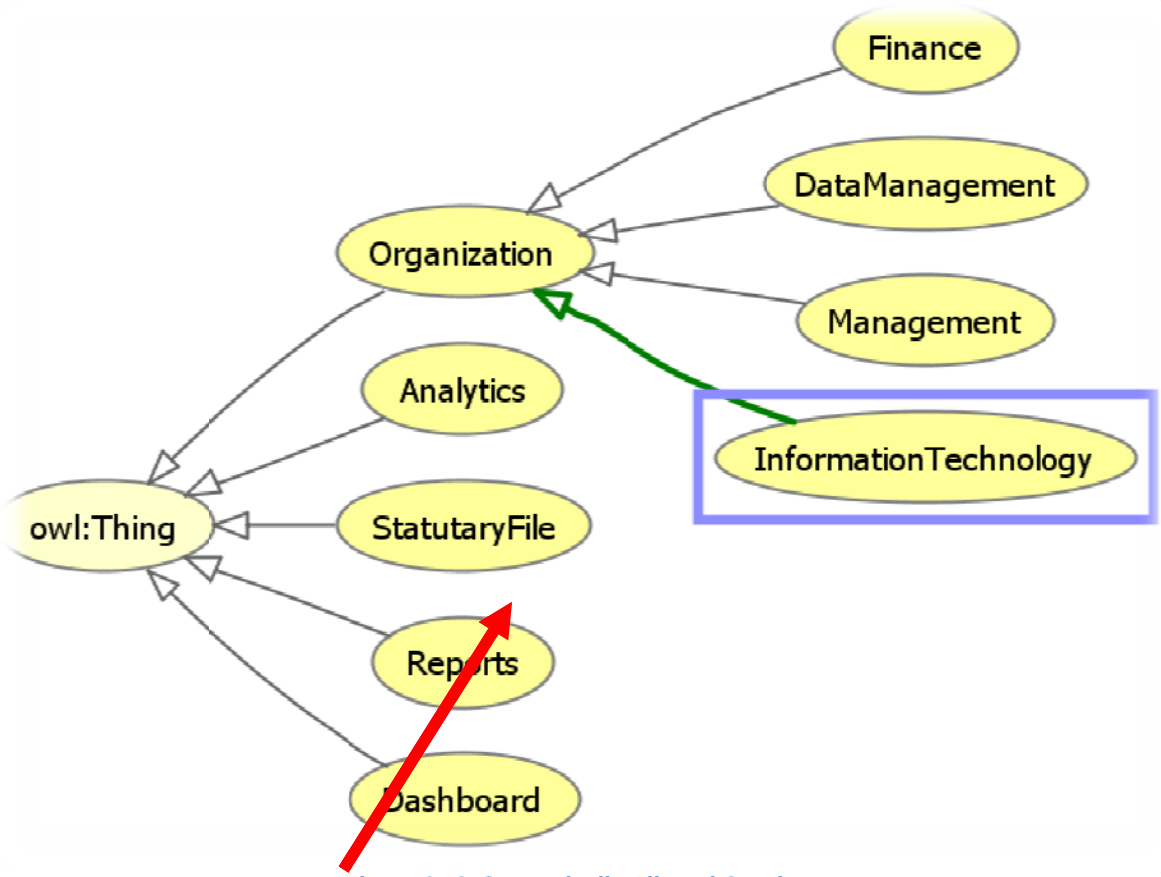


Figure 6.12 Semantically Aligned Ontology

It has been done by finding semantics of statuary file and analytics, have same meaning as being declared with attributes (refer to code), while Report and Dashboards are generalized, therefore can be omitted. Again, the issue persists that these are assigned to some classes, which are not tested in semantics.

As we have results of traditional methods, we now apply Ass/Com method to align the Ontology.

6.25 Assumption/Commitment Method

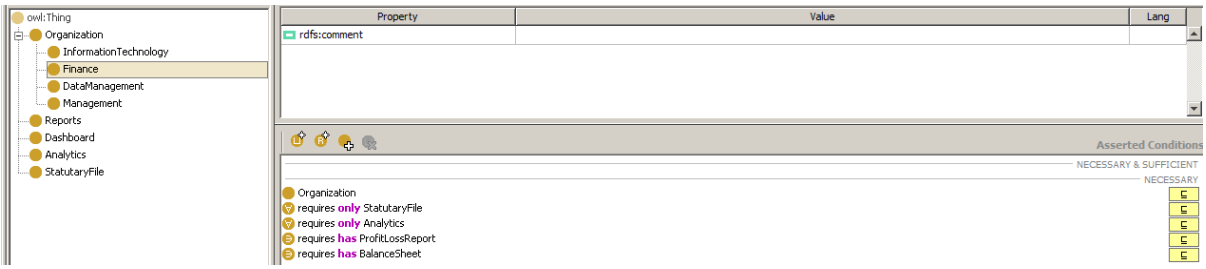
As provided earlier in the introduction section of Ass/Com, the idea is to align entire Ontology by following the concept from parallel computing of dependency of input in a system over the output of the prior system. For this, lets have a look at properties of each class

6.25.1 Classes

Information Technology

From attributes, we can see that Information Technology inputs data from Finance department or Management, prepares either Statuary Report or Analytics

Finance



From attributes, we can see that Finance department only requires Statuary Report or Analytics. Thus, it provides information to Information Technology department to develop and provide the same.

Data Management

In Data Management, it gathers requirement for reports from Finance and for dashboards from Management. It then develops as per the requirements and provide the same.

Management

From attributes, we can see that Management asks Data Management to fulfill their requirements by providing required data of Dashboards and Analytics.

6.25.2 Alignment

From the concept of Ass/Com, we first create sub-components so as to make a model of -- outcome is an input for next system

Dashboards have the requirement of Pie Chart, Bar Chart, Table, and KPI indicators, while Analytics has also the same requirement. These are not the only components of the class, but these are actually the requirements which they collect from prior system to process and produce output for another system as an input. They both produce the Analytical Platform known as Analytical Interface to monitor performance. Thus, as their Input and Output are the same, while their internal structure would surely be different, they can be classified as one class, with different inherited properties for internal behavior.

Reports have the requirement to produce structured information in Table form. Thus, its input is the data in required structural form. In Statuary Reports, we have Balance Sheet and Profit and Loss Statements. They have some particular structure, while they also ask to input data in required structural form. For instance, in our case the input will be Worth of Assets, Liabilities and Equities for Balance Sheet, while Expenses, Revenue and Profit (Before and After Tax) for P&L Statement. The same is required for Reports by Management. Therefore, though, their structure is different for processing, but their input and output are same. They can be assigned as one with different properties declared within their process.

Finance department, for publishing their reports if an organization is public-sector or like to increase their share want to publish their statements, they ask Information Technology department to provide

the same report. For historic reports, they can ask Data Management department to provide the balance sheet and profit & loss statements of previous years.

