Explication and semantic querying of Enterprise Information Systems

Milan Zdravković, Hervé Panetto, Miroslav Trajanović, Alexis Aubry

Explication and semantic querying of Enterprise Information Systems

Milan Zdravković

Laboratory for Intelligent Production Systems, University of Niš,
Faculty of Mechanical Engineering, Aleksandra Medvedeva 14, Niš, Serbia
milan.zdravkovic@masfak.ni.ac.rs
Hervé Panetto

Université de Lorraine, CRAN, UMR 7039, Campus sciences
B.P. 70239, Vandœuvre-lès-Nancy Cedex, 54506, France,
CNRS, CRAN, UMR 7039, France
herve.panetto@univ-lorraine.fr
Miroslav Trajanović

Laboratory for Intelligent Production Systems, University of Niš,
Faculty of Mechanical Engineering, Aleksandra Medvedeva 14, Niš, Serbia
miroslav.trajanovic@masfak.ni.ac.rs
Alexis Aubry

Université de Lorraine, CRAN, UMR 7039, Campus sciences
B.P. 70239, Vandœuvre-lès-Nancy Cedex, 54506, France,
CNRS, CRAN, UMR 7039, France
alexis.aubry@univ-lorraine.fr

Many researches show that the ability of independent, heterogeneous enterprises' information systems to interoperate is related to the challenges of making their semantics explicit and formal, so that the messages are not merely exchanged, but interpreted, without ambiguity. In this paper, we present an approach to overcome those challenges by developing a method for explication of the systems' implicit semantics. We define and implement the method for the generation of local ontologies, based on the databases of their systems. In addition, we describe an associated method for the translation between semantic and SQL queries, a process in which implicit semantics of the EIS's databases and explicit semantics of the local ontologies become interrelated. Both methods are demonstrated in the case of creating the local ontology and the semantic querying of OpenERP Enterprise Resource Planning system, for the benefit of the collaborative supply chain planning.

Keywords: Ontology; Systems Interoperability; Database semantics; Enterprise Information System; OpenERP.

1. Introduction

Traditional approaches for configuring high-speed, low-cost collaborative enterprises and corresponding technical requirements for tight systems integration appeared ineffective in the age of growing demand for product customization. Today, new enterprise collaborations with short lifecycles, such as virtual enterprises [1] and virtual breeding environments [2] or organizations [3] are often formed to meet the increasing demand for engineer-to-order products. Such loosely-coupled and temporary collaborations need to be facilitated by corresponding IT infrastructure, which exhibits the similar behavior. Hence, it is made of interoperating autonomous, heterogeneous Enterprise Information Systems (EIS), instead of mash-up of physically or functionally integrated systems.

ISO/IEC 2382 [4] defines interoperability as the "capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units". The notion of interoperability refers to the ability of heterogeneous, autonomous EISs to perform interactions, namely to exchange information and services [5]. Such paradigm is related to the federated approach, which implies that systems must accommodate on the fly in order to interoperate; in other words, no pre-determined assets for interoperability are assumed.

There is an agreement in the research community that ontologies can be used for the reconciliation of the interoperating EISs. Even so, there are opinions [6] that the main conditions for making two loosely coupled systems interoperable are: 1) to maximize the amount of semantics which can be utilized and 2) to make it increasingly explicit.

While "traditional" interoperability research is oriented to the use of ontologies as facilitators for reconciliation, the semantic interoperability focuses on data interpretation rather than mere data exchange, independently of implementation details [7]. Semantic interoperability means ensuring that the precise meaning of exchanged information is uniquely interpreted by any system not initially developed for the purpose of its interpretation [8][9]. It enables systems to combine and process received information with other information resources and even, to improve the expressivity of the underlying ontologies and consequently –

to increase the relevance of the data models which are formalized by those ontologies [10].

Domain ontology provides a general context for semantic interoperability of systems, namely, the language which they are going to use to communicate. Local ontologies introduce actual enterprises contexts to a collaboration environment. The capability of enterprises to efficiently collaborate between each other depends on the correspondence between their local semantics and the abovementioned general context. While this general context or the common language may be found in large number of relevant enterprise domain ontologies [11,12,13], the local semantics is not easily accessible, due to diversity and heterogeneity of the EISs landscape.

Three main problems for unveiling this local semantics are identified: it is hidden behind some design and development patterns, it is specified in implicit form and it is represented by using arbitrary syntax. As a motivation to resolve these problems, it is assumed that the corresponding solution will facilitate easier reconciliation of the local and domain dictionaries and hence, it will reduce the effort needed for achieving the semantic interoperability of the systems.

The main focus of the work, presented in this paper, is on the analysis of the source of the local semantics of the EISs, that is, relational database management systems and, consequently, its explication. Hence, the objectives of the research presented in this paper are to answer on the following questions: Where can the semantics of EISs be found? How can it be transformed to an explicit form? How can the common business concepts of the different explicit models be uniquely accessed and, consequently, processed?

In defining the methodological approach for answering these questions, we assume that: 1) the enterprises' realities are represented by the corresponding EISs' models, and 2) enterprises' message models (crucial for flexible economic integration) are based on EISs' data models, represented implicitly in their databases. The approach described in this paper aims at making this representation - explicit.

The research addresses some of the identified weaknesses of the existing approaches (see Section 2.2) to database to ontology mapping and aims at using the expressivity of OWL (The Web Ontology Language) language for enriching the implicit semantics of ER (Entity-Relationship) models. It delivers a method and a corresponding software tool which: 1) imports the database structure and classifies ER entities; 2) classifies (infers) OWL types and properties; 3) enables lexical refinement and 4) generates a local ontology. The concepts of the local ontology are then mapped backwards to the corresponding concepts of the intermediary models, for enabling the transformation of semantic to SQL queries. The method is described in Section 2.3.

Semantic querying of the databases cannot be considered independently from the method used to transform the database schema to a formal model. The main reason for this constraint is the dependence on the approach to mapping the concepts of the formal model to database data. Thus, the method for execution of semantic queries on the local ontology, namely, instantiation of its concepts according to the content of the relevant database, is developed and presented in Section 2.4.

The approach for generating a local ontology is implemented on the case of the OpenERP Enterprise Resource Planning information system. OpenERP is an open source suite of business applications including sales, CRM, project management, warehouse management, manufacturing, accounting and human resources. It is a client-server suite, where the client communicates with the server by using XML-RPC interfaces. It uses PostgreSQL relational database for data storage.

The case considers the generation of a local ontology, based on an ER schema of the OpenERP's database (Section 3.1) and querying of this local ontology (Section 3.2).

Previous work of the authors [14] introduced a theoretical perspective for an ontological framework for semantic interoperability of EISs in supply chain networks. The current paper extends this perspective by: providing the expanded details on the implementation of the methods for explication and semantic querying; discussing about the meaning of different constructs and design patterns of database development; validating the method in a case study. Finally, the discussion and experiences from the case study are used now to precisely define some research gaps that will emphasize the future directions of work.

It is important to note that the scope of the presented approach is limited only to selected ER patterns which are associated to semantics, expressed by the OWL constructs. Although the process overcomes some of the gaps, identified in the current state-of-the-art in database to ontology mapping, its end result typically requires a considerable amount of customization and additional work. Since direct mapping is unlikely to produce a useful ontology, the result of this analysis may be considered as intermediary. Thus, a complementary work can be done for semantically enriching this intermediary ontology by using some approaches as in [15].

2. Theoretical background

One of the major challenges in the efficient use of computer systems is the interoperability between multiple representations of reality (data, processes, etc.) stored inside the systems, or actual representations and reality itself – systems' users and their perception of reality [16]. Where the latter can be formalized by the domain ontologies, as shared specifications of the conceptualizations, the former relies upon the local ontologies – wrappers for heterogeneous sources of individual enterprises' information, business logic and rules.

The local ontologies formalize the implicit data from the heterogeneous sources in order to facilitate the semantic interoperability of the systems that store these data. In order to cope with the implicitness of semantics of the enterprises' realities, we assume that these realities are represented by the corresponding EISs, and that the enterprise message models are based on EISs' data models, represented implicitly in their databases. The proposed approach aims at making this representation - explicit. We employ a database-to-ontology method in order to transform implicit Entity-Relationship (ER) models to explicit OWL representations, namely, local ontologies.

Then, these local ontologies can be mapped to a common, shared knowledge of the enterprise collaboration environment, such as a formal model of supply chain, while other contexts may also be added. Each of the contexts corresponds to a domain ontology, whose concepts are logically related to the concepts of the local ontologies. Thus, each domain ontology becomes a dictionary – a common knowledge of particular enterprise perspective which can be used to query the hidden, implicit knowledge stored in EISs. Then, single and integrated access to the multiple contexts of the particular enterprise concept becomes possible.

The above assumptions about correlations between local ontologies and ER models are made for the purpose of making the process of local ontology creation – automatic. Otherwise, the precondition for this process would be a detailed

analysis of the involved EISs. An example of the work which follows this approach was provided by Castano and Antonellis [17]. The authors “analyzed the process descriptions for the aspects related to information and operation similarity, to evaluate semantic correspondences between processes and identify activity replication and overlapping, as well as for the aspects related to interaction/cooperation, to evaluate the degree of coupling between processes and identify the type and the nature of exchanged information flows”.
In our work, the range of semantic interoperability is clearly set to EISs. The semantic interoperability of the enterprises is considered as a more complex problem and is not addressed in this paper. The conceptualization of their information systems is made also on the basis of business logic, which is hidden in the actual code, in most cases, and data models, represented by the corresponding relational database structure. Obviously, the business logic which is encapsulated in the EIS’ will remain hidden – only the underlying data model is exposed by ontology. The exceptions are database triggers, which can be considered as business rules, if they are not implemented only to enforce referential integrity of the database. Even so, they represent some “kind of” semantics that has to be taken into account in the local ontology.
Another issue with the above assumptions is that, in the case of this approach, the domain of the conceptualization is restricted to database schemas. Sometimes, ER models, namely database schemas, do not capture the semantics of the application functionality and underlying data models; when information systems are highly generic, the application semantics is actually captured in the populated table rows. For example, in Business Process Management systems, the structure of the enterprise processes, i.e. activities, associated data structures (messages), compensation and error handling blocks, etc., are defined by a system user and are not expressed by the database schema. Moreover, those schemas are, sometimes, adapted to cope with implementation constraints and thus, they are losing part of their semantics.
This issue is evident even in trivial cases. For example, the attribute of “type” is often used by database developers to describe some entity. It is typically transformed to hasType(string) property. In this case, the meaning of this property is unknown, because of the ambiguity of the linguistic term of “type”. A similar remark can be made also for often used notion of “status”. However, sometimes,

the meaning of the ambiguous notion can be determined if the list of associated data (strings) in database rows is semantically analyzed in the context of the domain (entity) of the property above. For example, if OWL is used as a formalism, the “hasType some bNode” construct may be used to model this property, where bNode is an anonymous class that contains enumerated (owl:oneOf) elements which correspond to data associated to the attribute. In a more formal approach, the values of those attributes may be considered as classifiers of the subsumed classes. For example, the property hasType(string) of the concept Machine tool, asserted with one of the following values: “turning”, ”milling” and ”drilling” may enable the inference of the respective sub-concepts of the Machine tool concept - Lathe, Mill and Drill.

