An Ontology-Driven Framework towards Building Enterprise Semantic Information Layer

Fuqi Song, Gregory Zacharewicz, and David Chen

Univ. Bordeaux, IMS, UMR 5218
F-33400 Talence, France
(fuqi.song,gregory.zacharewicz,david.chen)@ims-bordeaux.fr

Abstract. Enterprises Information Systems (EIS) have been applied for decades in Computer-Aided Engineering (CAE) and Computer-Aided Design (CAD), where huge amount of increasing data is stored in the heterogeneous and distributed systems. As systems evaluating, system redesign and reengineering are demanded. A facing challenge is how to interoperate among different systems by overcoming the gap of conceptual heterogeneity. In this article, an enlarged data representation called Semantic Information Layer (SIL) is described for facilitating heterogeneous systems interoperable. SIL plays a role as mediation media and knowledge representation among various systems. The SIL building process is based on ontology engineering, including ontology extraction from relational database (RDB), ontology enrichment and ontology alignment. Mapping path will maintain the links between SIL and data source, and query implementation and user interface are applied to retrieve data and interact with end users. We described fully a practical ontology-driven framework for building SIL and introduced extensively relevant standards and techniques for implementing the framework. In the core part of ontology development, a dynamic multi-strategies ontology alignment with automatic matcher selection and dynamic similarity aggregation is proposed. A demonstration case study in the scenario of mobile phone industry is used to illustrate the proposed framework.

Keywords: semantic information layer; ontology engineering; ontology-driven; system interoperability
1. Introduction

Enterprise Information Systems (EIS) have been applied widely in every professional domain for years, and the number of new information systems is growing rapidly, such as the systems in the domain of Computer-Aided Engineering (CAE) and Computer-Aided Design (CAD). Huge amount of data is stored in various information systems via diverse formats, including databases, text files, multimedia files, etc. A challenge facing is enabling interoperability between these systems. Most of data is stored isolatedly in specific information systems and is often difficult to share the information. Even in the same systems, the right information could not be found when it is needed. One essential reason is that structural and semantic heterogeneity are existing among heterogeneous systems, this gap obstructs the systems interoperable.

Data integration was proposed since decades to solve data heterogeneity issues, it has drawn many researchers’ attention and obtained promising results [1-3]. It aims to integrate heterogeneous data sources without concerning the differences among them. Latterly, ontology is widely applied to data integration, since semantic heterogeneity became a more significant issue over the structural aspects. Ontology is a formal representation of set of concepts within a specific domain and relationships between these concepts. It plays important roles in computer-aided design and engineering. The roles of ontology in engineering can be classified into three categories: 1) conceptualizing knowledge: using ontology as a formal representation to illustrate the models and concepts. A specific ontology will be proposed to illustrate certain domain knowledge; 2) supporting design annotation and enrichment; using ontology to enrich and elaborate concepts, in order to make concepts understood easily and explicitly; 3) serving as mediation media: ontology is used as a representation form to share concepts among heterogeneous systems. Unlike the traditional data integration approaches, ontology-based method seeks to achieve high extendibility and less coupling among the existing systems.

As the advert and the development of semantic web technologies rapidly, many ontology-based tools have been developed and commercialized. Also, several working groups of W3C are dedicating to standardize various techniques needed in semantic web, such as RDF, OWL, RDB2RDF, SPQRAL, etc. This supplies solid technical and theoretical ground for applying semantic web to engineering systems. Although semantic web was developed from the domain of internet and also being applied mostly in the domain of internet, the semantic web technologies can be adverted to engineering systems. They will aid engineering systems to solve data interoperability and heterogeneous data query issues as one promising alternative.

In order to extend the narrow scope of pure ontology and suit to engineering systems, an enlarged information layer with semantics representation is described as semantic information layer (SIL). SIL not only represents the semantic information, but also provides mechanism to maintain the links with data sources, and the accessibility to upper level applications. To build SIL, the first issue is developing semantics from relational database (RDB), which is the main storage in many engineering information systems. Ontology extraction and enrichment will map the schema and populate records to ontology. When facing with several extractions, ontology alignment can link and map concepts between different extracted ontology. The second issue is how to make the SIL accessed by upper applications. Mapping representation will maintain connection between SIL and RDB, and with query language the information will be retrieved either from SIL or from RDB via mappings.

The objective of the paper is to describe a practical ontology-driven framework, and illustrate the components and steps for building Semantic Information Layer (SIL), eventually to achieve interoperability among different engineering systems. Some suggested and mature techniques are introduced. The data source is considered to be relational database (RDB), the other kinds of data sources are not discussed in this paper. The framework serves as reference architecture and process to develop SIL. In real implementation, the specific techniques used can vary depending on particular demands.
Besides the introduction section, there are 5 sections in this article. The reminder of this article is organized as follows. Section 2 introduces ontology representation and development for engineering informatics, also investigates some applications with ontology in engineering. Section 3 describes the framework for constructing semantic information layer (SIL), including ontology development: ontology extraction, enrichment and alignment, also mapping representation and query implementation. Section 4 describes the dynamic multiple strategies-based ontology alignment (dmsOA) approach, addressed the architecture and each component, the alignment strategy and matching process in details. Section 5 uses a demonstration case to illustrate the framework, also discusses the contributions and challenges in the application of the proposed framework. Section 6 draws some conclusions.

2. Ontology for Engineering Informatics

An ontology is defined as a formal, explicit specification of a shared conceptualization [4]. In this definition, “formal” indicates that ontology is machine-readable, namely, the format of ontology can be understood and processed by computers. “explicit” refers to that the concepts and relations in an ontology are defined explicitly. From this perspective, an ontology is described as a 6-uple:

\[
(C, P, H^C, H^P, A, I),
\]

including a set of concepts \( C \) and a set of properties \( P \), the hierarchy relationship between concepts and its sub-concepts is denoted by \( H^C \), in the same way, \( H^P \) denotes the hierarchy relations between properties and its sub-properties. \( A \) is a set of axioms and \( I \) is the set of instances of concepts and properties.