As we have earlier said that inputs and outputs of Statuary Report and Reports are same, therefore they can be considered as an instance of single class. By applying Ass/Com Input is the requirement by Finance Department, while two have same Input association for Finance Department, i.e. Information Technology and Data Management. They are producing final output, i.e. Reports; therefore, their Input and Output are same as well. Hence, we can make single class for Information Technology and Data Management as Data Warehousing Solutions department.

The same is applicable for Management. This includes asking for the required monitoring system to develop for better performance measurements to Information Technology Department, while for Mining and Trending, they can ask for Data Management.

As we have earlier said that inputs and outputs of Analytics and Dashboards are same, therefore they can be considered as an instance of single class. By applying Ass/Com Input is the requirement by Management, while two have same Input association for Management, i.e. Information Technology and Data Management. They are producing final output, i.e. Reports; therefore, their Input and Output are same as well. Hence, we can make single class for Information Technology and Data Management as Data Warehousing Solutions department.

The aligned Ontology through Ass/Com will become as follows:

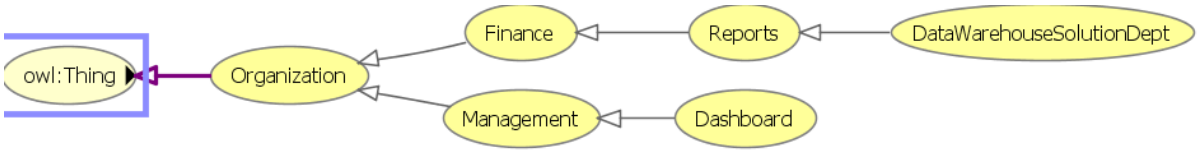


Figure 6.13: Ass/Com Aligned Ontology

In other words, we can define the Ontology as follows:

From figures above, which are self-explanatory, the output of Finance and Management department is actually the Input of Data Warehouse (formerly Information Technology and Data Management) while they receive same output after processing.

Chapter 7 Evaluation

7.1 Scenario

Align Subsets of Information Systems Ontology with Organization Structure and Data-Warehouse Ontology, In Organization Structure Ontology, we have a class of Productivity, Infrastructure, and Legal. Similarly for Information Systems, we are having the classes of System Integration and Information Integration. For Data-warehouse Ontology, we have classes of Information, Scenario and Data-Models.

7.2 Alignment: Traditional Method

Through traditional method, we can see in our models designed earlier, that classes of Productivity, Infrastructure, and Legal in Organization Structure Ontology cannot be reutilized in either Information Systems classes as well as Data-warehousing structurally, programmatically and semantically. The same is true for the Data-warehousing where we have classes of Information, Scenario and data-models, among which only Information class is syntactically available in Information System~ System Integration, which is semantically decline the mapping process as well. In other words, all objects are Unaligned and cannot be mapped to make the Ontology well aligned

If we assign efforts equals 1 to each class of Ontology, the sum of efforts will become as follows:

Ontology	**Efforts (Summation = 8)**
Organization Structure	3
Information Engineering	2
Data-Warehousing	3

The same after Alignment can be made as follows:

Ontology	**Efforts (Summation = 7)**
Organization Structure	3
Information Engineering	2
Data-Warehousing	2

Therefore, after applying traditional methods, we reduced the Ontology efforts by 1 however, only Information Engineering ~ Information Integration and Data-warehousing ~ Information are combined into one, while Organization Stucture will remain a different Ontology.

The runtime analysis will remain (3x2x3)^3 = 5,832 times checking of the attributes to make and provide aligned solution.

7.3 Alignment: Rely-Guarantee

With our proposed method, we first assign efforts the same as what we assigned in aforementioned section when the Ontology was unaligned

Ontology	**Efforts (Summation = 8)**
Organization Structure	3
Information Engineering	2
Data-Warehousing	3

After going through rely-guarantee alignment approach, we have the following changes in the semantics, and thus coming up with the 1 Ontology

With the application of rely-guarantee, we can come up with the integrated Ontology, therefore making the Ontology entire dependent and therefore well aligned by means of mapping. We can see in our Organization Ontology, where Organization Structure ~ Productivity can only be measure by means of Information System ~ Information Integration. Information Integration is also dependent on Data-warehouse ~ Information and therefore feeding data to it. They all are 1 sub-set of Ontology. Similarly, when it comes to Organization Structure ~ Legal, we have policies in Data-warehouse ~ Policies which are semantically related and dependent as being fed by Organizations' policies, therefore they are considered as single Ontology. Now for the Data-warehousing ~ data-models, they are being fed by System ~ Information Integration, and therefore can be considered as single sub-set of Ontology.

We can clearly see from above that through rely-guarantee approach, we have data-feeders, the others are therefore dependent, combining into sub-set of Ontology. With the integration of all of the sub-sets of Ontology, the final Ontology would be called Organization Information and having the efforts as:

Ontology	**Efforts (Summation = 3)**
Organization Information	3

As we can see that Ontology is completely aligned into single Ontology, therefore well-aligned in a proper way.

With runtime analysis, we have 3^3 = 27, which is drastically reduced as compared to traditional alignment which was having a runtime of 5,832, when matching each attribute.

7.4 Algorithm

The following is the algorithm to come up with the proposed way of aligning Ontology

Declare Aligned Classes Container **O**

Declare Checking Container **A**

Declare Checking Container **B**

Loop - For each Class **X** in an Ontology

Gather Input Parameters

Gather Output Parameters

Add collected parameters from step (4.a) and step (4.b) in Container **A**, declared in step (2)

Start Nested Loop – Traverse to Entire Ontology

Gather Input Parameters

Gather Output Parameters

Add collected parameters from step (4.d.i) and step (4.d.ii) in Container **B**, declared in step (2)

Compare Container **A** and Container **B**

IF Equals

Add Class **X** to Container **O**

End IF

End Nester Loop

End Loop

Loop - For each Class **Y** in an Ontology

Check existence of Class **Y** in Container **O**

IF exists

Replace Class **Y** with reference value from Container **O**

END IF

End Loop

The above algorithm can be created easily in JAVA or similar Object Oriented Language by their expert, and therefore can be integrated with any OWL tool, e.g. Protégé.

7.5 Application in Protégé

7.5.1 Utilization

There are different tools available in the market that can create OWL Ontology of the specific scenario/domain to create interoperability in semantic web. There are different add-ons that can help in aligning the ontology, but all in traditional way. We checked different websites that publishes add-ons for Protégé. After the observation there was not an algorithm which is similar to ass/com method. As computer based solution runs on the systems that are perfectly provided in the frame of which the computer program understands, it is always difficult to architect data in a way to develop ontology, the basis of which will be the computer program to let it align.

7.5.2 Reason to Incorporate

Applications that actually proposes the Ontology Alignment, but all by through the traditional methods. Although there should be a direct availability of such feature within the Main tabs of the software e.g. Protégé. This is because, alignment is the main part of any ontology, while if the ontology is not aligned, there are high chances of duplication of the data types. Thus each data can be associated with different data type and thus can have different semantics while it will not be functional as interoperable for globally published classes.