In the above cases, the intervention of the domain expert in enriching the conceptual model is inevitable [15]. Some research tackles this issue by providing the tools to automatically or semi-automatically discover the semantics buried into existing data patterns [18].

Although the conceptualization of the ER model is not a novel topic, the existing results did not provide the method which delivered a usable conceptual model. This is argued in the following subsections. It seems that most of the past work is motivated by the problems of database interoperability, which does not necessarily consider the semantics of the ER model. Systems interoperability, on the other hand, needs somewhat different approach; it aims at conceptualizing the intent of the schema developer and, thus, at making the reality of the enterprise information system – explicit.

2.1. Schema integration

Current research and practices of database interoperability are based on the earlier efforts in schema integration. Schema integration typically occurs [19] in the context of view integration (during database design) or in database integration (in distributed database management). The process of schema integration implies the development of a single - federal schema [20], expressed by using a common data model, for the purpose of integrating the schemas of existing or proposed databases into a global and unified one [19].

The mismatch between the schemas is caused by the fact that a single concept in the universe of discourse is sometimes represented in different ways, while there

are also cases where the single representation is associated to the meaning of different concepts. Typically, schema integration assumes that these conflicts are resolved in the process of schema transformation. This process is formalized by McBrien and Poulovassilis [21]. Its outcomes are equivalent schemas, which may then participate in the database federation.
It is important to note that most of the approaches to schema integration do not make an attempt to interpret or formalize the implicit semantics of the schemas. Instead, they are using a notion of common data model (which does not necessarily reflect an ontological commitment) to enable the federation of databases and thus, to make those interoperable.
With the development of the formalisms for semantics representation, the new approaches to database interoperability are increasingly focused on the transformation of the implicit semantics of the database schemas to explicit conceptual models. Many researchers have worked on schemas mapping [22][23] or data integration in ontology [24]. William et al [25] considered different groups of semantic relations between schema objects in order to find the corresponding similarities. Zhao and Ram [26] took into account the instance information in the process of integrating heterogeneous data sources. In one of the recent efforts, Ozgul and Afsarmanesh [27] used a variety of metrics and algorithms from the domains of Natural Language Processing and Graph theory for schema matching. In general, the existing approaches suffer of their applicability on existing large data sets. Moreover, the most of these approaches cannot be implemented in real cases because of the large amount of manual intervention. Some of the examples of the existing but practical work in database to ontology mapping are presented below.

2.2. Existing database to ontology mapping approaches and tools

The review of the relevant literature shows that many researchers dealt with the problem of database-to-ontology mapping, for different purposes and with different approaches. In this section, we present the main features of four distinctive frameworks, made with different objectives, and we identify gaps, in terms of the selected criteria. In particular, we are interested in how the existing frameworks address three specific aspects related to the database-to-ontology process: 1) semantic interpretation of E/R patterns, i.e. a level of database schema conceptualization; 2) instance population, i.e. ontology concepts instantiation; and 3) use of the framework, i.e. translation of semantic to database queries. As the latter two are mostly related to the technical challenges, we consider the level of database schema conceptualization as the most important.

The work on DB2OWL mapping facility is a part of development of a general interoperability architecture [28] that uses ontologies for the explicit description of the semantics of information sources, and web services for facilitating the communication between the different components of the architecture. DB2OWL [29] looks for some particular cases of database tables to determine which ontology component has to be created from which database component. According to these cases, the conversion process is performed (table → class, column→ property, constraint → relation) where the set of correspondences between database and ontology components is conserved, thus enabling the translation of ontological to SQL queries and retrieval of corresponding entities. However, it remains unclear how this translation will be implemented. More important, the semantics of existential constraints of the columns and cardinality of relations is not taken into account. The major feature of this approach, as claimed by the authors, is that it aims at separating data mapping from schema mapping. Any data manipulation with a database will not affect the ontology; the consistence of two corresponding data and sets of individuals will be maintained by the queries which will populate the ontology with instances at the moment of the semantic query execution. This method is referred to as a query-driven population, in contrast to a massive export (also referred to in literature as "massive dump"), which maintains the full correspondences between ontology individuals and database tables' data. The latter approach is taken by the Relational.OWL model.

Relational.OWL [30] is a candidate for data and schema representation format, relevant for database-to-ontology mapping, developed with a primary motivation to facilitate data and schema exchange in the Peer-To-Peer (P2P) database environment. It provides a meta-model, which describes the components of the relational database. In contrast to DB2OWL, it does not attempt to interpret the semantics of the E/R patterns. It does not conceptualize the E/R model but only provides its replica. However, it can be used as an intermediary in the process of database-to-ontology mapping, instead of a document with correspondences, used by DB2OWL. In that sense, it can be considered as a complementary work. Unfortunately, as it is the case for DB2OWL, it does not model multiplicity of the foreign keys. Thus, it is not possible to use it to assign source and destination cardinality to OWL properties.

Where DB2OWL and Relational.OWL are used to create new ontologies from existing schemas, there are tools that take a different approach by facilitating automatic, semi-automatic or manual mapping between existing ontologies and schemas. In this paper, we refer to the work of Konstantinou et al [31], and Xu et al [32].

Vis-A-Vis tool [31] uses the Protégé libraries for graphically representing ontology, a database model (MySQL or PostgreSQL) and it facilitates manual establishment of the mappings between them. Hence, it is not relevant to discuss conceptualization on the level of ER schema as it mainly depends on the outcomes of the manual work. The Protégé plug-in allows queries to be asked to the ontology and returns results from the database; it takes a query-driven approach to instance population. The key motivation of this approach is to keep the instances stored in a database while maintaining a link to the dataset, so ontologies become smaller.

In contrast to Vis-A-Vis that only facilitates manual mapping, D2OMapper [33] is a tool for automatic or semi-automatic creation of the mappings between database schema and existing ontology. This work is based on the authors' experience in developing ER2WO [32] tool for translating ER schema into OWL ontology. The key motivation of the authors was to develop a framework which would facilitate the generation of ontological annotations for dynamic Web pages, extracted from the database. D2OMapper outputs express the conceptual, in specific element (naming matching) and structural (predefined heuristic rules), correspondences between the schema and ontology. Although it is not explicitly mentioned in the reported work, the purpose of the approach implies that a query-driven approach to data population is taken.

Even though the topic of database to ontology mapping is still very active, it seems that no recent work addressed the semantic issues of ER schema, in detail. In contrast, more focus is given to use the ontologies (actually, semantic queries) to gain access to the large volume of data residing in the (often, distributed) database systems [34, 35, 36, 37, 38, 39] or to enable the exposure of the relational databases in the Linked Data environment [40] and discovery of links between open data sources [41]. In this context, the performance of reasoning becomes one of the dominant issues [42]; it is even addressed by scaling the reasoning capability from inference engines to database systems [43]. Other issues are related to making this large data more accessible, by increasing the expressivity [44] and robustness [45] of query languages or even to extending the representation languages (OWL) to support the integrity constraints from the relational databases [46]. The above trends correspond to the dominance of so-called lightweight semantics in meeting the promises of Semantic Web paradigm, especially driven by the recent success of linking open data initiatives [47].

We consider that most of the existing database-to-ontology methods are not suitable for generating the local ontology, which can then be used in the application framework for collaborative enterprises, for at least three reasons. First, and most important, they do not interpret the semantics of all ER constructs and patterns. Similarly, a remark can be made that the existing approaches do not use the full expressivity of the OWL language. The above statements are argued in this section, above. Second, approaches to instance population are not fully appropriate for use in the collaborative enterprise settings. This is elaborated in more detail in Section 3.2. Third, some of the authors claim that they provide a method for translating semantics to SQL queries, but no detailed information about this method is presented in their papers.

In our approach, we address the above-mentioned gaps by developing the presented explication and semantic querying methods. The major feature of this approach, in contrast to the existing research is the increased level of conceptualization. This is considered as vital for the outcomes of the semantic analysis, enactment and matching process, in which the local ontologies are typically semi-automatically related to a domain of interest.

3. Our approach to database-to-ontology mapping

Database-to-ontology mapping is a process in which the implicit semantics of a database schema is correlated to the explicit and formal knowledge structure of the ontology. In our approach, we use the database schema to generate this formal structure in section 3.1, while the logical mappings between the ER meta-model and generated local ontology are preserved. These mappings will enable the translation of semantic to database queries, as detailed in section 3.2.

3.1. Generating local ontologies

One of the objectives of the work presented in this paper was to increase the level of conceptualization detail of the resulting ontology, namely, to restrict possible use of concepts used in the description of the system, assumingly hidden in the database schema. Thus, this description would become more explicit and consequently, computable, in contrast to implicit meaning, which is expressed indirectly and needs additional inference (including additional facts to facilitate this inference). Our approach considers the existential constraints and cardinality of relationships to unhide some semantic features of the assumed local ontology, such as necessary conditions for a class, functionality of properties and uniqueness.

Existential constraints from the ER-model ("not-null") posed on the source of the foreign key reflects the intention of the database developer to enrich the description of the source with some destination concept or concepts. In other words, the former cannot exist without the latter. Thus, the relationship described by the foreign key can be considered as a necessary condition for a given concept. More important, the meaning of the source concepts can be attributed to these necessary conditions. This approach to a conceptualization is referred to as intensional, and is considered as equivalent to human thinking [48], in contrast to extensional approach, which implies that the elements of the mental image of the specific domain are simply enumerated or listed. A special case of a necessary condition may be defined by using functional or single-valued properties. A functional property is a property that can have only one (unique) value y for each instance x. The functional properties are established between two concepts when two corresponding tables are related with source and destination cardinality which equals 1.