2.1 Ontology Representation

Traditional ontology representation is based on predicate and description logic, F-logic [5] and Ontolingua [6] are frame-based; RuleML, LOOM, and CycL are logical-based. As well as some modeling languages are used for presenting ontology, e.g. IDEF5 and UML. Lately, the languages based on HTML and XML were developed since internet grown rapidly, such as, SHOE (Simple HTML Ontology Extensions) [7] and XOL (XML-based ontology-exchange language) [8]. As the advent of RDF(Resource Description Framework), which was designed originally as a metadata data model by W3C, a few languages are developed based on RDF, including OIL, DAML, and OWL [9].

OWL is derived from the DAML and built upon the RDF. It is now a standard semantic web ontology language. It has strong support for expression of web ontology and powerful engine to represent semantics. Besides OWL Full, there are two specific subsets: OWL DL (Description Logie) and OWL Lite. OWL DL was developed for supporting existing description logic and to supply a subset of language that includes computational properties for reasoning systems. OWL Lite was designed for simple implementation and to provide users with a functional subset at the beginning of using OWL.

Fig. 1 illustrates the ontology languages stack. The traditional methods refer these based on predicate and description logic, they are applied originally for reasoning and artificial intelligence. Recent methods are developed based on XML and RDF(s), that enhance the features for internet and semantic web. Three types of classifications are presented, frame-based, logical language and web-based. In each type, some typical languages are listed. Among web-based approaches, there is a precedence relation, generally XML is the base for all and these languages are developed based on RDF/RDF(s). Specially, SWRL [10] combines OWL and RuleML to benefit both features. RIF and KIF are interchange languages among these ontology languages, for converting format from one to another.
2.2 Ontology Development

Ontology engineering [11] defines how to build ontology from a collection of knowledge sources. It consists of six steps: gathering, extraction, organization, merging, refinement, and retrieval. The first five steps concern building the ontology and the last step is for using ontology. The process is iterative and can be executed as many times as needed.

The first step is to identify the knowledge source, where the ontology is built upon. The forms of knowledge sources are diverse: structured data, such as database; semi-structured data, such as XML and HTML from internet; free documents and multimedia. In engineering system, most of data is stored in relational database (RDB). The schema of RDB is a kind of semantic representation. So, usually it is taken as the starting point to extract ontology. Depending on different knowledge sources, the extraction rules vary. As for extraction from RDB, rule-based and data mining-based approaches are commonly used. The next steps refine the extracted ontology by formalizing the representation and adding semantics with reference ontology and external knowledge resources. These steps need both automatic and manual actions involved.

Besides building new ontology from nothing, ontology reuse is an important way to constructing ontology. There are different modes for various purposes: ontology integration, ontology merging, and ontology alignment, which are easily got confused. [12] gave a comprehensive comparison of them. Ontology alignment seeks to find correspondences and sets up connection between ontologies, whereas ontology integration and merging create a new integrated ontology with the information of the source ontologies. An intuitive illustration is shown as Fig. 2 a) and b) respectively. In table 1, the input and output of different ontology reuse approaches is listed, which reflected the application conditions and goals.

<table>
<thead>
<tr>
<th>Type</th>
<th>Input 2..* ontologies from</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ontology merging</td>
<td>Similar domains</td>
<td>One single coherent ontology</td>
</tr>
<tr>
<td>Ontology alignment</td>
<td>Complementary domains</td>
<td>Two or more ontologies which are linked</td>
</tr>
<tr>
<td>Ontology integration</td>
<td>Different domains</td>
<td>One single ontology</td>
</tr>
</tbody>
</table>

Fig. 1. Ontology languages RIF/KIF/OIL

Fig. 2. a) Ontology alignment and b) Ontology integration and merging
2.3 Applications in Engineering

In engineering applications, ontology has been applied widely. Respecting to the three classifications introduced in the introduction section: 1) conceptualizing knowledge, 2) supporting design annotation and enrichment, and 3) serving as mediation media, some applications in the domain of design and engineering are investigated.

Knowledge representation for conceptualizing a certain domain, for example, ONTO-PDM [13] represented a product model in manufacturing environment, and [14] presented a domain ontology for electrical power quality (PQ) called PQONT. Ontology ONTO-PDM aims to facilitate the interoperability among software applications during the physical product lifecycle. PQONT tried to build a shared vocabulary in the domain of PQ for different systems. In [4], ontology of interoperability (OoI) is described to formalize the concepts and their relations in enterprise interoperability from a system theory point of view.

Supporting design annotation and concepts enrichment: ontology is adopted to annotate and enrich certain concepts or knowledge. The enriched object could be a model or a document. The enrichment can be performed in two ways: 1) Basing on the original object, enrichment is made to create a new enriched larger object, which is different to the original one. The added information becomes part of the object and is essential. [15] defines a method to add semantic annotations to concepts; 2) Enrichment, as concepts annotation, provides additional information and semantic to the object, which is complementary. [16] used ontology to develop semantics for enterprise models, in order to understand among different modeling languages. In [17], the authors proposed a methodology to build semantically annotated multi-faceted ontology for product family modeling, and [18] use semantic methods to support engineering design innovation.

Serving as mediation media: ontology is used as a representation form to share concepts among heterogeneous systems. Unlike the traditional data integration approaches, ontology-based method seeks to achieve high extendibility and less coupling among the existing systems. [19] proposed a knowledge-centered approach with ontology to integrate different information portals of European Commission (EC) and British institutions. The work aims to provide an integrated and semantic-based user interface for accessing documents, which are published by different partners. In their work, ontology is taken as a library of shared concepts with global ontology mode. This ontology is used to eliminate semantic ambiguity among different contexts. [20] aims to implement a integrated “intelligent” environment for CAD by using ontology. Since ontology is a kind of semantic representation, the model and data can be denoted like human beings, in order to understand by computers and eventually achieve the goal of “intelligence”. MAFRA [21] is an ontology-based methodology for integrating different systems. The authors proposed a comprehensive framework to illustrate how integrate distributed ontology. The method can be adopted for integrating various knowledge and information sources.

3. Framework for building Semantic Information Layer (SIL)

Semantic information layer refers to an information layer with semantics and has interface to interact with upper application level and has access to lower data source layer. SIL differs from a single pure ontology. Besides taking ontology as main representation of information, SIL extends the ability to be accessed from user level and maintains the connections to data sources.