However, as other traditional method which includes Semantic method, Structural Method, Terminological Method, and Expansion Method, are provided as an add-on to the Protégé and other OWL Ontology development toolkits, and Ass/Com is a way better way to align the ontology, which

make almost perfectly interoperable classes over semantic web for a better exchange of information, we consider it to be the must have tool for the OWL Ontology developer to let it align on the fly.

The other advantage will be that it can easily interconnect with other Business Process Management (BPM) tools, to let it known the inputs and outputs, while integrated systems, actors and position types can better be analyzed and integrate for effective communication.

We therefore recommend to have ass/com method as a mandatory tool for Ontology Alignment within the standard system to enable OWL Ontology developers to align their Ontology on the fly, and publish it so that the others can utilize the same, while information can be published over semantic web, making each attribute unique for better searching capabilities and information interoperability.

7.6 Conclusion

It can concluded with the statement that there are traditional methods which can help us in aligning the ontology; however, there isn't a direct algorithm that can directly align ontologies. We have introduced Rely/Guarantee method to actually align the ontology. Although, extensive dictionary is required to be published for global accessibility with proper algorithm to manage, so alignment by any means can be possible through traditional ways. However, there is a need of refined algorithms to align ontology. With the proposed method, it may be difficult to align however, alignment can be achieved far better than the traditional. It's also implementable, as their algorithm can easily be developed. Thus, if classes satisfies the environment and so as its consequent classes, there is a possibility to align the entire ontology through Rely/Guarantee, according to which, final process state will also satisfy all the environmental factors.

Ontologies can be aligned through traditional methods. However, it may not result in an efficient manner as every method has its drawbacks and doesn't completely align the ontology. With our proposed method of introducing "parallel computing" method in the area of Ontology alignment, we are sure to provide grounds to align Ontology in a more effective manner by not actually checking the semantics, but comparing the Inputs and Outputs of the system can be aligned. Also explanations of an examples from real-life while proposed method of how effectively can it be implemented through Ass/Com language.

Chapter 8 Proposed Utilities to Potential Tool Components

8.1 Introduction

From earlier section we concluded that we can come up with the aligned ontologies globally with the effective utilization of practices, developing such an interconnected environment that help coming up with globally accepted ontologies, and different toolkits to make it easy for the developer in order to design originally as per best practices, while content should be made available as an open-source, so as to be available for further optimization of different scenarios. There is a need to add a new toolkit to actually come up with the realized solution. We earlier provided algorithm of how the ontology will be working however, there are different more aspects that a developer should know in while implementing the solution to design such a toolkit that can function the algorithm effectively. We have discussed our suggestions later in the section.

8.2 Centralized Body for Governance

To make the Ontology for semantic world effective, there should be a single governance body who can monitor all the semantic web articles, including Ontology, while there should be a single repository where everyone can post their Ontologies. The same will be published and made available so as to come up with the interconnected semantic web. At current, as there isn't any scrutinizing procedure available, anyone can design and let the world know of the designed ontology. However, there should be a check-n-balance in order to come up with Ontology for specific scenario. This enforces reutilization of the same Ontology, rather generating different ontologies for the same purpose. At other side, rather coming up with different Ontologies for same purpose, that single Ontology can be made optimized so as to have such a comprehensive scenario that can interconnect with Ontology of other domain in order to make the complete Ontology well aligned.

With our proposed algorithm of rely-guarantee approach to align ontology, this will make our semi-automatic approach as completely automated. This is because there will be a single repository where all metadata would exists. Thus, when Ontology design engineer come up with new design, the same will be uploaded while it will be verified online so as to get acknowledgment if the person is working on new stuff, or reinventing the wheel.

8.3 Development Environment

For a proper utilization of any new technology, what matters is the development environment. It is the environment that either can prosper the technology or laid it down. For instance, if we analyze business process re-engineering, which is a great advancement in the field of resource planning, was evolved after having a stable market for Enterprise Resource Planning. Same is the case with ERPs, which evolved with the maturity of Material Resource Planning, which again was evolved with the stability of technology in information systems.

For semantic world, there needs to be a stable environment to have consolidated ontologies at single place. In other words, having a single repository of different Ontologies.

For a proper environment for the Ontology, for which we propose to have automated reconciliation and alignment, the following parameters should be considered well before going after introducing the method.

8.3.1 Semantic Web

Unlike having a current platform to have design and print the Ontology on papers, there is a need to publish at global level. This is required to have access of single Ontology to different engineers around the globe. The same therefore can be aligned effectively, as being available to all the engineers. Further, there is no one who is well aware of each and every domain. Like SAP, which came up with different Information Systems for each domain, they completed their project in more than 15 years. Same is the case with Ontology. With the semantic environment to have, we need to have people around the global with different cultural background as well as from different domain. A consistent semantic environment can then be created by incorporating their local policies, policies pertaining particular domain, and in-depth areas of the domain.

Therefore, for a consistent and sharing knowledge of Ontology, there is a need to put effort to come up with stable platform of sharing, where the developed Ontologies can be optimized.

8.3.2 Web Crawlers

For search giants, web crawlers are the key to sense the terms, being utilized world-wide. The same is utilized in search engines to show the results corresponding to the search terms for their end-users. Similar technology can be adopted for the Semantic world, where whenever the term is searched or uploaded on the portal, the crawlers gets updated. The same then be utilized whenever new

development is being done, so the agent can automatically indicate of the availability of same semantics, which make easy for the developer not to work over already developed Ontology, rather optimize the already available one. The platform there will help coming up with the Ontologies of different areas, while can be easily integrated with other processes to streamline them and make the entire Ontology well-aligned

8.3.3 Security Risks and Strategies

With the platform to provide interoperability of data-objects in order to have interconnected processes after Ontology alignment, the main concern is of the security. If there becomes strong dependency over the technology, the security concerns will be raised, and are of high consideration. For the reason, there should be a body to maintain well encrypted medium for interoperability of aligned Ontologies, to get data to and from interconnected Ontologies. We recommend to have RSA encrypted technique in order to get the better security, which is fairly vulnerable.

This is required to gain not only the customers to adopt the technology, but also for the developers to gain confidence in them in order to get into the technology and fully utilize it. Further, this will bring new eras for online business in order to get the things work in lesser time, and with improved data flow and connectivity.

We therefore like to focus on more over the strategies in future work so as to consider the security breaches, which can be overcome by following devised strategies.

8.3.4 Consultancy Firms

With the evolution of technology, the main part is not only to market but to train the individual to utilize it. With lots of benefits of aligned Ontology, we need to have a platform where individuals can exchange their concerns and equally resolves it. Further, it is required to engage consultancy firms, and train them, in order to introduce the concept with their customers, which we anticipate that the idea will be bought, because of many advantages which companies thinks of. The idea should also be penetrated within the varsity students so as to think them in this direction when stepping into the real-economic-world, to gain benefit and rule the market.