The uniqueness attribute of the database fields is used to improve the performance of the inferences on the corresponding local ontologies. One of the consequences of Open World Assumption [49] of the description logics based languages, such as OWL, is the inability to assume the difference of two individuals, unless it is explicitly stated by using owl:differentFrom relation. Imposing such a relation between all the individuals that correspond to the database rows may significantly decrease the reasoning performance. In our approach, this problem is addressed by assigning owl:hasKey properties to the concepts, whose corresponding tables have "unique" constraint posed on their fields and thus, making those explicitly different.

The generation process consists of 4 phases: a) data import and classification of ER entities; b) classification (inference) of OWL types and properties; c) lexical refinement; d) generation of local ontology.

Before the generation of local ontology, it is necessary to deliver two intermediary models. The first model is OWL replica of a database schema, a database ER model. Its primary role is to store the references to the actual database artifacts which will be exploited later to execute the semantic queries. It can also be considered as an input to other explication or discovery tools (for example, in lightweight semantics' Linked Data environment), because it uses OWL to represent all the artifacts of the database schema, similar to Relational.OWL approach, presented above. The second model – a meta-model, which classifies OWL types and properties is used as a subject of lexical refinement, if necessary.

The process of local ontology generation is supported by a web application which consists of modules for data import/assertion of ER meta-model instances, lexical refinement and transformation of classified OWL types and properties to a local ontology. The web application requires input of the database connection parameters. First, two intermediary models are created, where the generation of the meta-model is facilitated by firing the conversion rules (classification of the meta-model concepts is done by inference engine, based on these rules). Then, user can download both of the models and/or choose to proceed to a lexical refinement (phase 3) module or to finalize the process by initializing the local ontology generation (phase 4). Different phases of the local ontology generation are illustrated on Figure 1. The details of the process are presented below.

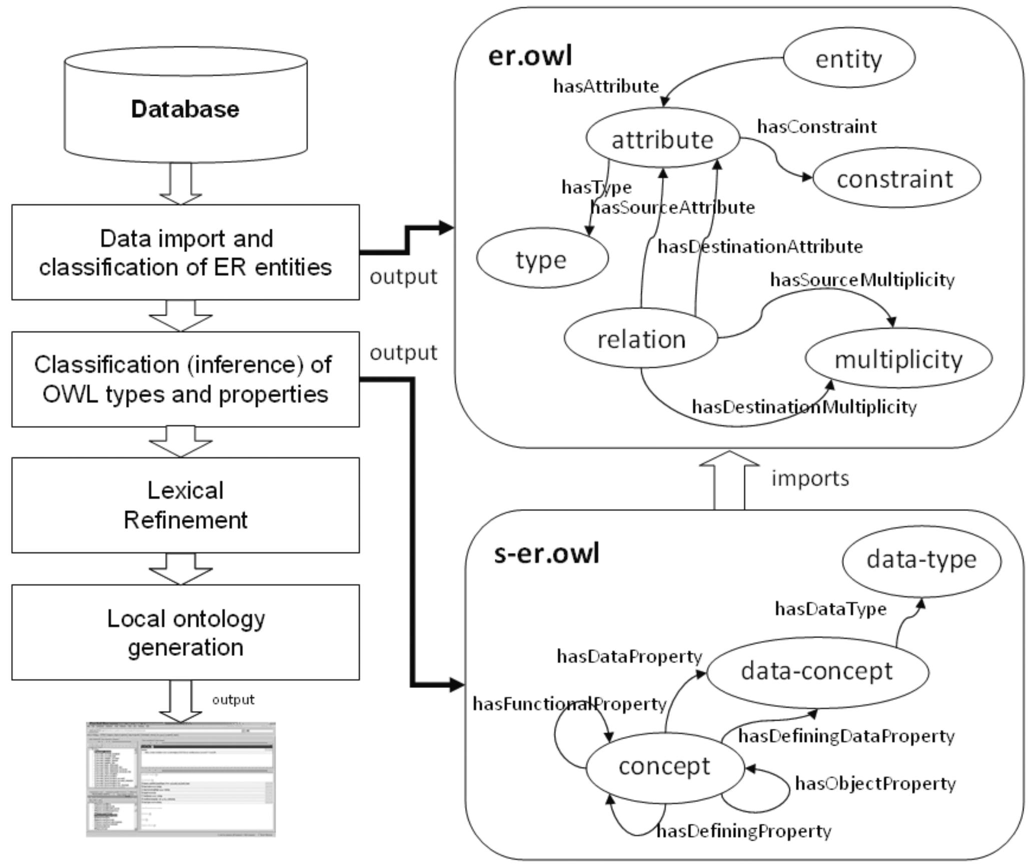


Figure 1: Approach to database-to-ontology mapping

First, the database schema is investigated and OWL representation of the ER-model is constructed. In this step, a web application connects to the database, uses introspection queries for discovering its structure and asserting the relations between the artifacts by using the proposed ER formalization (namely, er.owl). The following assertions are made for each field of the corresponding table: hasAttribute (entity, attribute), hasType(attribute, type) and hasConstraint(attribute,'not-null') and/or hasConstraint(attribute,'unique') (if applicable). The following assertions are made for each relation: hasDestinationAttribute (relation, attribute), hasSourceAttribute(relation, attribute).

Second, the resulting (serialized) OWL representation of the database ER-model is imported into the meta-model (s-er.owl), which classifies future OWL concepts (conversion rule R1, below) and domains and ranges of the object and data properties, according to the defined conversion rules (rules R2 and R4, below). Although the specifications of object and data properties may impose the unnecessary restrictions on the resulting ontology, we consider those as important for improving the efficiency of mapping or alignment process, which is critical for the semantic interoperability. Another reason for the assertion of object properties in OWL representation of database ER-model is that the object properties of the resulting local ontology will be annotated with the URI's (Uniform Resource Identifiers) of the respective relations, in order to enable the correspondence between the ontology and database representation, for the benefit of query transformation.

According to the above constraints, the rules for intensional conceptualization (inherited anonymous classes) for a particular entity are identified by inferring ranges of hasDefiningProperty(concept, concept) and hasDefiningDataProperty(concept, data-concept) relations (rules R2.2 and R4.2, below). Finally, the approach takes into account the functionality of the properties (owl:FunctionalProperty). Functional properties are classified when a one-to-one relation is identified between two concepts (rule R2.3, below).

The classification of future OWL concepts is then inferred by exploiting the following conversion rules:

R1. Concepts are all entities of the ER model's OWL representation, except the entities whose all attributes are relation sources (corresponding to intermediary tables, connecting two tables with many-to-many relationship).

er:entity(x) $\wedge$ not (er:hasAttribute only (er:attribute $\wedge$ (er:isSourceAttributeOf some er:relation))) $\Rightarrow$ s-er:concept(x)

R2.1. Domains and ranges of the object properties are inferred by using the rule below.

er:entity(x) $\wedge$ er:entity(y) $\wedge$ er:relation(r) $\wedge$ er:hasAttribute(x, a1) $\wedge$ er:hasAttribute(y, a2) $\wedge$ er:isDestinationAttributeOf(a2, r) $\wedge$ er:isSourceAttributeOf(a1, r) $\Rightarrow$ s-er:hasObjectProperty(x, y)

R2.2. Domains and ranges of the defining properties (necessary conditions of the concept) are inferred by using the rule below. The defining property is a sub-property (rdfs:subPropertyOf) of the object property (hence, simplified representation of the rule below).

s-er:hasObjectProperty(x, y) $\wedge$ er:hasConstraint(a1,'not-null') $\Rightarrow$ s-er:hasDefiningProperty(x, y)

R2.3. Domains and ranges of the functional properties are inferred by using the rule below. The functional property is a sub-property (rdfs:subPropertyOf) of the defining property (hence, simplified representation of the rule below).

s-er:hasObjectProperty(x, y) /\ er:hasConstraint(a1,'not-null') ⇒ s-er:hasDefiningProperty(x, y)

R3. Data concepts are all attributes of the ER model's OWL representation which are not at the source of any relation.

er:attribute and not (er:isSourceAttributeOf some er:relation) ⇒ s-er:data-concept

R4.1. Domains and ranges of the data properties are inferred by using the rule below. Ranges of the data properties are data types, corresponding to the simple types from XML schema.

er:type(x) ⇒ s-er:data-type(x)

s-er:concept(c) /\ er:attribute(a) /\ er:type(t) /\ er:hasAttribute(c, a) /\ er:hasType(a, t) ⇒ s-er:hasDataProperty(c, t)

R4.2. Domains and ranges of the defining data properties are inferred by using the rule below. The defining data property is a sub-property (rdfs:subPropertyOf) of the data property (hence, simplified representation of the rule below).

s-er:hasDataProperty(c, t) /\ er:hasConstraint(a,'not-null') /\ er:hasConstraint(a,'unique') ⇒ s-er:hasDefiningDataProperty(c, t)

The above conversion rules are specified in s-er.owl by using SWRL. SWRL (Semantic Web Rule Language) [50] is a proposal for a Semantic Web rules-language, combining sub-languages of the OWL with those of the Rule Markup Language (RuleML). Below are some examples of SWRL representations of the conversion rules.

(R1) entity(?e), hasAttribute max 0 attribute(?a), isSourceAttributeOf some relation(?r) -> concept(?e)

(R2.1) entity(?e1), entity(?e2), relation(?r), attribute(?a1), attribute(?a2), hasAttribute(?e1,?a1), hasAttribute(?e2,?a2), isDestinationAttributeOf(?a2,?r), isSourceAttributeOf(?a1,?r)->hasObjectProperty(?e1,?e2)

(R2.2) entity(?e1), entity(?e2), relation(?r), attribute(?a1), attribute(?a2), hasAttribute(?e1,?a1), hasAttribute(?e2,?a2), isDestinationAttributeOf(?a2,?r), isSourceAttributeOf(?a1,?r), hasConstraint(?a1,"not-null")->hasDefiningProperty(?e1,?e2)

The rules are stored in the meta-model and once fired, they classify instances of the OWL representation of the database ER model (er.owl) into the concepts of meta-model (s-er.owl). Inferred triples in the meta-model can then be edited in a simple web application (lexical refinement), which also launches the process of local ontology generation. In this process, meta-model entities are finally transformed into corresponding OWL, RDF and RDFS constructs – a resulting local ontology. Concepts of the generated local ontology are annotated with URI's of the corresponding ER entities from er.owl model, so that the translation of semantic to SQL queries may become possible.