3.1 Overview

This framework is designed in the context of Framework for Enterprise Interoperability (FEI) [22] and a concrete architecture for EIS interoperability using semantic web technologies [23], focusing on solving data interoperability for EIS. The framework is illustrated in Fig. 3. In the framework, the main components are data sources (RDB), semantic information layer (SIL), query and answer engine (Q&A), user interface, and upper application. However, except SIL and Q&A, the other parts are not going to be discussed. Section 3.2 and 3.3 will discuss the
ontologies with ontology extraction, enrichment and alignment. Section 3.4 will discuss how to build the mapping between semantic layer and data sources, and how to query data from SIL to make it available for upper level applications.

3.2 Ontology Extraction and Enrichment

Ontology extraction is a pre-process for developing the semantics. Because the information is stored in the knowledge sources in various formats, such as RDB, files and HTML, the first step is to extract the main semantic and structure. The extracted ontology is regarded as “raw ontology”, which needs to be further enriched and elaborated by reference ontology or manual adjustment. The two processes are illustrated in Fig. 4. Raw ontology is extracted with defined rules, and then enriched with upper or reference ontology to obtain final ontology.

3.2.1 Schema Extraction

Extraction rules are classified as explicit and implicit. Explicit rules are the rules defined from structural aspects. The mappings between RDB and ontology are set up directly. For instance, a table in RDB is mapped to class concept in ontology, and field in RDB to property in ontology. In addition, [24, 25] defined implicit rules. The implicit rules mine the hidden information from RDB, which is not explicitly available, such as [26] uses hierarchy mining to discover relation “subClassOf” and [27, 28] extract “sameAs” property using similarity. [29-31] designed and implemented extraction systems with rules. We summarized and adapted some commonly used rules in table 2, which could represent almost the semantic relation in database schema.

Table 2. Ontology extraction rules (adapted from[24-31] )

<table>
<thead>
<tr>
<th>Rule type</th>
<th>From RDB</th>
<th>To ontology(OWL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit</td>
<td>Table</td>
<td>class</td>
</tr>
<tr>
<td>explicit</td>
<td>Column</td>
<td>property</td>
</tr>
<tr>
<td>explicit</td>
<td>Row</td>
<td>individual/instance</td>
</tr>
<tr>
<td>explicit</td>
<td>Primary key</td>
<td>owl:inverseFunctionalProperty</td>
</tr>
<tr>
<td></td>
<td></td>
<td>owl:minCardinality rdf:datatype=”&amp;xsd:int” 1/</td>
</tr>
<tr>
<td>explicit</td>
<td>Foreign key</td>
<td>owl:objectProperty</td>
</tr>
<tr>
<td>explicit</td>
<td>Check</td>
<td>owl:hasValue</td>
</tr>
<tr>
<td>Explicit</td>
<td>Unique</td>
<td>owl: inversFunctionalProperty</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Explicit</td>
<td>Not null</td>
<td>owl: minCardinality rdf: datatype=&quot;&amp;xsd:int&quot; 1/</td>
</tr>
<tr>
<td>Implicit</td>
<td>Similarity check</td>
<td>owl:sameAs</td>
</tr>
<tr>
<td>Implicit</td>
<td>Hierarchy mining</td>
<td>rdfs: subClassOf</td>
</tr>
</tbody>
</table>

Besides these rules, a few mapping languages are proposed to enable the mappings between RDB and different ontology representation. [32] gave a comprehensive comparison among different main mapping languages. W3C is working on standardizing the mapping language from RDB to RDF, a named R2RML ongoing work is being carried out. Hopefully this mapping language would contribute to the ontology extraction from RDB in not far future.

### 3.2.2 Instance Population

Ontology population is the process of transforming data from database into ontology instances. There are two general approaches to do this, normally regarded as static and dynamic or massive dump and query driven [33] as illustrated in Fig. 5. In previous case, static ETL (Extraction, Transformation and Load, massive dump) uses a batch process to transform all the database records into ontology instances (Fig. 5.a), whereas dynamic approach (query driven) only transforms part of database data in response to certain queries when requests are made (Fig. 5.b)). [34, 35] are such application cases for static and dynamic population respectively.

Current work is towards to enable automatic extraction, since manual ontology building is time-consuming and error-prone. Even though many promising extraction rules and methods have been proposed and defined, the full-automatic extraction tools are unsatisfactory when applying to real engineering databases. Normally a combined way is used, so-called semi-automatic approach combining automatic process and manual involvement. More details could be found from previous work, [36] evaluated some tools with a comprehensive evaluation framework. [37] surveyed current approaches and tools implemented for mapping of RDB to RDF.

![Fig. 5. a) Static mode and b) Dynamic mode in instance population](image)

### 3.2.3 Ontology Enrichment

In engineering and manufacturing domain, some reference ontologies have been developed for representing domain knowledge. These ontologies can be used to enrich and elaborate the extracted ontology. Suggested Upper Merged Ontology (SUMO) [38] and Object-Centered High-level Reference Ontology (OCHRE) [39] are two ontologies for general concepts in any domain. Infrastructure Product Ontology (IPD) [40], ONTO-PDM [41] emphasizes in product and services. In IPD, products span all 5 sectors of utilities: water, wastewater, gas, electricity, and telecom. In manufacturing domain, MASON [42] and [43] are two propositions.

A few methodologies have been proposed to enable automatic enrichment. [44] proposed an approach to enrich ontology and annotate documents automatically based on semantic annotation of on-line glossaries. [45] used upper ontology SUMO [38] as a base to extract knowledge and analyze semantics from text. [46] presented a framework to evaluate integrations between ontological and linguistic resources, it defines sets of shared vocabularies for addressing the knowledge about heterogeneous linguistic resources.
3.3 Ontology Alignment

Ontology alignment is a process for ontology reuse and a prerequisite for further data integration in many areas. Ontology alignment seeks to find semantic correspondence between a pair of ontology elements by identifying similarity. Strictly speaking, ontology alignment includes two phases: finding the correspondences and setting up connections as Fig. 6 shows. The “Match” process seeks the possible correspondences and “Align” process sets up the links with the constraints, e.g. threshold, weight.