8.3.5 Online Marketplace APIs

The Application Programming Interfaces should be introduced to the open-source platform in order to have a space to have more research in designing the algorithm, while equally optimizing it in order to come up with better solution, with the utilization of different resources, introducing frequently.

Such marketplace can also be served as a buy-n-sell of the designed APIs, so it can provide opportunities for new-comers to sell their concepts of Ontology, which may provide benefit to specific industry etc.

8.3.6 Introduction of Best Practices

The body, as recommended, should not only maintain the mentioned platform, but also should work on developing the community from various domains, while with the mutual understanding, introduce best practices for aligned Ontology through Rely-Guarantee approach. Unlike traditional methods, which cannot further be enhanced as much of the work has already been done for more than 20 years, they should start working on the proposed alignment methodology in order to make the most of it in the industry, while making the platform to have personnel from various domains. Together, they can come up with the best practices of different domains, and can introduce standard Ontologies. The same then can be utilized, while with the help of feedback, it can be enhanced further by time to get the stable platform for reutilization of aligned Ontology.

8.4 Integration Tools

We suggest having a separate tool for Ontology that functions on the following main parts

1. Availability of Deployed Attributes over Semantic Web
2. Availability of Deployed Classes over Semantic Web
3. OWL Ontology Importer
4. Integration with BPM Tools
5. Ass/Com Alignment

8.4.1 Availability of Deployed Attributes over Semantic Web

By availability of deployed attributes, we mean to have a tool that can connect to semantic web server and able to access all attributes to reutilize the attribute within our class, having same semantics.

With that said, it means to have a single repository, may be a data-warehousing solution, which contains all attributes, classes and Ontologies with the meta data, so as to do Business Intelligence in order to get the possible semantic Ontology of the design being used. The same then can be effectively utilized by

means of indexing terminologies to quickly provide new Ontologies, if they are already in-place and don't need to be further implemented.

8.4.2 Availability of Deployed Classes over Semantic Web

By availability of deployed classes over semantic web, we mean to have a tool that not only have access to globally published attributes but also the classes to reutilize the properties defined within the class, to instantiate object of particular class, thus having same semantic object, related to same class. Thus interoperability is achievable.

Same is the case here as what we discussed earlier. For that we need to maintain a metadata for each semantic class, so as to get it well classified. The same then need to be configured in Data-warehousing solution for analytics and auto-assignment of already developed Ontologies. This enable mapping the classes having same semantics, while this will make our proposed algorithm automated, as the concept will be inherited and mapped of the semantics, rather going to attribute level.

8.4.3 OWL Ontology Importer

By OWL Ontology importer, we mean to have a functionality within a tool, that can import already defined OWL Ontology, and after parsing each attribute and classes while importing, provide suggestions of the replaceable objects, to directly convert it into the globally published attributes/classes.

This is only possible if the classes are defined well in our repository. For the importer, it is mandatory to have solution to have consistent database from which it can be passed while parsing the elements and replace all attributes/classes which are semantically same, or are dependent on some other class, so as to feed data, and to make the algorithm run automatically.

8.4.4 Integration with BPM Tools

By integration with business process management (BPM) tool, we mean to have direct connectivity with the tool, to pick the connected resources, so as to identify pattern of which object is actually initiating the project, and which object is at the receiving object. Thus, with such information, we can have known semantics of the particular class or attribute

8.4.5 Ass/Com Alignment

By our proposed method, ass/com algorithm provided in the earlier section, we can align the ontology by actually analyzing the dependent objects of particular class, that what input the class has been

receiving, while what output the class is providing, who is actually sending the input information and who is receiving the processed information for further processing. This helps in identifying the pattern and thus proposing the aligned ontology, as discussed above in earlier sections.

8.4.6 Benchmarking

There should be a component to evaluate the designed ontology, while benchmarking it with the globally approved and published Ontology. If it gets up to the mark, consider for the next best practice, while if it is not, propose ways to make it optimize. With the mutual learning in the beginning will let the Ontology gets up to benchmark, while benchmarking will also get matured by time.

Chapter 9 Conclusion and Future Work

9.1 Conclusion

The goal of This research was to solve the a problem of ontology alignment, thus enabling semantic interoperability between different domains this thesis introduces a framework based on ontologies and their alignment. More specifically, a new generic approach has been developed In the Semantic Web, the area of particular concern to this study, data will be extracted from many different ontological structures, and information processing across ontologies is not possible without knowledge of the semantic mappings between them. Therefore, semantic alignment between ontologies is a necessary precondition for discovering the alignment across ontologies or establishing interoperability between agents or services using different ontologies. In the field of ontology matching, one of the main issues is the need for flexible algorithms and tools, capable of adapting to different domains and also to different interpretations of the notions of alignment and similarity. Our approach mplements a variety of techniques that are based on terminology, which is itself based on the study of words in conceptual or structural labels, which are in turn based on the relative positions of concepts in the taxonomies.

Recent research has focused on the development of quality alignment techniques and methods, however, there has been little attention paid to how un-aligned classes can be aligned. Has been conclude that by using Ass./Com approach not only ensures the ontology alignment, but also let the semantic web converge to get the systems connected. Which was demonstrated on our two designed otnologies while compared its results with application of traditional alignment methods

It has thesis achieved the following :

An in–depth review to the definition and structure of ontology. Also, various operations over ontologies and a different set of ontology matching methods have been proposed A detailed review state-of-the-art of the ontology languages which are used to express ontology over the Web; all these terms were shown in order to provide a basic understanding of ontologies and of description logics, which are the basis of ontology languages. A comprehensive review of existing ontology alignment tools. Several existing ontology mapping methods were analysed and compared, since before creating a new approach it is essential to fully understand related work. Here, this means both theoretical work and existing approaches to the alignment of ontologies or other well-defined structures, including their weaknesses, which must be understood if they are to be improved.

An analyses of business and engineering organizations hierarchy structure, has been semantically model to show the advantages in integrated semantics to business process Semantic modelling it allows Business users to be more specific by adding more formality and ensuring consistency in business definitions. In essence this mean it describes your business terms and relating them to each other , inserting them into a hierarchy / taxonomy and adding rules Semantic data model serves that purpose. It's a user-oriented metadata layer that comes on top of the data and enriches it. Business developers model the data in such a way that it represents something meaningful to the end-users. The quality of the analytical solution depends on two things: the richness or the capabilities of the underlying semantic model and the knowledge business developer possesses regarding that model.

Rely/ guarantee which is the technique used in parallel computing this research redefines it's use for Ontology Alignment in Algorithmic Way .The method assumes that not only a component is verified by just satisfying all of its commitments, but also verifies all the assumptions, being exposed from the internal or external environment. We adopted the idea and the paradigm so that the method can now be applied to business domain knowledge expression for machine learning /intelligent understands purpose.