The process of transformation is illustrated on Figure 2. It shows the fragment of the OpenERP database (a), its meta-model generated by firing the rules above (b) and corresponding concepts of the resulting local ontology (c). The database ER model is not illustrated, because it is only a replica of ER diagram (a).

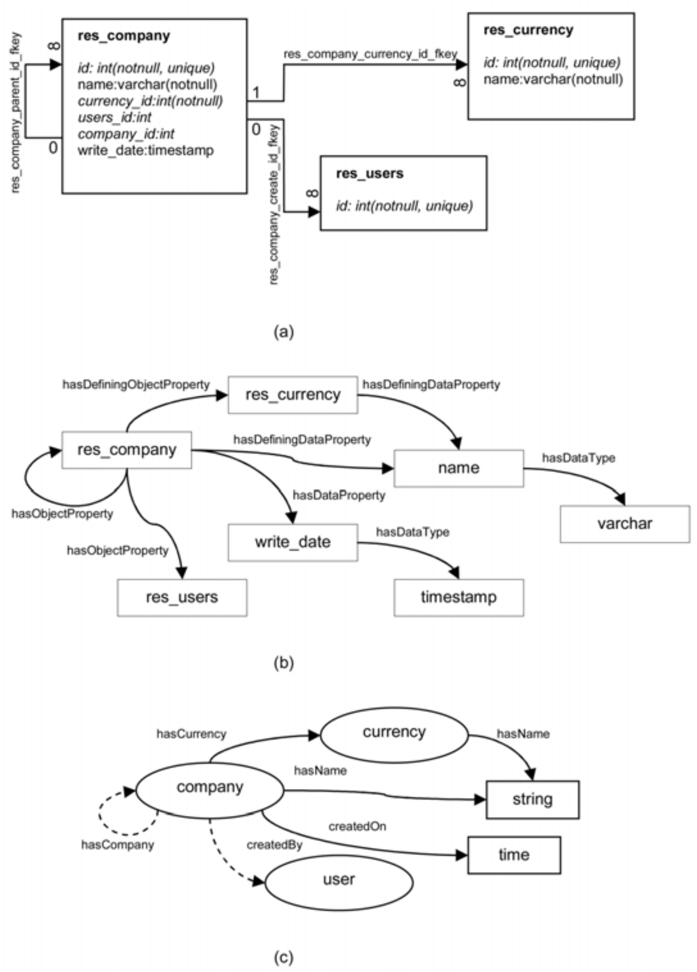


Figure 2: Illustration of the process of local ontology generation

Based on the design of the illustrated fragment of the database (Figure 2a), entities and data entities of the database ER model are classified (Figure 2b) as concepts (Rule 1), data concepts (R3) and data types (R4.1); the focal concept of the fragment – "res_company" has two object properties (R2.1), data property (R4.1), defining data property (R4.2) and defining object property (R2.1 and R2.2).

On Figure 2c, the dashed lines represent the constraints posed by the defined domains and ranges of the introduced properties ("hasCompany" and "createdBy"). Solid lines represent necessary conditions for a class, namely the defining object ("hasCurrency") or data properties ("hasName", "createdOn").
In the phase of lexical refinement, the coding style, used by the database designer is handled (e.g. "res_company" is renamed to "company"). Also, at this point, the meaning of some relationships can be associated to the concepts of local ontology by using proper lexical terms. For example, the meaning of the relationship between "res_company" and "res_users" concepts of the meta-model is implied only by the name of the relation between two corresponding tables: "res_company_create_id_fkey". Thus, in the manual intervention during the lexical refinement, the automatically proposed title of the relationship between company and user concepts – "hasRes_users" is replaced with "createdBy".
Another challenge for the development of local ontologies is related to instance population, namely, to how and when database data is represented in the local ontology. As it is mentioned before, two types of approaches are applied in the reported work. Massive export assumes that all data is represented as individuals in the process of ontology generation (or mapping of existing ontology with a database schema). Besides obvious maintenance related difficulties, this type of approach is unacceptable mainly because of the size of the resulting ontology and the mapping document and, consequently, performance issues related to reasoning processes. Query-driven population approach assumes that individuals are asserted to ontology during exploitation, upon execution of a semantic query. Here, some kind of query rewriting mechanism is involved to transform the semantics to SQL query or queries which are executed in the database; result-sets are then represented as logical statements which are finally asserted to local ontology. For many purposes, the existing query-driven approaches to population seem good candidates. However, when semantic interoperability between diverse and heterogeneous EISs is discussed, we believe that there are some concerns, mostly related to the complexity of inferences when modular ontological framework is queried and handling of data access rights. Those concerns are elaborated in Section 3.2.

3.2. Reasoning with local ontologies and translation of semantic to SQL queries

Semantic interoperability of systems enables a single point of access to the overall knowledge of the “interoperable world”. Not only that it makes possible to use a single semantic query to extract and combine relevant information from the multiple sources of implicit data, but it also enables the usage of different dictionaries for writing this query.

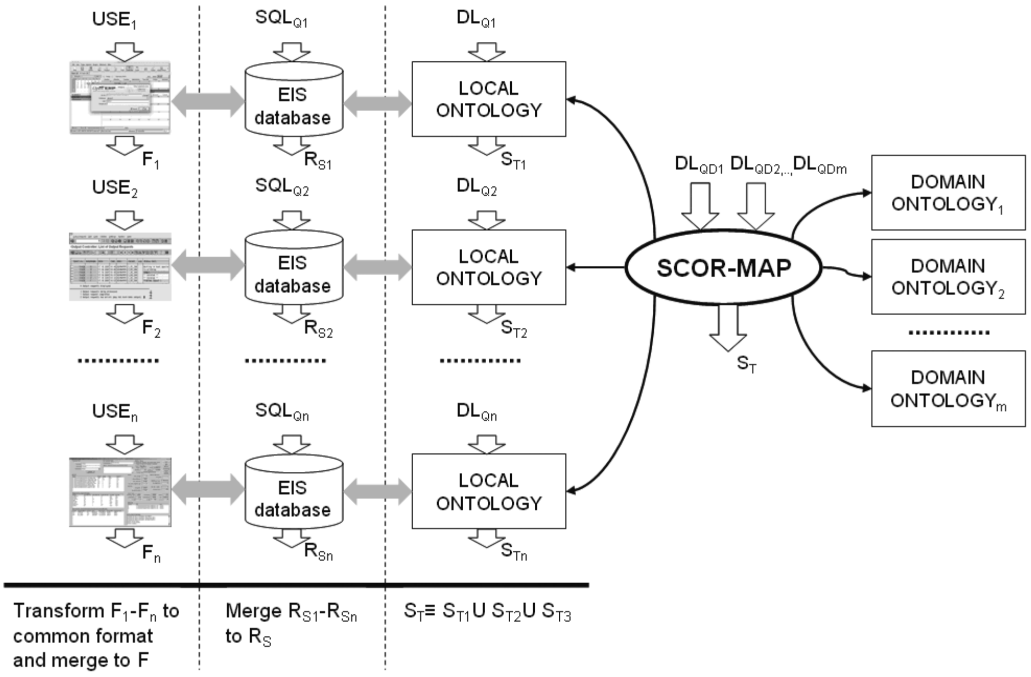


Figure 3: Extraction of data from heterogeneous sources

Figure 3 illustrates how the data is extracted from heterogeneous sources by using three different approaches: 1) simple use of EISs; 2) merging the relevant result-sets from the databases; and 3) executing semantic queries. In the first case, one can use (USE_i) the EISs’ data exchange facilities to export data files (F_i) and then transform each of the files to a common format and merge. In the second case, the SQL queries (SQL_{Qi}) are executed against EISs’ databases to get relevant result-sets (R_{Si}) and then merge. In the case of semantic queries data extraction, and if the assumption that logical mappings between local and domain ontologies are consistent and complete holds true, a single DL query (DL_{Qi}) can be constructed by using any dictionary, formalized by the domain ontologies, to extract the same data, but represented in different ways, depending on the used formalisms. Thus, no matter which dictionary is used to build the query, the result of its execution will be the union of semantically equivalent sets of triples (S_{Ti}).

In this section, we describe the method for instance assertions to a local ontology on the basis of semantic query results. The method consists of the following steps: 1) decomposition and analysis of the semantic query; 2) data extraction and instance assertions and; 3) reasoning. The method is illustrated in Figure 4.
A semantic query can be considered as a pair (O, C), where O is a set of concepts which need to be inferred and C is a set of restrictions to be applied to their properties, namely, value (owl:hasValue and qualified cardinality restrictions, owl:allValuesFrom, owl:someValuesFrom) and cardinality constraints (owl:cardinality, owl:minCardinality, owl:maxCardinality). This consideration corresponds to a simplified representation of a SQL query which includes tables (and fields) and comparison predicate, that is, restrictions posed on the rows returned by a query. In addition, different types of property restrictions correspond to different cases (or patterns, where complex semantic query is mapped) of SQL queries.
Since relevant entailments can be reasoned only by the property domain and range inferences, a set C may be considered as necessary and sufficient for representing the semantic query. For example, in the local ontology that is generated from the database schema of OpenERP EIS (see Section 4), a DL query "hasAccountAccountType some (hasCode value 3)" returns all instances of account_account concept whose type code is exactly 3. This kind of query representation (only by using properties restrictions) may produce unpredictable and misleading results when the restrictions are posed on the common lexical notions of different concepts, such as "name", "type", "id", etc. The ambiguity of the corresponding properties is reflected on the relevant ontology in the sense that their domains are typically represented as a union of large number of concepts. For example, in OpenERP ontology, the domain of the "hasName" data property is the union of 170 concepts.
However, this ambiguity may be considered as an advantage in some cases. Value restrictions on ambiguous data properties may produce relevant inferences, thus facilitating semantic querying without a need to have extensive knowledge of the underlying ontology structure. This kind of query is mapped to a SQL UNION query which combines SELECT sub-queries made on each element of the property domain, with the WHERE statement corresponding to the relevant rows restrictions. For example, in a mapping process, DL query "hasName value

‘Derek Porter’” is first used to infer all 170 possible entailments (property domains) in OpenERP ontology, which are, then, used to assemble qualified (O,C) pairs, e.g. “res_users and hasName value ‘Derek Porter’”. When the corresponding element of the UNION query is assembled, a static field with appropriate label (a reference to the concept) is added to each of the elements, so as to become possible to decide on the entailments. In other words, we need this to determine which sub-query actually returns the results.