![Fig. 6. Ontology alignment process](image)

3.3.1 Correspondence

Adapted from the representation format of correspondence in [47], there are two types of correspondences for multiple matchers-based approach: intermediate and final. Intermediate correspondence is discovered by a specific matcher, and then several intermediate correspondences are combined into a final correspondence using pre-defined combination strategy. An intermediate correspondence $ic$ is defined as

$$ic = \{e_1, e_2, r, v, M, id\} \quad (2)$$

where $e_1$ and $e_2$ are elements from ontology $O_1$ and $O_2$ to be matched. Respectively, $v$ denotes the confidence between $e_1$ and $e_2$ identified by matcher $M$ with a relation $r$, which could be “equal”, “subClassOf”, “superClassOf”, etc. $id$ is a unique identifier for this correspondence. A final correspondence $fc$ is similar to intermediate correspondence without the information concerning to a specific matcher $M$, defined as

$$fc = \{e_1, e_2, fr, fv, fid\} \quad (3)$$

where $fr$ is a relation derived from relations in intermediate correspondences and $fv$ denotes a confidence combined from intermediate correspondences’ confidences. For both types of correspondences, the relation $r$ and $fr$ refer to equal in our approach.

3.3.2 Matching Techniques

In order to find equivalent correspondences between two elements in ontologies, different aspects of ontology need to be measured, three levels of ontology is summarized in Fig.7. Most of the matching techniques are proposed from this point of view.

1. At element level, the class itself is treated as the studied object; the label, comment and internal information of it are investigated. The most used techniques are string metric [48], string similarity, domain, property and data type comparison [27], etc.
2. At local level, the objects and the relations linked to the studied element are taken into account, such as similarity flooding [49], graph-based, taxonomic-based, etc.
3. At global level, the whole ontology is taken as a context and environment, the relation and affect between it and the studied object is investigated, machine learning, artificial neural network [50] are some methods applied in this level.

In the book [51], the author addressed a comprehensive introduction of different approaches. [52] listed the main approaches with short analysis of advantages of disadvantages. We summarized the main techniques by emphasizing on analyzing the strength and limitations. The summary is listed in table 3, where columns are general classification, techniques in the category and main limitations. Some categories and techniques may intersect with the others, so we just put them regarding the major features. According to this, we defined three matchers (see section 4.1) with multiple strategies.
Table 3. Ontology alignment techniques

<table>
<thead>
<tr>
<th>Category</th>
<th>Main techniques</th>
<th>Strength and constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>String-based</td>
<td>Normalization, substring test, edit distance, token-based distance, Soundex index, n-gram</td>
<td>+simple string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- semantic aspects</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- complex strings or phrases</td>
</tr>
<tr>
<td>Language-based</td>
<td>Tokenization, stopword elimination, term extraction</td>
<td>+complex strings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- semantic level</td>
</tr>
<tr>
<td>Constraint-based</td>
<td>Explore internal information of the concepts, e.g. property comparison, data type comparison, domain comparison</td>
<td>+internal information of concepts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- semantics</td>
</tr>
<tr>
<td>Structure-based</td>
<td>Graph matching (ex. Similarity flooding), Taxonomy-based method</td>
<td>+structure</td>
</tr>
<tr>
<td>Semantic-based</td>
<td>SAT solver, Description Logic(DL)</td>
<td>+semantics</td>
</tr>
<tr>
<td>External knowledge</td>
<td>Using a thesauri and dictionary, e.g. WordNet, SimMetric</td>
<td>+semantic aspects</td>
</tr>
<tr>
<td>- based</td>
<td></td>
<td>- extend the local limitations</td>
</tr>
<tr>
<td></td>
<td>Upper/reference/top-level ontology</td>
<td>- require domain knowledge or</td>
</tr>
<tr>
<td></td>
<td>Alignment reuse</td>
<td>specific resources</td>
</tr>
</tbody>
</table>

3.4 Mapping Path and Querying Implementation

To make the semantic layer available to upper level, a link is maintained between semantic layer and data sources. XPath rules in a XSLT stylesheet could be used to represent the mappings between RDB and RDF/OWL [37], such as D2R MAP [53] and RoO [54] in a XML-based declarative language, Virtuoso’s meta-schema [55] language. In D2R MAP [53], three elements ClassMap, DataTypePropertyBridge and ObjectPropertyBridge are used to describe the mappings. An example of mapping product information is illustrated as following, in which three fields: product code, product name, and product family name from two tables product_table and product_family_table are mapped to datatype property code, name and object properties family_of in ontology.

```xml
<Map>
  <DBConnection odbcDSN="ProductDB" />
  <ProcessorMessage outputFormat="RDF/XML-ABBREV"/>
  <Namespace prefix="ex" namespace="http://example.org#"/>
  <ClassMap type="ex:Product" sql="SELECT product_table.code, product_table.name, product_family_table.name product_table.FROM product_table, product_family_table WHERE product_table.family_id = product_family_table.id;" groupBy="product_table.code">  
    <DatatypePropertyBridge property="ex:productCode" column="code"/>
    <DatatypePropertyBridge property="ex:productName" column="name"/>
    <ObjectPropertyBridge property="ex:family_of" referredClass="ex:Product" referredGroupBy="code"/>
  </ClassMap>
</Map>
```
R\textsubscript{2}O allows the description of arbitrarily complex mapping expressions between ontology elements (concepts, attributes and relations) and relational elements (relations and attributes) [54]. R\textsubscript{2}O provides conditions & operations and the rule-style mapping definition for attributes, which allowed extendable ability to define more complex mappings.

Once the semantics and mappings have been built, there are two means to implement queries corresponding to static and dynamic data population (see section 3.2.2): 1) with semantic querying language, e.x. SPARQL [56], to execute queries against semantic layer. It does not access the data source since all the data are dumped to the semantic layer. 2) transforming query language (e.x. SPARQL) to SQL and execute SQL against the RDB directly. [57] discussed transformation from SPARQL to algebra and further into SQL.3) querying from the mappings, such as ODEMapster [54].