9.2 Future work

The following topics, arising out of this thesis, seem worthy of further research.

- Based on the work done in this research, especially the case study, future researchers could make meta-data to bring external stakeholders to the ground to interconnect with organizations for better evaluation of their systems and streamline connectivity for optimized processing within inter-industry. In parallel, social networking can be brought to semantic world for resource hiring and outsourcing, so the systems can automatically finds best people around the globe.

- The algorithm in this thesis could be implemented as utility to integrated for Ontology designing software Protégé

- More case studies with different domain areas other than business and engineering can be carried out to further extend and adjust the ass/com framework.

- since the current tool used is only a prototype system, more work can be done to fully implement the tool with more complete functions

References

Aleksovski, Z., Klein, M., Ten Kate, W., van, F. (2006) *Harmelen:* Matching Unstructured Vocabulariesusing a Background Ontology In Proceedings of EKAW.

Antoniou, G., (2004) *Web Ontology Language: OWL*, Presented at Handbook on Ontologies, pp.67-92.

Barwise, J., & Seligma, J., (1997) *Jon Barwise and Jerry Seligman. Information flow:* the logic of distributed systems, volume 44 of Cambridge Tracts in Theoretical Computer Science.

Barwise, J., & Seligman, J., (1997) *Information Flow:* the logic of distributed systems.

Beckett, D. (2000) *The Design and Implementation of the Redland RDF Application Framework*, In Proceedings of 10th International World Wide Web Conference, Hong Kong.

Bench-Capon, Trevor J. M. & Malcolm, G., (1999) *Formalising ontologiesand their relations.* Proceedings of the Tenth International Conference and Workshop on Database and Expert Systems Applications pp. 250-259.

Berners-Lee, Hendler, J., & Lassila., Q. (2001) *The Semantic Web*, Scientific Am, May pp. 34–43.

Ciocoiu, M., Nau, S., & Gruninger, M., (2001) *Ontologies for integrating engineering applications:* Journal of Computing and Information Science in Engineering pp. 12-22.

Corcho, O., & Gómez-Pérez. A. (2001) *Solving Integration Problems of E-Commerce Standards and Initiatives through Ontological Mappings*, In Proceedings of IJCAI 2001 Workshop on E-Business & the Intelligent Web, Seattle, USA.

De Bruijn, J., Ehrig, M., Feier, C., Martm-Recuerda, F., Scharffe, F., & Weiten, M., (2006). *Ontology mediation, merging, and aligning*.

De Bruijn, J., Ehrig, M., Martin-Recuerda, F., PoUeres, A., & Predoiu, L., (2005) SEKT deliverable D4.4.1: Ontology mediation management vl.

Dean, M., Schreiber., G. (2003) *OWL Web Ontology Language Reference*. W3C Recommendation

Dean, M.F., Connolly, D., van Harmelen, F., & Patel-Schneider. (2002) *OWL Web Ontology Language 1.0 Reference*

Ding, Ying, Fensel, Dieter, Klein, Michel C. A., & Omelayenko, B., (2002) *The Semantic Web:* yet another hip? Data Knowledge Engineering, pp. 205-227.

Doan, A., Domingos, P., & Halevy, A., (2003). *Learning to match the schemas of data sources:* A multistrategy approach. VLDB Journal, pp. 279-301.

Doan, J., & Madhavan, D., (2002) *Learning to Map Between Ontologies on the Semantic Web*, In Proceedings of WWW, pp.662-673.

Euzenat, J., Scharffe, F., & Serafini, L., (2006) Knowledge Web deliverable D2.2.6: Specification of the delivery alignment format.

Fensel, D. (2001) *Ontologies:* Silver Bullet for Knowledge Management and Electronic Commerce, Springer.

Fensel, D., McGuinness, D.L., & Patel-Schneider. (2001), *OIL: An Ontology Infrastructure for the Semantic Web*, Presented at IEEE Intelligent Systems, pp.38-45.

Fisher, M., Dean M., & Joine, G. (2008) *Use of OWL and SWRL for Semantic Relational Database Translation, Arlington.*

Ganter, B., & Wille, R., (1999). *Formal Concept Analysis:* Mathematical Foundations. Springer.

Giunchiglia, F., & Shvaiko, Y,. (2004) *S-match:* an algorithm and an implementation of semantic matching. In Bussler C., Davies J., Fensel D., and Studer R., editors, The semantic web: research and applications.

Giunchiglia, F., Yatskevich, M., & Giunchiglia, E., (2005) *Efficient semantic matching*: In Proceedings of the 2nd european semantic web conference.

Gotoh, O., (1982) An *Improved Algorithm for Matching Biological Sequences*, Presented at Journal of Molecular Biology, 162:705-708.

Gruber., T. (1993*) Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, In International Journal Human-Computer Studies 43, pp 907-928.

Hitzler, P., Krotzsch, M., Ehrig, M., & Sure, Y., (2005). *What is ontology merging:* A category-theoretical perspective using pushouts pp. 104-107.

Jannink, J., Pichai, S., Verheijen, D., & Wiederhold, G., (1998). Encapsulation and composition of ontologies.

Kalfoglou, Y., and Schorlemmer, M., (2003) *Ontology mapping*: the state of the art. Knowledge Engineering Review, pp. 1-31.

Kalfoglou, Y., and Schorlemmer, M., (2002) Information-flow-based ontology mapping. pp. 1132-1151.

Kalfoglou, Y., and Schorlemmer, M., (2003) *If-map:* an ontology mapping method based on information flow theory. Journal of data semantics pp. 1:98–127.

Kent, & Robert E. (2000). *Dynamism and Stability in Knowledge Organization:* Proceedings of the Sixth International Conference of the International Society in Information Systems (ISKO), Toronto, Canada. Ergon- Verlag.

Kotis, K., & Vouros, G.A (2004) *HCONE approach to ontology merging.* In Proc. 1st European Semantic Web Symposium (ESWS), volume 3053 of Lecture notes in computer science, pp. 137–151.

Kotis, K., Vouros, G.A , & Stergiou, K., (2006) Towards automatic merging of domain ontologies: The HCONE-merge approach. Journal of Web Semantics (JWS), pp. 60–79.

Lassila, O., Swick., R. (1999), *Resource Description Framework (RDF) model and syntax specification* (http://www.w3.org/TR/REC-rdf-syntax)

Maedche, A., (2001) '*Ontology Learning for the Semantic Web'* Presented at IEEE Intelligent Systems, pp.72-79

McBride., B., Staab, S., Studer, R. (2003) *The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS, in:* The Handbook on Ontologies in Information Systems, Springer-Verlag.

McGuinness, D.L. , Fikes, R., Stein, L.A. (2003) DAML-ONT: An Ontology Language for the Semantic Web", In Proceedings of Spinning the Semantic Web, pp.65-93.