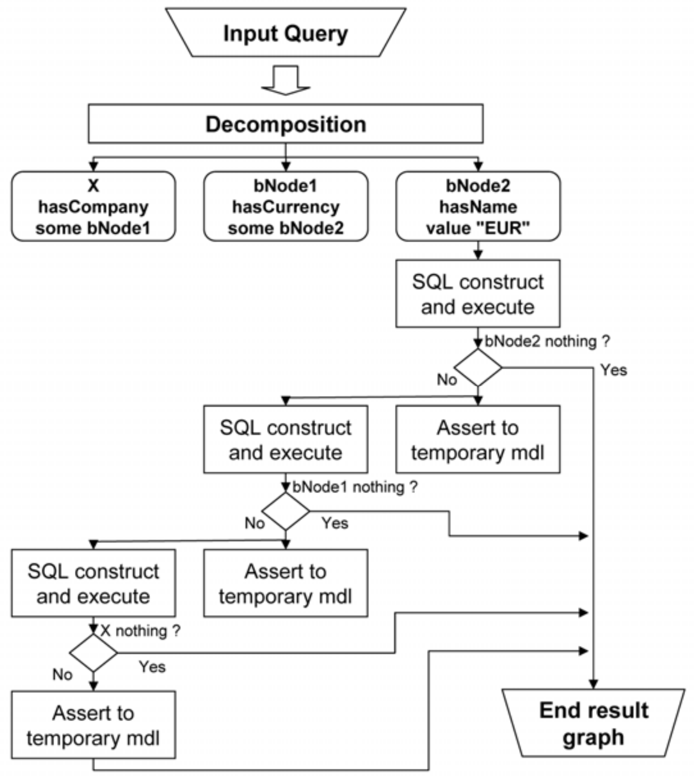


Figure 4: Execution of the example semantic query in local ontology

In the first step of the method, decomposition and semantics analysis of the input query is performed. The 4-tuples in forms of (subject predicate some|only|min n|max m|exactly o bNode) and (subject predicate value {type}) are extracted from the input query. In the case of the DL query which returns all concepts which are related to a company whose primary currency is EURO (“hasCompany some (hasCurrency some (hasName value "EUR"))”), the following 4-tuplets are identified:

X hasCompany some bNode1

bNode1 hasCurrency some bNode2

bNode2 hasName value "EUR"

In some cases, more complex queries may be needed to define the requirements of the user. This occurs when multiple restrictions on a desired object are given, so that the intersection of two or more sets, corresponding to these restrictions, is taken into account. For example, all payable accounts for companies whose primary currency is EURO are inferred by using DL query: hasAccountType value "Payable" and hasCompany some (hasCurrency some (hasName value "EUR")). In this case, the following 4-tuples are identified:

X hasAccountType value "Payable"

X hasCompany some bNode1

bNode1 hasCurrency some bNode2

bNode2 hasName value "EUR"

In the next step of semantic query execution, a database connection is established and sets of SQL queries are constructed and executed for each element of a 4-tuple, in reverse order, as a result of analysis described above. Each query returns data which is used to generate OWL statements which are asserted to a temporary model. Each set of the OWL statements corresponds to a sub-graph whose focal individual is an instance of the concept, inferred on basis of the property domain or returned result (label) of a 4-tuple. Other individuals or values correspond to the defining properties of this concept (inherited anonymous classes). In the case of ambiguity, the resulting blank nodes are represented as sets, which are filtered as a result of range inference of the parent 4-tuple, in the final stage of the method.

As it is shown in Figure 4, the output of the process of semantic querying of local ontology is a set of OWL triples which formalizes the parts of the local ontology, asserted with individuals whose properties match the restrictions, defined by the DL query.

Obviously, a query-driven population is applied in this case. As it is mentioned before, this approach separates data from the meta-model and, hence, it enables better performance of the reasoning processes. However, at this moment, a query-driven population cannot be applied in a more complex environment of interrelated ontologies, such as the scenario of semantic interoperability of EISs. In the remainder of this section, we discuss the two main arguments for this statement.

Almost all of the work on semantic reasoning still assumes a centralized approach where all inferences are carried out on a single system. The consequence of this approach is that all ontologies that need to interoperate (typically interrelated by "imports" relations) have to be loaded by the reasoner software before the inference is even started. In a semantic interoperability scenario, the reasoner uses asserted logical correspondences between the local ontologies and the domain ontology to infer about the individuals of the local ontologies by using the language of the domain ontology. Since all ontologies need to be loaded into the memory space of the reasoner, it is not possible to apply a query-driven approach because the database is not accessible. This issue may be resolved by enabling more flexible and dynamic imports, where, for example, imported local ontologies are populated by dynamic services, capable of processing restrictions from the semantic query executed in the parent ontology. At this moment, we are not aware of any efforts of the scientific community to tackle this problem. A possible workaround for resolution of this problem may be the usage of formally-defined interfaces which allow that different ontology modules are developed completely independent of each other's signature and even a language [51].

Another issue of the query-driven population of local ontologies in inter-organizational settings is data security, namely, access authorization. In a massive export population approach, specific export and synchronization rules may be implemented to publish only some parts of the EIS's database to a local ontology. However, a query-driven population, as explained above is done at the runtime, when the query itself is executed. Hence, it is very difficult to implement and manage access rules. Even, a more complex, but realistic scenario can be imagined, where an enterprise wants to manage the access to particular information per request in the process of a query execution. It is important to note that, in this case, the process of semantic querying will become asynchronous. Again, it seems that no relevant work on this topic has been done so far.

Despite the fact that the above concerns are serious, we still believe that the query-driven population is a better candidate approach for application in semantically interoperable EISs than the massive export. The problems of static and restricted imports are mainly related to technical challenges, which are expected to be faced more likely than performance issues of DL-based reasoners. The problems of access rights may be resolved by considering a middleware which will be used for implementing the cross-organizational processes with strictly defined information access policy. For example, the cross-organizational process may combine customized views on internal business processes that hide their private internal details [52].

4. Explication and semantic querying of OpenERP Enterprise Information System

Our approach for generating a local ontology and semantic querying is implemented on the case of extracting the semantic information from the OpenERP enterprise information system. This example is a part of the greater case study which demonstrates the benefits of the semantic interoperability of systems for a lifecycle management of the virtual enterprise for manufacturing of the custom orthopedic implants [53]. These, engineered-to-order, highly customized products are vital for the effectiveness and efficiency of the clinical practice. Today, they are rarely produced because of extremely long lead time, even up to three months, and corresponding high costs. The above cited case study shows how the uptake of the relationship management in the cost and lead time can be reduced by enabling the efficient communication between partners in the dynamic supply chain.

In this example, we show how the enterprise with a customer role in a virtual enterprise for custom orthopedic implants manufacturing (or any other engineered-to-order product) could gain a transparent access to the information needed for production planning. In specific, we demonstrate how the presented methods for explication and semantic querying could be used to by the customer to achieve access to the production schedules of its supplier for a given part, and hence, to increase the efficiency of its production planning processes..

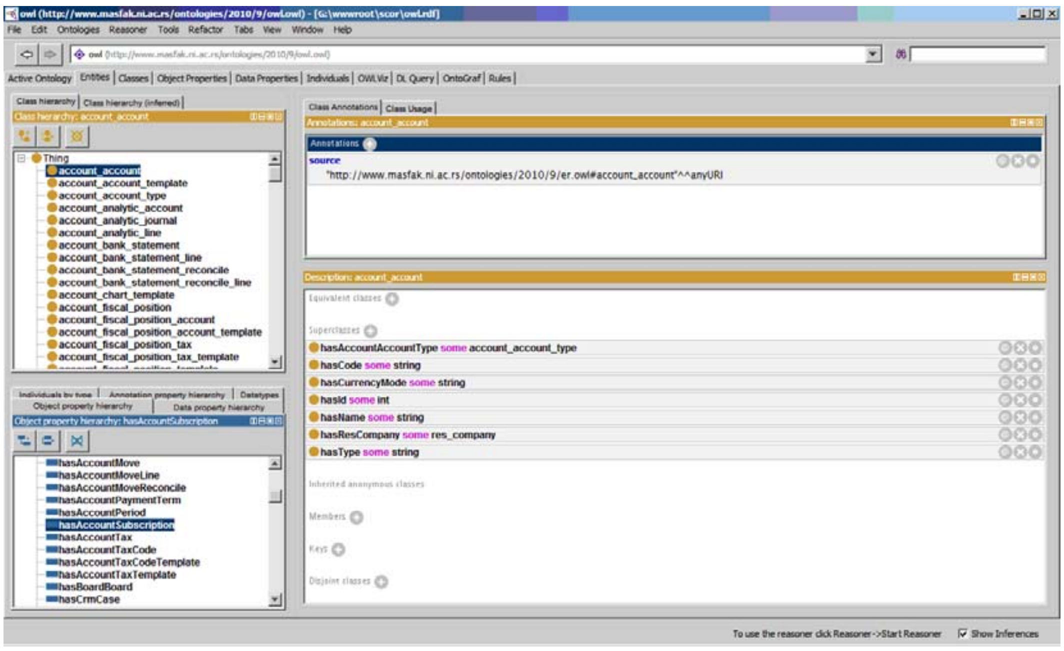
First, it is demonstrated how the supplier is represented into the “interoperable world”, namely, how our database-to-ontology method is used to generate a local ontology for OpenERP enterprise information system. Then, it is shown how the semantic query for extracting the production schedule for a given part is executed in this local ontology.