For the first case, we use ontology “eClassOWL” (http://ebusiness-unibw.org/cgi-bin/mailman/listinfo/eclassowl) as querying example. “eClassOWL” is an OWL ontology for describing the types and properties of products and services. A query sample (http://www.heppnetz.de/projects/eclassowl/) for “Find all product models of pencils that have a pointed tip.” is as following.

```
PREFIX eco: <http://www.ebusiness-unibw.org/ontologies/eclass/5.1.4/#>
PREFIX gr: <http://purl.org/goodrelations/v1#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?model WHERE {
  ?model a eco:C_AKF303003-gen.
  ?model a gr:ProductOrServiceModel.
  ?model eco:P_BAG073001 eco:V_BAC386001.
}
```

4. dmsOA: dynamic multiple strategies-based Ontology Alignment

Most of current ontology alignment approaches adopt multiple matching techniques to adapt various source ontology[58]. RiMOM [59] used two similarity factors: label similarity factor ($F_{LS}$) and structure similarity factor ($F_{SS}$), computed from concepts and structures features of source ontologies, to help defining the weight of different matchers. UFOme [60] proposed two affinity confident: lexical affinity coefficient $L_a$ and structural affinity coefficient $S_a$, which also concluded from information of source ontologies. The weight is calculated based on the two affinity coefficients using a heuristic function. PRIOR+ [61] proposed a term “harmony” $h$ to estimate the importance and reliability of different similarities and use it as the weight. In our approach, the average similarity value obtained one matcher is used as factor to calculate the weight. This method tries to overcome the local constraints and balance between different matching algorithms.

A dynamic alignment system with multiple strategies called “dmsOA” is designed, with 7 components illustrated as Fig. 8. Preprocessor imports source ontology (in .owl format) and divides it into elements by recognizing the classes and properties. Tokenization and elimination will be applied in this step, and then these elements are taken as input for matchers. Strategy adapter will decide a strategy according the features of the elements from source ontology. With the selected strategy, specific matchers will be assigned to perform matching tasks and generate intermediate correspondence. On the other hand, with the factors obtained by factor calculator, weight calculator computes and gets the weights of each matcher. Aggregator combines the intermediate correspondences into final correspondence. Aligner sets up the equality between ontologies with final correspondence.
4.1 Matching with multiple strategies

Before carrying out the matching, a general pre-process will be taken to prepare the source ontologies. Unnecessary information, such as stop words, preposition, is eliminated. Compound label will be tokenized into single words.

For a specific matcher, it may work more efficiently and obtain more accurate results on some ontology than the other ones. This indicates that the matcher has a matching condition to achieve better results. So does the ontology, for the specific ontologies, they need a suitable matcher to match them. The key issue is how to compromise different matchers to make them complementary. With the comparison of different methods for ontology alignment, we designed three matchers which can complement each other. 1) String-Based Matcher (SBM) uses Jaro-Winkler[62] distance and N-GRAM with a condition constraint. SBM seeks to find the matches by taking the elements in ontologies as sequence of characters. 2) Taxonomy-based matcher with WordNet (TMW) adapted from [63]’s model, it uses a lexical database WordNet [64] for identifying the semantic similarity. 3) Matcher with Alignment Reuse (MAR) reuses the alignments identified to aid the current matching.

Fig.9 shows how to select the matchers. A pair of elements from pre-processed ontology is as input, and then several conditions are examined to decide which strategy to apply. First if a pair of elements has identical label, then they are matched with SAME_WORD: confidence = 1 without using the other matchers. Otherwise, three conditions are checked to decide which matcher to use. Afterwards, aggregator will combine the matching results.
String-Based Matcher (SBM):

**Jaro-Winkler Distance.** Levenshtein distance (also known as edit distance) is the mostly known distance function, in which distance is the cost of operations, including insertion, deletion and substitution, for converting $s_1$ to $s_2$ in a best sequence. Jaro-Winkler [62] distance is a broadly string metric-based on Jaro distance [65].

**N-Gram.** Jaro-Winkler has limitations when matching between two strings which have big differences in length, such as “member” and “conference_member”. When $\min(|s_1|,|s_2|) > \max(|s_1|,|s_2|)/2$, SBM applies n-gram [66] as matching technique, more specifically, trigram. $trigram(n,s)$ denotes the set of trigrams, where $n$ is the length of substring $s$. The similarity between strings $s_1$ and $s_2$ with n-gram is

$$TG(s_1, s_2) = \frac{|trigram(s_1) \cap trigram(s_2)|}{\min(|s_1|, |s_2|) - 2}$$  \hspace{1cm} (5)

**Taxonomy-based Matcher with WordNet (TMW):** Besides using the string metric to identify the similarity, the semantic is another significant aspect. A taxonomy-based approach [63], called Lin model, for semantic similarity matching is applied in TMW. In Lin model, the taxonomy is taken as a tree, WordNet [64] is used as the taxonomy. It returns a semantic similarity by measuring commonality between two words in the taxonomy tree. Assuming that the taxonomy is a tree, if $s_1 \subseteq C_1$ and $s_2 \subseteq C_2$, the commonality between $s_1$ and $s_2$ is $s_1 \subseteq C_0 \cap s_2 \subseteq C_0$ where $C_0$ is the most specific class that subsumes both $C_1$ and $C_2$. $P(C)$ is the probability that a randomly selected object belongs to $C$. The similarity of matcher TMW is defined as

$$TMW(s_1, s_2) = \frac{2 \cdot \log P(C_x)}{\log P(C_1) - \log P(C_2)}$$  \hspace{1cm} (6)

**Matcher with Alignment Reuse (MAR):** Matching ontology with alignment reuse is a new research topic, a few work addressed this method, such as COMA [67] and OLA [68]. The motivations to reuse alignment are twofold: 1) the ontologies in a same area are similar and many ontologies have been aligned. These resources could be adopted; 2) with the existing alignments information, to mine the hidden information which could not be found directly with the information in $O_1$ and $O_2$. Ontologies $O_1$ and $O_2$ are the target ontologies to be matched, $O_t$ is a third ontology, which has existing alignments with $O_1$ and $O_2$, then the alignments between $O_1$ and $O_2$ could be derived via $O_t$.