McGuinness, D.L., van Harmelen, F. (2004) *OWL Web Ontology Language Overview:* Technical report, W3C, (http://www.w3.org/TR/owl-features)

McGuinness, Deborah L., Fikes, R., Rice, J., & Wilder, S., (2000). *The Chimaera ontology environment.* pp. 1123-1124, Austin, TX, USA.

Mihoubi., H., Simonet., A. (2000) *An Ontology Driven Approach to Ontology Translation*, In Proceedings of DEXA, pp.573-582.

Miller., G., A. (1995) *WordNet:* A Lexical Database for English", presented at Commun. ACM, pp.3941.

Mitra, P., & Wiederhold, G., (2002) *Resolving terminological heterogeneity in ontologies.*

Mitra, P., & Wiederhold, G. (2001) *An ontology-composition algebra*. Technical report, Stanford University, Stanford, CA, USA.

Mitra, P., , Wiederhold, G., & Kersten, M., (2000) A graphoriented model for articulation of ontology interdependencies, pp. 86-1

Noy, H., & McGuinness, D., (n.p) Ontology *Development 101: A guide to creating your first ontology*, Technical Report, Stanford University, Stanford, CA, US.

Noy, Natasha F. and Musen, & Mark A. (2002) *PromptDifi':* a fixed-point algorithm for comparing ontology versions. In Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02), pp. 744-750.

Noy, Natasha F., Musen, & Mark A. (2001) *Anchor-PROMPT:* Using nonlocal context for semantic matching. In Workshop on Ontologies and Information Sharing at the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001), pp. 63-70.

Noy, Natasha F., Musen, & Mark A. (2000) *PROMPT:* Algorithm and tool for automated ontology merging and alignment. In Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), pp. 450-455.

O'Connor, M., Knublauch, H., Samson, T., & Musen, M. (2005) *Writing Rules for the Semantic Web Using SWRL and Jess*.

Paolucci., M., Kawamura., T., Payne, & Sycara, K., (2002) *Semantic Matching of Web Services Capabilities*, In Proceedings of International Semantic Web Conference, pp. 333-347.

Patel-Schneider, P.F., & Horrocks., I. (2000) Reference Description of the DAML+OIL Ontology Markup Language (http://www.daml.org/2000/12/reference.html)

Patel-Schneider, P. F., Hayes,& P. Horrocks. (2004) *OWL Web Ontology Language: Semantics and Abstract Syntax*: W3C Recommendation (Available via http://www.w3.org/TR/2003/WD-owl-semantics-20030331/ , last visited March 2009).

Popa, L., Velegrakis, Yannis, Miller, Renee J., Hernandez, Mauricio A., Fagin, & Ronald. (2002) Translating web data. In Bernstein et al. pp. 598-609.

Quix C., Roy P., D. (2011) *Kensche:* Automatic Selection of Background Knowledge for Ontology Matching. 3rd International Workshop on Semantic Web Information Management.

Rahm, E., & Bernstein, A. (2001) *Journal:* Very Large Data Bases, pp. 334-350.

Rahm, P. A. (2001) *Bernstein:* A survey of approaches to automatic schema matching, The VLDB Journal 10: pp 334–350.

Shvaiko, J. (2008) *Euzenat:* Ten Challenges for Ontology Matching In Proceedings of ODBASE.

Shvaiko, P., & Euzenat, J., (2005) *A survey of schema-based matching approaches*: Journal on Data Semantics (JoDS) pp. 146-171.

Smith, M., Welty, C. (n.p) OWL *Web Ontology Language Guide* (http://www.w3.org/TR/2003/WD-owl-guide-20030331/)

Stumme, G., & Maedche, Al., (2001). *FCA-Merge:* Bottom-up merging of ontologies, pp. 225-230.

Taye., M. (2009) *Ontology Alignment Mechanisms for Improving Web-based Searching*, PhD thesis, De Montfort University, Leicester, UK.

Tidwell., D. (2000) *Web Services-The Web's Next Revolution*, IBM Web Service Tutorial, (http://www-106.ibm.com/developerworks/edu/ws-dw-wsbasics-i.html) [29 Nov. 2000]

Uschold, M., & Gruninger, M., (2004) *Ontologies:* Principles, Methods and Applications", Knowledge Engineering Review, vol. 11, no. 2.

Vögele, T., Visser, U., & Stuckenschmidt, H. (2001) *Ontology-based integration of information - a survey of existing approache*s, In Workshop: Ontologies and Information Sharing, pp. 108-117.

Vouros, G., & Kotis, K., (2005) *In Proc. 2nd European Semantic Web Conference (ESWC)* pp. 198–210.

Wiederhold, G,. (1994). *An algebra for ontology composition*: In Proceedings of 1994 Monterey Workshop on formal Methods, U.S. Naval Postgraduate School, Monterey, CA, USA. Pp. 56-61.

Appendix A

Full Owl code – Business

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#" x
mlns:xsp="http://www.owl-
ontologies.com/2005/08/07/xsp.owl#" xmlns:owl="http://www.w3.org/2002/
07/owl#"xmlns="http://www.owl-
ontologies.com/Ontology1344155193.owl#"xmlns:xsd="http://www.w3.org/20
01/XMLSchema#" xmlns:swrl="http://www.w3.org/2003/11/swrl#"xmlns:swrlb
="http://www.w3.org/2003/11/swrlb#" xmlns:rdfs="http://www.w3.org/2000
/01/rdf-schema#"xml:base="http://www.owl-
ontologies.com/Ontology1344155193.owl">
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="Contracts">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasProperties"/>
</owl:onProperty>
<owl:someValuesFrom>
<owl:Class rdf:ID="Department_Base"/>
</owl:someValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:ID="Procurement"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Monitoring">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hassome"/>
</owl:onProperty>
<owl:someValuesFrom>
<owl:Class rdf:ID="System"/>
</owl:someValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:ID="Management"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Frameworkds">
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom>
<owl:Class rdf:about="#Department_Base"/>
```

```
</owl:someValuesFrom>
<owl:onProperty rdf:resource="#hasProperties"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="#Procurement"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="CustomerRelationshipManagement">
<rdfs:subClassOf>
<owl:Class rdf:ID="Optimization"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Trend">
<rdfs:subClassOf>
<owl:Class rdf:about="#System"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Department_Base">
<rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasProperties"/>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Management">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasProperties"/>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:ID="Financials"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#System">
<rdfs:subClassOf>
<owl:Class rdf:ID="TechnologyBase"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Legal">
<rdfs:subClassOf>
<owl:Class rdf:ID="External"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Report">
<rdfs:subClassOf rdf:resource="#System"/>
</owl:Class>
```