4.1. Generation of OpenERP local ontology

With all modules installed, OpenERP database counts 238 tables. The database is transformed to an OpenERP local ontology by the software prototype that implements the approach, described in Section 3.1. In the first step of database import into er.owl model, namely, instantiation of the OWL representation of the ER model, 3806 individuals are created (2633 of "attribute" type, 238 of "entity" type and 934 of "relation" type) and 7999 object property assertions are made. These individuals and their asserted properties directly correspond to the structure of OpenERP database schema and they are their literal OWL representation.

In the second step of the transformation process, the classification of OWL concepts and properties is done and s-er.owl model is generated. 193 concepts, 493 data-concepts and 2779 properties are inferred, on the basis of the SWRL rules, presented in Section 3.1, executed on the literal OWL representation produced in the former step. All inferences are stored in a separate OWL file, which is considered as the meta-model of the OpenERP database schema, in order to reduce the processing requirements for the final step.

In the final step of the local ontology generation, the software transforms the classified instances of the meta-model of the OpenERP database to the corresponding OWL concepts and properties (see Figure 5).


owl (http://www.masfak.ni.ac.rs/ontologies/2010/9/owl.owl) - [G:\wwwroot\scor\owl.rdf]
File Edit Ontologies Reasoner Tools Refactor Tabs View Window Help
owl (http://www.masfak.ni.ac.rs/ontologies/2010/9/owl.owl)
Active Ontology | Entities | Classes | Object Properties | Data Properties | Individuals | OWLViz | DL Query | OntoGraf | Rules
Class hierarchy | Class hierarchy (inferred)
Class hierarchy: account_account
Class Annotations | Class Usage
Annotations: account_account
Annotations
Descriptions: account_account
Equivalent classes
Superclasses
Inherited anonymous classes
Members
Keys
Disjoint classes
Individuals by type | Annotation property hierarchy | Datatypes
Object property hierarchy | Data property hierarchy
Object property hierarchy: hasAccountSubscription
To use the reasoner click Reasoner->Start Reasoner
Show Inferences

Thing
account_account
account_account_template
account_account_type
account_analytic_account
account_analytic_journal
account_analytic_line
account_bank_statement
account_bank_statement_line
account_bank_statement_reconcile
account_bank_statement_reconcile_line
account_chart_template
account_fiscal_position
account_fiscal_position_account
account_fiscal_position_account_template
account_fiscal_position_tax
account_fiscal_position_tax_template

source
"http://www.masfak.ni.ac.rs/ontologies/2010/9/er.owl#account_account"^^anyURI

hasAccountAccountType some account_account_type
hasCode some string
hasCurrencyMode some string
hasId some int
hasName some string
hasResCompany some res_company
hasType some string

hasAccountMove
hasAccountMoveLine
hasAccountMoveReconcile
hasAccountPaymentTerm
hasAccountPeriod
hasAccountSubscription
hasAccountTax
hasAccountTaxCode
hasAccountTaxCodeTemplate
hasAccountTaxTemplate
hasBoardBoard
hasCrmCase

Figure 5: OpenERP local ontology in Protégé

The resulting OWL file is considered as the output of the described database-to-ontology transformation process. In the case of OpenERP, additional work on lexical refinement is not vital because the database developers use natural language to describe the entities and their attributes.

The resulting conceptualization, that is, the generated local ontology corresponds to the user perspective of OpenERP system. This is demonstrated below, in the

description of the manufacturing module of OpenERP system. The lexical refinement is intentionally skipped in order to better illustrate the process of semantic querying (by preserving the similarity of the database tables and concepts' and relationships' names).
The manufacturing module of OpenERP EIS facilitates the management of master data about products, master Bill of Materials, work centers and routings; it automates procurement management, manufacturing and purchase scheduling; it facilitates the management of manufacturing and delivery orders and after-sales services. Figure 6 displays the fragment of the UML representation of OWL concepts and relations (from the generated local ontology) that describe the manufacturing module of OpenERP EIS.
The basis for manufacturing management in OpenERP is the management of master data, namely, bills of materials, work centers and routings. Bills of materials (mrp_bom concept in Figure 6) describe the single or multi-level structure of the product (product_product concept) to be manufactured – sub-assemblies or raw material, each of which can be moved from stock or manufactured or purchased (determined by hasType functional property of mrp_bom concept). Work centers (mrp_work_center) represent units of production (machines or human resources, determined by hasType functional property), capable of doing material transformation operations, with a certain production capacity, expressed in cycles (for machines) or hours (for human resources). Routings (mrp_routing) define the manufacturing operations to be done in work centers to produce a certain product. They are associated to bills of materials.
Once the master data is defined, the system can automatically generate the production schedule (schedule of generation of production – mrp_production, and procurement – mrp_procurement orders) by using make-to-order rules, minimum stock (for make-to-stock production) rules or production plan (based on forecasts). For make-to-order production, orders are computed on the basis of quantity of the ordered product, bill of material and delivery date. For each of the product's elements which are supplied, a procurement order is generated. Planned dates (hasDatePlanned property) for the orders are calculated on the basis of a delivery date and manufacturing and purchase lead times for the product elements. For make-to-stock production, instead of the delivery date, minimum stock rules

are used for production scheduling. In this case, orders are launched when minimum stock thresholds are reached.

The logistics of production is managed on the basis of stock moves (mrp_stock_move concept). OpenERP supports three types of stock locations: physical stock locations (warehouses), partner locations (customers' and suppliers' stocks) and virtual locations. The notion of stock location is used to define pull and push flows and to manage all types of storage places, including internal, supplier, customer, production and others. It is used to manage manufacturing logistics, since each of the manufacturing operations (described by mrp_routing concept) can be associated to a single stock location.

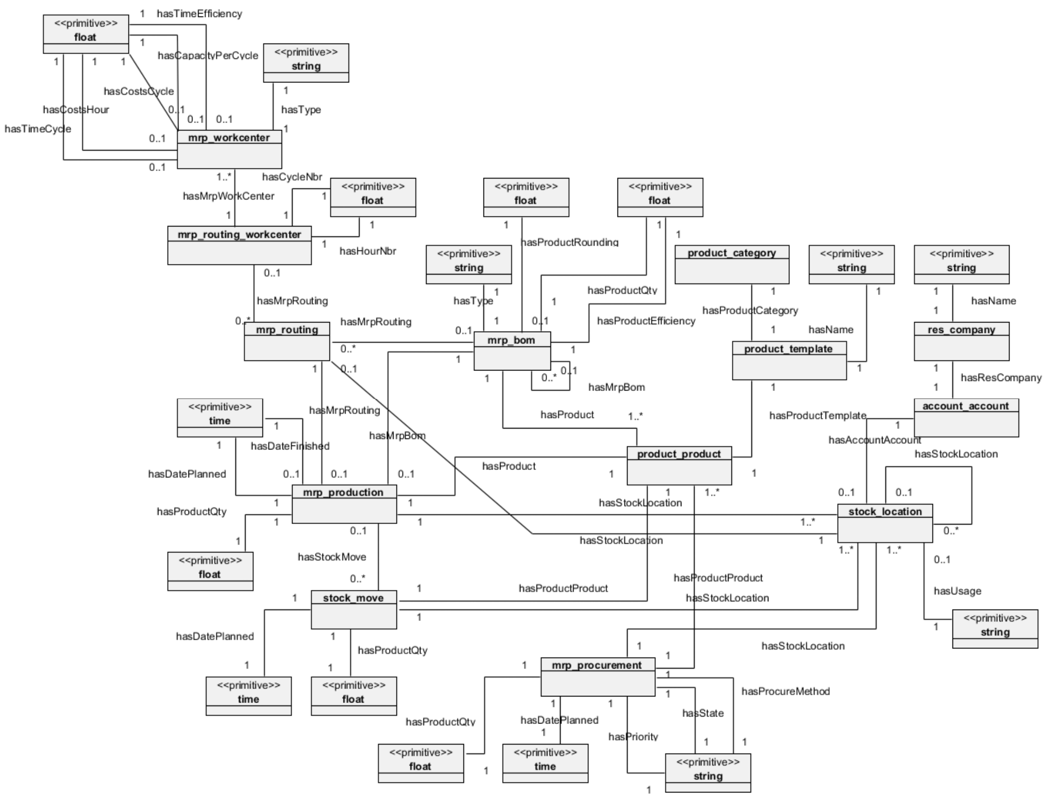


Figure 6: Fragment of UML representation of OpenERP local ontology

The cardinalities of the properties on the UML representation of OpenERP ontology highlight the specific semantic features of the illustrated concepts, which are not clearly obvious when the ER model of the database is considered. For example, the bill of material cannot exist without an associated product or products; thus, (hasProduct product_product) is a necessary condition for the mrp_bom concept. The bill of material must have a type assigned (hasType functional property). It may (or may not) be associated with a parent bill of

material element (hasMrpBom property). Similar reasoning can be applied for the other concepts of OpenERP ontology.
The above description of how OpenERP system works with manufacturing management corresponds to the conceptual model of this domain, illustrated in Figure 6. However, although the principles above are used to manage the production in many other (if not all) ERP systems, they are all realized by the different database schemas. The differences in conceptualization approaches of the ERP systems designers have a negative effect on the capabilities of these systems to cooperate. This problem is typically addressed by making two conceptual models correlated. However, the reconciliation of different semantic models, such as different explicit representations of the implicit realities of two EIS, namely, local ontologies and conceptual models of a specific domain, are beyond the scope of this paper.

4.2. Execution of semantic queries

Once the local ontology of OpenERP system is generated, our method for semantic querying of the local ontologies can be applied to facilitate the extraction of the relevant information. In order to demonstrate this, we consider a case in which a manufacturing enterprise queries the local ontologies of its suppliers in order to extract the information about a production schedule for a specific part of the custom orthopedic implant – an inner fixture.
In the query-driven population approach, two types of query re-writing mechanisms are needed. The first query needs to transpose the semantic query, written by using the language of domain ontology, into another semantic query, which can be then executed on the local ontology. The second type of query rewrite mechanism is needed to transform the semantic query to SQL query or queries which are executed in the database; result-sets are then represented as logical statements which are finally asserted to local ontology.
The DL query which returns the production schedule for the product (part) with name "Custom fixture F12" from the local ontology of OpenERP system is:
mrp_production and hasProductProduct some (hasProductTemplate some (hasName value "Custom inner fixture F12"))
According to the method, in the first step of semantic query execution, the query is decomposed to the following 4-tuplets:

X hasProductProduct some bNode1
bNode1 hasProductTemplate some bNode2
bNode2 hasName value "Custom fixture F12"

In the next step, SQL queries are generated for each of the 4-tuplets, from the bottom up. The domain of “hasName” property of OpenERP ontology is the union of 170 sets – concepts, each of which corresponds to a data table. Hence, the resulting SQL query is an array of 170 SELECT queries.