A definition for alignment reuse is, $\forall c_i \in O_1, c_j \in O_2$ and $c_i \in O_{o_x}$, if $\exists$ final correspondence $(c_o, c_o, r_o, v_o, id)$ between elements $c_i$ and $c_o$, and final correspondence $(c_j, c_o, r_p, v_p, id)$ between elements $c_j$ and $c_o$, then the correspondence between $c_i$ and $c_j$ is defined as

$$\{c_o, c_o, R(r_o, r_p), V(v_o, v_p), id\}$$  \hspace{1cm} (7)

where $R, V$ are the functions to compute respectively the relation and similarity values between $c_i$ and $c_j$. For $V$ function, we use several possible solutions as defined in (8). The selection of $V$ function will depend on the running results for different ontologies. The relation function $R$ is as defined in (9), for our current approach, it is the first case: $r = r_o = "$equal"$.

$$V = \begin{cases} \text{avg}(v_i, v_j), & \text{or} \\ \text{max}(v_i, v_j), & \text{or} \\ \text{min}(v_i, v_j), & \text{or} \\ v_i * v_j \end{cases}$$  \hspace{1cm} (8)

$$R = \begin{cases} \text{equal}, r_i = r_j = "$equal" \\ \text{superclassof}, r_i = r_j = "$superclassof" \\ \text{subclassof}, r_i = r_j = "$subclassof" \\ \ldots \end{cases}$$  \hspace{1cm} (9)

The alignments reused should be from related domain and context to avoid inaccuracy. An alignments database could be set up in order to be reused by the others. In a special case, when external resources are not available, we could use reference ontology as the intermediate, because it contains a comprehensive and standard concepts and relations.
4.2 Dynamic Similarity Aggregation

To obtain an overall better matching results and overcome the local limitations, we use average matching results of one matcher to decide the weight. The factor of one matcher is denoted as

\[
fx = \frac{\sum_{i,j=1}^{m,n} \text{Matcher}_x(e_i, e_j)}{m \times n}
\]  

(10)

where \( f_x \) denotes the factors of the three matcher SBM, TMW or MAR respectively. \( e_1 \) and \( e_2 \) are elements, and \( m, n \) is the number of elements to be matched in \( O_1 \) and \( O_2 \).

The weight is computed based on these factors. First we define two vectors: 1) \( b = (b_{sbm}, b_{tmw}, b_{mar}) \) consisting of boolean values, where \( b_i = 1 \) means that the matcher is used and has discovered an intermediate correspondence, otherwise \( b_i = 0 \); 2) \( f = (f_{sbm}, f_{tmw}, f_{mar}) \) is the vector of factors obtained. The weight of each matcher is computed as

\[
W_x = \frac{b_x \cdot f_x}{b \cdot f}
\]  

(11)

where \( W_x \) is the weight of matcher SBM, TMW or MAR. The weights have following features,

- if \( \sum b = 1 \), \( W_{sbm} = W_{tmw} = W_{mar} = 1 \),
- if \( \sum b = 2 \), \( W_{sbm} + W_{tmw} = 1 \) or \( W_{sbm} + W_{mar} = 1 \) or \( W_{tmw} + W_{mar} = 1 \),
- if \( \sum b = 3 \), \( W_{sbm} + W_{tmw} + W_{mar} = 1 \).

Final similarity \( sim \) between elements \( e_1 \) and \( e_2 \) is

\[
sim(e_1, e_2) = \sum_{i=1}^{3} V_i W_i
\]  

(12)

A threshold \( th \) indicates which level of similarity can be taken as equality. Once the similarity is greater than \( th \), the equality is set up, otherwise the link is dismissed. The normally used methods are:

- **Constant method**: The user defines a constant \( c \) as threshold. This value is from the experienced domain expert with or from experiments.
- **N-Percent method**: A certain percentage based on the maximal similarity is taken, \( th = \max(\text{sim}(e_1, e_2)) \times (1 - \text{np}) \), where \( np \) is a fixed percentage defined by user.

4.3 Implementation

A software prototype was developed in java to validate the proposed ontology alignment approach. To illustrate the implementation, we use core class diagram to explain in Fig. 10. The attributes and methods in classes are hidden to simplify the illustration. The class diagram respects to the architecture (see Figure 8). Ontologies are processed to a set of elements \( \text{Element} \) and are encapsulated into class \( \text{OWL Ontology} \). One element notes the basic information of label name, filtered label name, type of node and hierarchy. \( \text{Strategy Selection} \) will choose a specific \( \text{Matching Strategy} \) according to the features of source ontology. Then chosen \( \text{Matcher} \) will perform the matching tasks to generate \( \text{Inter Correspondence} \). On the other hand, \( \text{Factor Calculator} \) calculates the \( \text{Factors} \) and with them to compute the \( \text{Weight} \). \( \text{Aggregator} \) will use weight and \( \text{Aggregation Strategy} \) to combine the intermediate correspondences into \( \text{Final Correspondence} \), which is the final alignment result expected. \( \text{Evaluator} \) will be used to evaluate the alignment results with reference alignments. The generic alignment process algorithm is presented in pseudo code as following.
Generic algorithms in pseudo code:

Input: ontology O1 and O2, reused alignments list ListOfReuseAlignment
preprocess(O1, O2) generates ListOfElements1 and ListOfElements2;
initialize ListOfFinalCorrespondence;
initialize ListOfInterCorrespondence, ListOfFinalCorrespondence;
foreach element e1 in ListOfElements1
    foreach element e2 in ListOfElements1
        select strategy for a pair of elements (e1, e2);
        //according to the strategy, one or more matchers could be selected
        locate matcher(s) by selected strategy;
        foreach matcher selected
            initialize InterCorrespondenceForEach
            generate confidence by specific matcher(s);
            wrap information to an intermediate correspondence;
            add intermediate correspondence to InterCorrespondenceForEach;
        end
        add InterCorrespondenceForEach to ListOfInterCorrespondence;
    end
end
calculate factors with ListOfElements1 and ListOfElements2;
compute weights using generated factors;
foreach InterCorrespondenceForEach in ListOfInterCorrespondence
    combine the confidences from intermediate correspondences into one final confidence;
    wrap information to an final correspondence;
    add final correspondence to ListOfFinalCorrespondence
end
Output: list of alignments: ListOfFinalCorrespondence

A sample fragment of the final correspondence obtained by the matching algorithms is as following. The format corresponds to the formula defined in section 3.3.1. Two elements identified between two ontologies product.owl and order.owl and the similarity is 0.8 between OrderId and OrderNumber.