```
<owl:Class rdf:ID="Analytics">
<rdfs:subClassOf rdf:resource="#System"/>
</owl:Class>
<owl:Class rdf:ID="Corporate">
<rdfs:subClassOf>
<owl:Class rdf:ID="Internal"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="BusinessProcessManagement">
<rdfs:subClassOf>
<owl:Class rdf:about="#Optimization"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Mining">
<rdfs:subClassOf rdf:resource="#System"/>
</owl:Class>
<owl:Class rdf:ID="Training">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasProperties"/>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:ID="HumanResource"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="HSE">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasProperties"/>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#Legal"/>
</owl:Class>
<owl:Class rdf:ID="Product_Service">
<rdfs:subClassOf rdf:resource="#Management"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
<owl:onProperty rdf:resource="#hasProperties"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Ethical">
<rdfs:subClassOf>
<owl:Class rdf:about="#External"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Customers">
```

```
<rdfs:subClassOf>
<owl:Class rdf:ID="Sales"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Section_Base">
<rdfs:subClassOf rdf:resource="#Department_Base"/>
</owl:Class>
<owl:Class rdf:ID="Decision">
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom rdf:resource="#System"/>
<owl:onProperty rdf:resource="#hassome"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#Management"/>
</owl:Class>
<owl:Class rdf:about="#Sales">
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
<owl:onProperty rdf:resource="#hasProperties"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="#Financials"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Internal">
<rdfs:subClassOf>
<owl:Class rdf:ID="Functions"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Furniture">
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
<owl:onProperty rdf:resource="#hasProperties"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:ID="Infrastructure"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Infrastructure">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasProperties"/>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#Corporate"/>
```

```
</owl:Class>
<owl:Class rdf:ID="SupplierRelationshipManagement">
<rdfs:subClassOf>
<owl:Class rdf:about="#Optimization"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#External">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="Services">
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
<owl:onProperty rdf:resource="#hasProperties"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#Infrastructure"/>
</owl:Class>
<owl:Class rdf:about="#Financials">
<rdfs:subClassOf rdf:resource="#Internal"/>
</owl:Class>
<owl:Class rdf:ID="Distribution">
<rdfs:subClassOf rdf:resource="#Sales"/>
</owl:Class>
<owl:Class rdf:ID="Recruitment">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasProperties"/>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="#HumanResource"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#HumanResource">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasProperties"/>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#Financials"/>
</owl:Class>
<owl:Class rdf:ID="RawMaterial">
<rdfs:subClassOf rdf:resource="#Product_Service"/>
</owl:Class>
<owl:Class rdf:ID="Accounting">
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
```

```
<owl:onProperty rdf:resource="#hasProperties"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#Legal"/>
</owl:Class>
<owl:Class rdf:ID="MarketResearch">
<rdfs:subClassOf rdf:resource="#Sales"/>
</owl:Class>
<owl:Class rdf:ID="Marketing">
<rdfs:subClassOf rdf:resource="#Sales"/>
</owl:Class>
<owl:Class rdf:ID="Compensation">
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
<owl:onProperty rdf:resource="#hasProperties"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#HumanResource"/>
</owl:Class>
<owl:Class rdf:about="#Optimization">
<rdfs:subClassOf rdf:resource="#Management"/>
</owl:Class>
<owl:Class rdf:ID="Licensing">
<rdfs:subClassOf>
<owl:Class rdf:ID="Development"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Welfare">
<rdfs:subClassOf rdf:resource="#Ethical"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasProperties"/>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Warehouse">
<rdfs:subClassOf>
<owl:Restriction>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
<owl:onProperty rdf:resource="#hasProperties"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#Infrastructure"/>
</owl:Class>
<owl:Class rdf:about="#Procurement">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasProperties"/>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
```

```
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#Corporate"/>
</owl:Class>
<owl:Class rdf:ID="HR">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasProperties"/>
<owl:someValuesFrom rdf:resource="#Department_Base"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#Legal"/>
</owl:Class>
<owl:Class rdf:ID="Positioning">
<rdfs:subClassOf rdf:resource="#Product_Service"/>
</owl:Class>
<owl:Class rdf:ID="Processing">
<rdfs:subClassOf rdf:resource="#Sales"/>
</owl:Class>
<owl:Class rdf:ID="Employees">
<rdfs:subClassOf rdf:resource="#Section_Base"/>
</owl:Class>
<owl:Class rdf:ID="Selling">
<rdfs:subClassOf rdf:resource="#Sales"/>
</owl:Class>
<owl:Class rdf:ID="Facilities">
<rdfs:subClassOf>
<owl:Class rdf:about="#Development"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Dashboard">
<rdfs:subClassOf rdf:resource="#System"/>
</owl:Class>
<owl:Class rdf:ID="Innovation">
<rdfs:subClassOf rdf:resource="#Product_Service"/>
</owl:Class>
<owl:Class rdf:about="#Development">
<rdfs:subClassOf rdf:resource="#Management"/>
</owl:Class>
<owl:Class rdf:ID="SkillSet">
<rdfs:subClassOf rdf:resource="#Product_Service"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasHead">
<rdfs:subPropertyOf rdf:resource="#hasProperties"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasManagers">
<rdfs:subPropertyOf rdf:resource="#hasProperties"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSectionHead"/>
<owl:ObjectProperty rdf:ID="hasEmployees">
<rdfs:subPropertyOf rdf:resource="#hasProperties"/>
```