The SQL queries, generated by the module for semantic query execution for the last 4-tuplet, are as follows:

(1) SELECT * FROM account_account_template WHERE name='Custom fixture F12'

(2) SELECT * FROM account_account_consol_rel WHERE name='Custom fixture F12'

....

....

(65) SELECT * FROM product_template WHERE name='Custom fixture F12'

....

....

(170) SELECT * FROM wkf_workitem WHERE name='Custom fix-ture F12'

The queries are executed and resulting datasets are transformed into logical statements which are, then, asserted to a temporary model. The query (65) returns the product template description that matches the given criteria. The result-set is then transformed into the logical statements that describe an instance of “product_template” concept and its necessary conditions.

custom-fixture_f12 type product_template
custom-fixture_f12 hasCostMethod 'Average price'
custom-fixture_f12 hasId 1332
custom-fixture_f12 hasMesType 'Measure type'
custom-fixture_f12 hasName 'Custom fixture F12'
custom-fixture_f12 hasProcureMethod 'Make to Order'
Inner-Fixtures type product_category
Inner-Fixtures hasName 'InnerFixtures'
Inner-Fixtures hasId 12
custom-fixture_f12 hasProductCategory Inner-Fixtures

custom-fixture_f12 hasStandardPrice 540.00

custom-fixture_f12 hasSupplyMethod 'Produce'

custom-fixture_f12 hasType 'Product type'

These logical statements are then asserted into a temporary model (stored in the memory space of the semantic querying engine).

It is important to emphasize that a query execution procedure is recursive. The query is expected to extract from the database and to assert all necessary conditions for a given concept. When the result-set includes a field which is at the destination of one-to-many schema relationship, the algorithm raises the occurrence of another concept (not a basic data type) as a necessary condition. In this case, another SQL query is executed to extract the result set which corresponds to this concept. In the above example, for the definition of necessary conditions of “product_template” concept, the instance of the “product_category” concept needs to be constructed and asserted to a temporary model.

In the next iteration of the query execution, the next 4-tuplet is transformed into a set of SQL queries. As it is shown above, value restrictions are transformed to SQL queries in a simple way, where basic data-types (in this case, strings) are used as criteria. In this iteration, the criterion is defined with an instance(s) of the ontology (in this case, bNode2 array). In the example above, only one instance is asserted into local ontology, as a result of the first iteration. Thus, in the second iteration, the following statement is transposed to SQL queries:

bNode1 hasProductTemplate custom-fixture-f12

When existential restrictions are used, SQL WHERE statements are interpreted as the values of the functional data properties of this instance:

custom-fixture_f12 hasId 1332

Given the fact that the domain of “hasProductTemplate” property is a union of three concepts (“product_pricelist_item”, “product_product” and “product_supplierinfo”) in OpenERP local ontology, the following set of SQL queries is generated:

(1) SELECT product_pricelist_item.* FROM prod-uct_pricelist_item, product_template WHERE product_pricelist_item.product_template_id=product_template.id AND product_template.id='1332'

(2) SELECT product_product.* FROM product_ product, product_template WHERE product_ product.product_template_id=product_template.id AND product_template.id='1332'
(3) SELECT product_ supplierinfo.* FROM product_ supplierinfo, product_template WHERE product_ supplierinfo.product_template_id=product_template.id AND product_template.id='1332'
In this example, only the second SELECT query returns a value, because the custom fixture product is engineered to order, so no pricelist or supplier information is relevant for its description. Similarly as in the case of the first iteration, a result set is transformed into a set of logical statements, which describe the instance of "product_product" concept of OpenERP local ontology, by using its necessary conditions:
custom-fixture_f12_p type product_product
custom-fixture_f12_p hasId 67
custom-fixture_f12_p hasProductTemplate custom-fixture_f12
These logical statements are also asserted into the temporary model. In the last iteration, a domain of "hasProductProduct" property is determined for a given range ("custom-fixture_f12_p" instance). Then, the value of the functional property of a criterion instance is used to generate SQL query. This set has 22 SELECT queries because the domain of the "hasProductProduct" property is the union of 22 classes:
(1) SELECT account_analytic_line.* FROM account_analytic_line, product_product WHERE account_analytic_line.product_id=product.id AND prod-uct.id='67'
...
(7) SELECT mrp_production.* FROM mrp_production, product_product WHERE mrp_production.product_id=product.id AND product.id='67'
...
(22) SELECT stock_warehouse_orderpoint.* FROM stock_warehouse_orderpoint, product_product WHERE stock_warehouse_orderpoint.product_id=product.id AND product.id='67'
In contrast to the previous iteration, in this step, the instances of more than one concept of OpenERP local ontology are returned – all instances to which the

custom fixture product is associated (the domain of “hasProductProduct” property), such as account_invoice_line, delivery_carrier, mrp_bom, and others. Then, the result-sets are transformed into logical statements that are asserted to a temporary model. Some relevant statements are:

custom-fixture_f12_prod_sched type mrp_production
custom-fixture_f12_prod_sched hasDatePlanned '2012-02-15 23:59:59'
custom-fixture_f12_prod_sched hasId 67
custom-fixture_f12_prod_sched hasName 'Production schedule for Custom fixture F12'
custom-fixture_f12_prod_sched hasProductProduct custom-fixture_f12_p
custom-fixture_f12_prod_sched hasProductQuantity 3.0
custom-fixture_f12_prod_sched hasDateFinished '2012-02-17 23:59:59'
stock_location_w2 type stock_location
stock_location_w2 hasAllocationMethod ''
stock_location_w2 hasChainedAutoPacking ''
stock_location_w2 hasChainedLocationType ''
stock_location_w2 hasId 8
stock_location_w2 hasName ''
stock_location_w2 hasUsage 'Warehouse 2'
custom-fixture_f12_prod_sched hasStockLocation stock_location_w2

At this time, all instances required for the semantic representation of the query result are stored in the temporary model, in the memory of the inference engine. The second step of the semantic query execution method – query execution and assertions can be considered as completed.

In the third and last step of the method; a semantic DL query is executed on the temporary model, in order to filter only relevant instances. Namely, as it is shown in the description of the third iteration of the query execution step, the property domain inferences may result in some excessive information which is not relevant for the case. Also, in the case where the complex semantic queries (with multiple restrictions on the desired instance) are executed, the intersection of the resulting sets of instances, each corresponding to individual restrictions, need to be inferred. Finally, this filtered model is returned as an end outcome of the semantic query execution. The representation of the outcome of the production schedule querying for the product “Custom fixture F12” is illustrated in Figure 7 (data properties are not displayed).

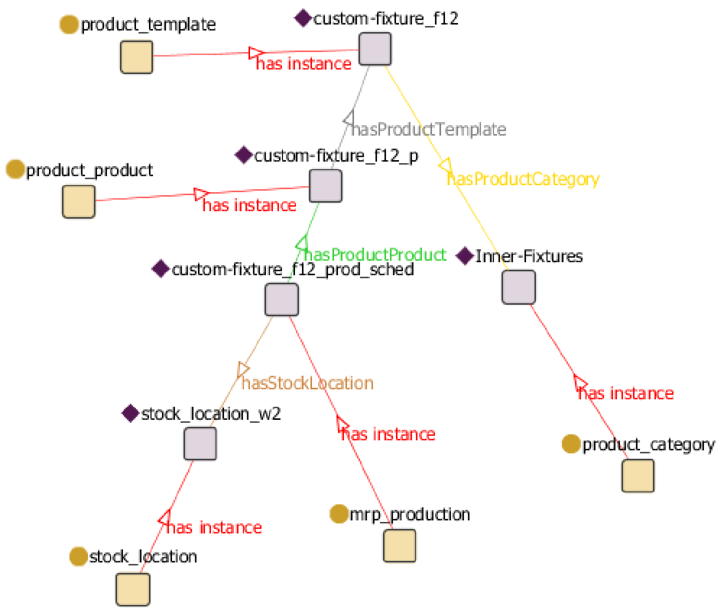


Figure 7: Visual representation of the production schedule for example product “Custom fixture F12”

The resulting graph is a semantic representation of the production schedule concept and is delivered after the semantic query is transformed to a set of SQL queries which are executed in the database of OpenERP system. Now, its concepts and instances can be mapped to the domain models and, hence, more advanced reasoning may be enabled. More importantly, a production schedule concept of OpenERP local ontology may become logically equivalent to the corresponding concepts of local ontologies of other systems. Thus, these systems will become capable of interpreting messages which encapsulate different production schedules. The main implication of this capability in a federated enterprise network is that any two systems may become semantically interoperable [54].