…

<Cell>
In OWL, there are some built-in constructs to link equivalent elements. \textit{owl:equivalentClass}, \textit{owl:equivalentProperty} and \textit{owl:sameAs} are such kinds of constructs. \textit{owl:equivalentClass} likes a class description to another class description, and this axiom requires that the two class description contains the same class extension. \textit{owl:equivalentProperty} is used to state that two properties have the same extension. \textit{owl:sameAs} links an individual to an individual. It means that the two URI references refer to the same elements, they have the same identity. The three constructs will be used to setup the links between different elements with discovered final correspondences.

5. Demonstration

5.1 Background introduction

To simplify the demonstration of the framework, we set up a scenario of virtual enterprises. This enterprise is a mobile phone manufacturing company. Among many of their information systems, there are two information systems: order management system and human resource management system. Now they are expanding and plan to open up marketing in film industry, in order to take up the filming market on mobile devices. As a start-up, they took over a film management and rental company.

In order to observe if the information systems can interoperate without re-developing existing ones, because they want to keep the current operations unchanged. The long-term strategy is to merge the business in the current film management company into their conventional business, such as taking the film rental and online watching as conventional orders. Also, they plan use human management systems to manage the staff in that company. So first they plan to build a mediator system to query information upon these information systems. They hope the query can be executed through a single interface and shared by all partners. The illustration of the scenario is shown as Fig. 11. Three information systems are located in two organizations and a mediation query interface is required.

The databases used are from the other projects with real data. \textit{Order DB} database is a minimal scale of a production order management database, including knowledge of order, product, customer and employee. \textit{HR DB} is database focusing the knowledge of human resources management. \textit{Film DB} is a database from film industry for managing film information and sales of films. The information of these databases is listed in table 4, including number of tables, number of fields and number of records.
Table 4. Information of source databases

<table>
<thead>
<tr>
<th>RDB name</th>
<th>Tables</th>
<th>Fields</th>
<th>Records</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order DB</td>
<td>8</td>
<td>59</td>
<td>3864</td>
<td>Production order management</td>
</tr>
<tr>
<td>HR DB</td>
<td>9</td>
<td>44</td>
<td>78635</td>
<td>Human resources management</td>
</tr>
<tr>
<td>Film DB</td>
<td>16</td>
<td>98</td>
<td>47273</td>
<td>Film rental management</td>
</tr>
</tbody>
</table>

5.2 Extraction and Enrichment

We use tool OntoToRDB [26], which was developed by Dassult Aviation in TAO European project (IST-2004-026460). OntoToRDB maps relation to class, field to datatype property, foreign key relationship to functional object property and composite key relation to object property. It also populates the records into instances with a RTAXON method, which can mine the hierarchical relation from data stored in database. We use it to extract each database and then obtained three ontology represented in OWL with instances as table 5 shows. The tool for editing and displaying ontology is Protégé v 4.1.0 [69]. The extracted object property and data property are as Fig. 12 shows. Fig. 13 display a snapshot of extracted class and instances contained. In this figure, left two columns are class and instances, and the right column is the information linked to each instance.

Table 5. Information of extracted ontologies

<table>
<thead>
<tr>
<th>RDF/OWL</th>
<th>Class</th>
<th>Object property</th>
<th>Datatype property</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>orders.owl</td>
<td>8</td>
<td>14</td>
<td>52</td>
<td>3864</td>
</tr>
<tr>
<td>hr.owl</td>
<td>10</td>
<td>12</td>
<td>18</td>
<td>78635</td>
</tr>
<tr>
<td>film.owl</td>
<td>16</td>
<td>42</td>
<td>67</td>
<td>42274</td>
</tr>
</tbody>
</table>

Fig. 12. Extracted object property and data property

Fig. 13. Snapshot of extracted instances in Protégé v4.1.0
And then the extracted ontology is further enriched with reference ontology SUMO and manual adjustment. The manual adjustment includes: 1) rename the object property. The extracted object property in OntoToRDB follows certain naming rules: to + class name, such as, toCustomers. That is because the relation information cannot be retrieved from the schema of database. Thus it is necessary to adjust the names of object property manually, for example, adjusting toCustomer to belongs_to; 2) enrich the meaning by manual adjustment, such as using full word department to replace abbreviation dept.

5.3 Ontology Matching and Query

The extracted ontology will be used as input source ontology to dmsOA matching system. The matching task is carried out only between concepts (class and property), instance matching is not included for simplifying the process. First, an input ontology is pre-processed and elements are identified. Following is a fragment of the processed elements, including the type of elements, label, filtered label name, and tokenized words.

```
Element [CLASS,employee, F:employee, C:(<employee, n>)]
Element [CLASS,order_detail, F:orderdetail, C:(<detail, n>,<MULTI_NOUN,order>)]
... 
Element [OBJECT_PROPERTY,paied_by, F:paiedby, C:(<pay, v>)]
Element [OBJECT_PROPERTY,ask, F:ask, C:(<ask, v>)]
Element [OBJECT_PROPERTY,belong_to, F:belongto, C:(<belong, v>)]
...
Element [DATATYPE_PROPERTY,amount, F:amount, C:(<amount, n>)]
...
```

The matching process generates final correspondences as following fragment shows: a pair of identified elements with a confidence and a unique identifier. In the fragment, the similarity between film.owl#OrderId and order.owl#OrderNumber, and film.owl#first_name and product.owl#firstName, is 0.8 and 1.0 respectively. We run the matching on each two of the three ontologies and get three sets of data.

```
<Map>
  <entity1 rdf:resource="http://ims-bordeaux.fr/film.owl#OrderId"/>
  <entity2 rdf:resource="http://ims-bordeaux.fr/order.owl#OrderNumber"/>
  <confidence rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.8</confidence>
  <relation>=</relation>
  <identifier>AL-FILM-ORDER-00001</identifier>
</Map>
...
```

With threshold ranged from 0.1 to 1.0, the number of filtered alignments is displayed in Fig. 14. In general, when the threshold increases the number of discovered alignments decreases. We can see that when threshold is less than 0.5, the number remains stable, around 102, 77 and 37 respectively. When the threshold is greater than 0.5, the number decreases smoothly.

```
<table>
<thead>
<tr>
<th>threshold</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>film_orders</td>
<td>103</td>
<td>103</td>
<td>102</td>
<td>102</td>
<td>93</td>
<td>71</td>
<td>53</td>
<td>40</td>
<td>27</td>
<td>24</td>
</tr>
<tr>
<td>orders_hr</td>
<td>78</td>
<td>78</td>
<td>78</td>
<td>77</td>
<td>67</td>
<td>38</td>
<td>27</td>
<td>19</td>
<td>17</td>
<td>14</td>
</tr>
<tr>
<td>hr_film</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td>37</td>
<td>37</td>
<td>32</td>
<td>27</td>
<td>16</td>
<td>16</td>
<td>15</td>
</tr>
</tbody>
</table>
```