```
</owl:ObjectProperty>
<owl:AllDifferent/>
<System rdf:ID="tchTrending"/>
<System rdf:ID="tchAnalytic"/>
<System rdf:ID="tchReport"/>
<System rdf:ID="tchDashboard"/>
<System rdf:ID="tchMining"/>
</rdf:RDF>
<!--
Created with Protege (with OWL Plugin 3.4.8, Build 629)
http://protege.stanford.edu
-->
```

Appendix B

Full Owl Code Engineering Model

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#" x
mlns:xsp="http://www.owl-
ontologies.com/2005/08/07/xsp.owl#" xmlns:owl="http://www.w3.org/2002/
07/owl#"xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:swrl="http
://www.w3.org/2003/11/swrl#"xmlns:swrlb="http://www.w3.org/2003/11/swr
lb#" xmlns="http://www.owl-
ontologies.com/Ontology1344589925.owl#" xmlns:rdfs="http://www.w3.org/
2000/01/rdf-schema#"xml:base="http://www.owl-
ontologies.com/Ontology1344589925.owl">
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="CheckOut">
<rdfs:subClassOf>
<owl:Class rdf:ID="Functions"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Move">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="InformationIntegration">
<rdfs:subClassOf>
<owl:Class rdf:ID="ACEEngineeringInformationManagement"/>
</rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:ID="InformationStorage">
<rdfs:subClassOf>
<owl:Class rdf:ID="InformationManagement"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Specify_Storage_Location">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="Define_Product">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="ProcessIntegration">
<rdfs:subClassOf rdf:resource="#ACEEngineeringInformationManagement"/>
</owl:Class>
<owl:Class rdf:ID="ProductProcessItemDefinition">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:ID="hasFunction"/>
</owl:onProperty>
<owl:allValuesFrom rdf:resource="#Specify_Storage_Location"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:ID="Specify_Relationship"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="#hasFunction"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:ID="Define_Component"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="#hasFunction"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="#Define_Product"/>
<owl:onProperty rdf:resource="#hasFunction"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:ID="InformationDefinition"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Communication">
```

```
<rdfs:subClassOf rdf:resource="#ACEEngineeringInformationManagement"/>
</owl:Class>
<owl:Class rdf:ID="CheckIn">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="StructuredItemDefinition">
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:ID="DefineAssociation"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="#hasFunction"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasFunction"/>
<owl:allValuesFrom>
<owl:Class rdf:ID="DefineInfoItem"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasFunction"/>
<owl:allValuesFrom rdf:resource="#Specify_Storage_Location"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="#InformationDefinition"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#InformationManagement">
<rdfs:subClassOf rdf:resource="#InformationIntegration"/>
</owl:Class>
<owl:Class rdf:ID="Import">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="Copy">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="PersonalWorkArea">
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:ID="PutAway"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="#hasFunction"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
```

```
<owl:Restriction>
<owl:onProperty rdf:resource="#hasFunction"/>
<owl:allValuesFrom rdf:resource="#Move"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#InformationStorage"/>
</owl:Class>
<owl:Class rdf:ID="CheckFlow">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:about="#InformationDefinition">
<rdfs:subClassOf rdf:resource="#InformationIntegration"/>
</owl:Class>
<owl:Class rdf:ID="IS_PersonalWorkArea">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasFunction"/>
<owl:allValuesFrom>
<owl:Class rdf:ID="Get"/>
</owl:allValuesFrom>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:ID="InformationSharing"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="InformationChange">
<rdfs:subClassOf rdf:resource="#InformationManagement"/>
</owl:Class>
<owl:Class rdf:ID="CheckStatus">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="IS_Library">
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="#CheckOut"/>
<owl:onProperty rdf:resource="#hasFunction"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:ID="Reference"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="#hasFunction"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="#Copy"/>
<owl:onProperty rdf:resource="#hasFunction"/>
```

```
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Class rdf:about="#InformationSharing"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ProcessInformationAssociation">
<rdfs:subClassOf rdf:resource="#InformationDefinition"/>
</owl:Class>
<owl:Class rdf:about="#InformationSharing">
<rdfs:subClassOf rdf:resource="#InformationManagement"/>
</owl:Class>
<owl:Class rdf:about="#Define_Component">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="InformationTracking">
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="#CheckFlow"/>
<owl:onProperty rdf:resource="#hasFunction"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom rdf:resource="#CheckStatus"/>
<owl:onProperty rdf:resource="#hasFunction"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#InformationManagement"/>
</owl:Class>
<owl:Class rdf:about="#PutAway">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="Append">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="IS_Database">
<rdfs:subClassOf>
<owl:Restriction>
<owl:allValuesFrom>
<owl:Class rdf:about="#Get"/>
</owl:allValuesFrom>
<owl:onProperty rdf:resource="#hasFunction"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#InformationSharing"/>
</owl:Class>
<owl:Class rdf:ID="Export">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:about="#Get">
```

```
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="Set">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="Database">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasFunction"/>
<owl:allValuesFrom rdf:resource="#Append"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#InformationStorage"/>
</owl:Class>
<owl:Class rdf:ID="Library">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasFunction"/>
<owl:allValuesFrom rdf:resource="#CheckIn"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#InformationStorage"/>
</owl:Class>
<owl:Class rdf:about="#Reference">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="Transmit">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:about="#Specify_Relationship">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="InformationDistribution">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasFunction"/>
<owl:allValuesFrom rdf:resource="#Transmit"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasFunction"/>
<owl:allValuesFrom rdf:resource="#Import"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource="#InformationManagement"/>
</owl:Class>
<owl:Class rdf:about="#DefineInfoItem">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
<owl:Class rdf:ID="SystemIntegration">
```

```
<rdfs:subClassOf rdf:resource="#ACEEngineeringInformationManagement"/>
</owl:Class>
<owl:Class rdf:about="#DefineAssociation">
<rdfs:subClassOf rdf:resource="#Functions"/>
</owl:Class>
</rdf:RDF>
<!--
Created with Protege (with OWL Plugin 3.4.8, Build 629)
http://protege.stanford.edu
-->
```

Appendix C

Full OWL code of the real-life example of Ontology

```
<rdf:RDF
 <owl:Class rdf:ID="Dashboard"/>
 <owl:Class rdf:ID="Management">
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:someValuesFrom>
     <owl:Class rdf:ID="Analytics"/>
    </owl:someValuesFrom>
    <owl:onProperty>
     <owl:ObjectProperty rdf:ID="requires"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:someValuesFrom rdf:resource="#Dashboard"/>
    <owl:onProperty rdf:resource="#requires"/>
```

```
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Organization"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="DataManagement">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="provides"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Analytics"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="develops"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Analytics"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Reports"/>
```

```
      </owl:allValuesFrom>
      <owl:onProperty rdf:resource="#develops"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#provides"/>
      <owl:allValuesFrom rdf:resource="#Reports"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Organization"/>
</owl:Class>
<owl:Class rdf:ID="Finance">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#requires"/>
      <owl:hasValue>
        <Reports rdf:ID="BalanceSheet"/>
      </owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:hasValue>
        <StatutaryFile rdf:ID="ProfitLossReport"/>
      </owl:hasValue>
      <owl:onProperty rdf:resource="#requires"/>
```

```
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Analytics"/>
      <owl:onProperty rdf:resource="#requires"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="StatutaryFile"/>
      </owl:allValuesFrom>
      <owl:onProperty rdf:resource="#requires"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Organization"/>
</owl:Class>
<owl:Class rdf:ID="InformationTechnology">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Dashboard"/>
      <owl:onProperty rdf:resource="#provides"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
```

```
    <owl:onProperty rdf:resource="#develops"/>
    <owl:allValuesFrom rdf:resource="#Analytics"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#provides"/>
    <owl:allValuesFrom rdf:resource="#StatutaryFile"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#develops"/>
    <owl:allValuesFrom rdf:resource="#StatutaryFile"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Organization"/>
 </owl:Class>
 <Analytics rdf:ID="LineGraph"/>
 <owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
   <Dashboard rdf:ID="KPI"/>
   <Dashboard rdf:ID="Graphs"/>
   <Dashboard rdf:ID="Charts"/>
  </owl:distinctMembers>
 </owl:AllDifferent>
 <owl:AllDifferent>
```

```
    <owl:distinctMembers rdf:parseType="Collection">
      <Reports rdf:about="#BalanceSheet"/>
      <Reports rdf:ID="ProfitAndLoss"/>
    </owl:distinctMembers>
  </owl:AllDifferent>
  <Analytics rdf:ID="BarChart"/>
  <Analytics rdf:ID="Table"/>
  <StatutaryFile rdf:ID="BalanceSheetReport"/>
  <Analytics rdf:ID="PieChart"/>
</rdf:RDF>
```