Fig. 14. Number of discovered alignments filtered by threshold from 0.1 to 1.0
With the constant method introduced, we set manually a constant threshold \( th = 0.8 \). Filtered by \( th = 0.8 \), the available alignments number and the percentage are shown in table 6. With the OWL equality constructs introduced in section 4.3, the alignments of class and property will be connected between different ontologies.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>No. of Alignments</th>
<th>Total</th>
<th>Threshold</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>orders.owl</td>
<td>film.owl</td>
<td>40</td>
<td>103</td>
<td>0.8</td>
<td>38.83%</td>
</tr>
<tr>
<td>hr.owl</td>
<td>orders.owl</td>
<td>19</td>
<td>78</td>
<td>0.8</td>
<td>24.35%</td>
</tr>
<tr>
<td>film.owl</td>
<td>hr.owl</td>
<td>16</td>
<td>38</td>
<td>0.8</td>
<td>42.10%</td>
</tr>
</tbody>
</table>

An aligned ontology aligned.owl is established so far, we use SPARQL to retrieve the data by one example. Query: Find the name and age of employees, who are responsible for the film rental orders? The querying and retrieved data is like following. Via this way, the data could be accessed by upper level applications.

**5.4 Discussion**

The demonstration case study illustrates an application scenario in mobile phone and film industry. It shows the process how to apply the framework to build a mediation system, which aims to solve data interoperability issues among different partners. In this process, several software, techniques and algorithms may be applied accordingly. Some of them have been introduced in previous sections of this article.

In the proposed framework, ontology serves as both knowledge representation and as mediation to enable heterogeneous systems interoperable. From our point of views, the framework can solve effectively the following issues in engineering:

1. By establishing a global semantic layer upon heterogeneous data sources, it could solve the issues of querying data from various data sources. Taking conventional data integration approach as an example, if we need to query data from different RDBs, we need to connect to each RDB system and use joint query to retrieve information. However, with SIL and the query mechanism in the framework, it is unnecessary to deal with each data source directly. Also, current in-use systems will remain unchanged and unaffected, so that no extra resources and cost concerning maintaining legacy systems will be needed.

2. Semantic information layer overcomes the weakness of structured data storage, since the ontology could represent knowledge in a semantic and machine-readable way. It will improve the gap of semantic heterogeneity among different engineering systems. Besides, as the increasing development in semantic web, the semantic information layer could enable rapid transition and connection to the emerging technologies.

3. The framework could be implemented as mediation system to interoperate among different systems, since usually there are many information systems isolated in one organization. Besides, there are demanding requirements to enable collaboration between the systems from various partners. To achieve these goals, a mediation system could be built based on the framework accordingly.

In Computer-Aided Design (CAD) and Computer-Aided Engineering (CAE), software applications help and aid human beings to facilitate the work. Usually, several information systems are involved in the manufacturing process, the proposed framework could be implemented as a coordination system to solve interoperability problems from data and semantic levels. Also, we believe the proposed framework could contribute to Enterprise Resources Planning (ERP) systems. ERP is widely used in
enterprises for managing all resources, including manufacturing, production, human resources, finances, etc. The conceptual heterogeneity also exists among these systems, thus the framework can be implemented as one component to integrate into current ERP systems. It will help when redesigning and reengineering current systems, especially focusing on solving conceptual heterogeneity and enabling system interoperability.

In this article, the data source is focusing on relational database (RDB), to extend the work more generic, the data resources can be free texts documents and semi-structured data. Many researchers are working on this topic for extracting ontology from free texts, documents, etc. The framework is also applicable to this situation.

Due to great complexity and variety of real situations in engineering, we predict some challenges may face when applying the framework. First, the framework relies much on automatic extraction and automatic alignment, especially for large databases. The current tools are not mature enough to deal with complicated industrial systems, so this may lead difficulties to apply for databases with huge amount of data. This issue has been attempted in [70], many researchers are still working on it, such as [71, 72]. Second, since the RDB has been used for decades and is dominant in most engineering systems. Many enterprises rely on it and are not likely to make changes on it. Third, the performance of mapping accessibility between semantic layer and RDB could become a bottleneck in this architecture when there are huge and complex mappings. But we believe in foreseeable few years, as the development of periphery techniques and approaches, these issues will not be barriers any more.

6. Conclusion

Ontology is widely applied to engineering area for modeling and representing knowledge in a semantic way. Based on previous work done in this area, we described a practical ontology-driven framework towards constructing semantic information layer (SIL), aiming to fulfill the gap of data and semantic heterogeneity between heterogeneous systems. How to develop SIL and make it accessible has been fully described in the article. The framework could be implemented according to particular demands in each organization. It could contribute to solving information heterogeneity and interoperability issues in engineering systems as one promising alternative.

References


List of Figures

Fig. 1. Ontology languages stack
Fig. 2. a) Ontology alignment and b) Ontology integration and merging
Fig. 3. Framework for building Semantic Information Layer (SIL)
Fig. 4. Ontology extraction and enrichment
Fig. 5. a) Static mode and b) Dynamic mode in instance population
Fig. 6. Ontology alignment process
Fig. 7. Different levels for ontology alignment
Fig. 8. Structure of dmsOA
Fig. 9. Strategy selection flow
Fig. 10. Core part of class diagram
Fig. 11. Scenario of virtual enterprise
Fig. 12. Extracted object property and data property
Fig. 13. Snapshot of extracted instances in Protégé v4.1.0
Fig. 14. Number of discovered alignments filtered by threshold from 0.1 to 1.0

List of Tables

Table 1. Input and output of ontology reuse
Table 2. Ontology extraction rules (adapted from[25-32])
Table 3. Ontology alignment techniques
Table 4. Information of source databases
Table 5. Information of extracted ontologies
Table 6. Available alignments filtered by threshold = 0.8