5. Conclusions and future work

The work presented in this paper is a part of the research of semantic interoperability in supply chain networks. In this paper, we focus on introducing the partial realities of the enterprises, that is, data representations of their information systems, into the heterogeneous environment of a supply chain network. In the presented approach, enterprise data models are used to generate local ontologies, by applying a set of rules for interpreting the semantics of an ER

model, namely a database schema. Although “database-to-ontology mapping” is not a novel concept, we showed that existing approaches are characterized by weaknesses, most of which are related to the lack of completeness of properties’ semantics. Our approach and corresponding tools aim at overcoming these weaknesses, thus enabling the complete (in aspects of ER patterns’ semantics and OWL expressivity) interpretation (explication) of the implicit semantics of the ER models, as well as the full correspondence between semantic and database queries. As we have shown, the precondition for this correspondence is the resolution of some technical problems, related to the lack of more expressive formalisms (or technical approaches) for correlating two ontologies and the lack of methods for enabling the management of access rights.
The generated local ontologies should be considered only as intermediary results of the process of conceptualization of one EIS. The main argument for the need of a human intervention is that a weak assumption is made that the ER schema of the EISs represents the semantics of their data models. There are obvious limitations introduced by this assumption, related to semantics coverage and even correctness (because it is more correct to say that ER schemas are conceptual models of the developers’ intents rather than databases of actual systems). However, the case study of generating a local ontology from the OpenERP system shows that the method provides an exhaustive semantic landscape by fully interpreting the semantics of ER underlying schema, by using full OWL/DL expressivity, automatically. As such, this landscape can be improved in the following human intervention by considering business rules, ambiguous types and more sophisticated semantic relations.
In the context of the semantic interoperability, the resulting local ontologies may be considered as enterprise message models. As such, they aim at enabling the semantic interoperability of corresponding enterprise information systems, not the enterprises themselves. Still, significant research efforts are needed for the representation and the exposition of the enterprise business logic, which is hard-coded in the systems, as well as the semantics of the instances, namely, information which are stored in the database. Hence, some of the identified future research topics aimed at improving the resulting conceptual model are: analysis of data patterns with the goal of discovering the semantics of the ambiguous notions of the local ontologies (e.g. type or status); and semi-automatic classification of

the concepts of local ontologies by analyzing necessary conditions for different concepts. There are other topics which relate to improving the application framework that uses the local ontologies, such as: developing a universal method for semantic query rewriting, where source and destination queries use the concepts of two ontologies, logically interrelated by using SWRL rules; and developing a method and tools for execution of "Tell" semantic queries. Finally, developments in some specific areas of Semantic Web tools and languages will certainly contribute to the improved performance and functionality of the application framework for semantic interoperability in supply chain networks. Some expected developments are related to: distributed reasoning capabilities for modular ontologies with dynamic imports, security and access control levels to the parts of ontologies in distributed ontological frameworks, and performance and quality of ontology matching tools.

We consider these research topics as important for increasing collaboration in a supply chain network, as its fulfillment will enable logic driven, automatic and transparent decision making, thus contributing to the transition from traditional supply chains to virtual enterprise and related paradigms.

Acknowledgement

The work presented in this paper was supported by the program for scientific cooperation between Serbia and France PHC PAVLE SAVIC (project No. 23494VF 2010/2011); and the Ministry of Education and Science of the Republic of Serbia (project No. III41017).

References

1. Browne J, Zhang J (1999) Extended and virtual enterprises – similarities and differences. International Journal of Agile Management Systems. 1 (1):30-36

2. Sánchez NG, Apolinar D, Zubiaga G, Atahualpa J, González I, Molina A (2005) Virtual Breeding Environment: A First Approach to Understanding Working and Sharing Principles. In: Proceedings of the 1st International Conference on interoperability of Enterprise Software and Applications. February 23-25, 2005. Geneva. Switzerland.

3. Panetto H, Molina A (2008) Enterprise integration and interoperability in manufacturing systems: Trends and issues. Comput Ind. 59 (7):641–646.

4. ISO/IEC 2382-1:1993. Information technology -- Vocabulary -- Part 1: Fundamental terms, International Organisation for Standardisation, Geneva

5. Chen D, Vernadat F (2004) Standards on enterprise integration and engineering - a state of the art. Int J Comput Integ M. 17 (3):235–253.
6. Obrst L (2003) Ontologies for semantically interoperable systems. In: Proceedings of the 12th International Conference on Information and Knowledge Management. November 3-8, 2003. New Orleans, USA.
7. Lee JL, Madnick SE, Siegel MD (1996) Conceptualizing semantic interoperability: A perspective from the knowledge level. Int J Coop Inf Syst. 5 (4):367-393.
8. Sowa J (2009) Knowledge Representation : Logical, Philosophical, and Computational Foundations. Brooks/Cole Publishing Co., CA
9. The IEEE Standard Upper Ontology web site. http://suo.ieee.org, Accessed: 17.5.2012.
10. Zdravković M, Panetto H, Trajanović M, Aubrey A (2011) An approach for formalising the supply chain operations. Enterprise Information Systems. 5 (4):401-421.
11. Grubic T, Fan IS (2010) Supply chain ontology: Review, analysis and synthesis. Comput Ind. 61 (8): 776-786.
12. Gailly F, Poels G (2011) Experimental Evaluation of an Ontology-Driven Enterprise Modeling Language. In: Advances in conceptual modeling, Recent developments and new directions. Lect Notes Comp Sc. 6999:163-172
13. Hoang HH, Tran PCT, Le TM (2010) State of the Art of Semantic Business Process Management: An Investigation on Approaches for Business-to-Business Integration. In: Intelligent information and database systems. Lect Notes Comput Sc. 5991:154-165
14. Zdravković M, Panetto H, Trajanović M (2011) Local ontologies for semantic interoperability in supply chain networks, In: Zhang, R., Cordeiro, J., Li, X., Zhang, Z., Zhang, J. (Eds.), Proceedings of the 13th International Conference on Enterprise Information Systems. SciTePress, pp. 22-31
15. Lezoche M, Panetto H, Aubry A (2012) Formal Fact-Oriented model transformations for Cooperative Information Systems semantic conceptualization. Enterprise Information Systems. Lecture Notes in Business Information Processing LNBIP 102 117-131
16. Hepp M (2007) Ontologies: State of the art, business potential and grand challenges. In: Hepp, M., De Leenheer, P., de Moor, A. and Sure, Y. (eds), Ontology Management – Semantic Web. Semantic Web Services and Business Applications. Springer, Berlin/Heidelberg, 2007, pp.3-22.
17. Castano S, De Antonellis V (1998) A framework for expressing semantic relationships between multiple information systems for cooperation. Inform Syst. 23 (3-4) (1998) 253-277.
18. Astrova I (2004) Reverse Engineering of Relational Databases to Ontologies. Lect Notes Comput Sc. 3053:327-341.
19. Batini C, Lenzerini M, Navathe SB (1986) A comparative analysis of methodologies for database schema integration. ACM Comput Surv. 18 (4):323-364.
20. Sheth A, Larson J (1990) Federated Database Systems. ACM Comput Surv. 22 (3):183-236.
21. McBrien P, Poulovassilis A (1998) A Formalisation of Semantic Schema Integration. Inform Syst. 23 (5):307-334.
22. Rahm E, Bernstein PA (2001) A survey of approaches to automatic schema matching. VLDB Journal. 10:334–350.

23. Doan A, Halevy AY (2005) Semantic Integration Research in the Database Community: A Brief Survey. AI Magazine. 26 (83)
24. Wache H, Vögele T, Visser U, Stuckenschmidt H, Schuster G, Neumann H, Hübner S (2001) Ontology-based Integration of Information - A Survey of Existing Approaches. In: Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA, 2001, pp. 108-117.
25. William WS, Johannesson P, Bubenko Jr. JA (1996) Semantic similarity relations and computation in schema integration. Data Knowl Eng. 19 (1):65-97.
26. Zhao H, Ram S (2007) Combining schema and instance information for integrating heterogeneous data sources. Data Knowl Eng. 61 (2):281-303.
27. Unal O, Afsarmanesh H (2010) Semi-automated schema integration with SASMINT. Knowl Inf Syst 23 (1): 99-128.
28. Ghawi R, Cullot N (2007) Database-to-Ontology Mapping Generation for Semantic Interoperability. In: Proceedings of 3rd International Workshop on Database Interoperability. September 24, 2007. Vienna. Austria.
29. Cullot N, Ghawi R, Yetongnon K (2007) DB2OWL: A Tool for Automatic Database-to-Ontology mapping. In: Proceedings of the 15th Italian Symposium on Advanced Database Systems. June 17-20, 2007. Torre Canne di Fasano (BR). Italy.
30. de Laborda CP, Conrad S (2005) Relational.OWL: a data and schema representation format based on OWL. In: Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling. January 30 - February 4, 2005. Newcastle. Australia.
31. Konstantinou N, Spanos D, Chalas M, Solidakis E, Mitrou N (2006) VisAVis: An approach to an intermediate layer between ontologies and relational database contents. In: Proceedings of International Workshop on Web Information Systems Modeling. June 6, 2006. Luxembourg.
32. Xu Z, Zhang S, Dong Y (2006) Mapping between Relational Database Schema and OWL Ontology for Deep Annotation. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence. December 18-22, 2006. Hong Kong.
33. Xu Z, Cao X, Dong Y, Su W (2004) Formal Approach and Automated Tool for Translating ER Schemata into OWL Ontologies. Advances in Knowledge Discovery and Data Mining, Lect Notes Comput Sc. 3056:464-475.
34. Rishe N, Furht B, Adjouadi M, Barreto A, Davis D, Wolfson O, Yesha Y, Yesha Y (2011) Semantic Wrapper: Concise Semantic Querying of Legacy Relational Databases. Handbook of data intensive computing. 2:415-444
35. Vavliakis KN, Grollios TK, Mitkas PA (2010) RDOTE - Transforming Relational Databases into Semantic Web Data. ISWC Posters&Demos 2010
36. Sane S S, Shirke A (2009) Generating OWL ontologies from a relational databases for the semantic web. In: Proceedings of ICAC3 '09 International Conference on Advances in Computing, Communication and Control. Pages 157-162, ACM New York, NY, USA
37. Fischer M, Dean M, Joiner G. (2008) Use of OWL and SWRL for Semantic Relational Database Translation. In: Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions. Washington, DC (metro), 1-2 April 2008.

38. Spanos DE, Stavrou P, Mitrou N (2012) Bringing relational databases into the Semantic Web: A survey. Semantic Web. 3(2): 169-209
39. Curino C, Orsi G, Panigati E, Tanca L (2009) Accessing and Documenting Relational Databases through OWL Ontologies. Flexible query answering systems. Lect Notes Comput Sc. 5822:431-442
40. Auer S, Dietzold S, Lehmann J, Hellmann S, Aumueller D (2009) Triplify: light-weight linked data publication from relational databases. In: WWW '09 Proceedings of the 18th international conference on World wide web, Pages 621-630, ACM New York, NY, USA
41. Hassanzadeh O, Kementsietsidis A, Lim L, Miller RJ, Wang M. (2009) A framework for semantic link discovery over relational data. In: CIKM '09 Proceedings of the 18th ACM conference on Information and knowledge management. 1027-1036. ACM New York, NY, USA
42. Vavliakisa KN, Symeonidisa AL, Karagiannisc GT, Mitkasa PA (2011) An integrated framework for enhancing the semantic transformation, editing and querying of relational databases. Expert Systems with Applications. 38 (4): 3844–3856
43. Hert M (2009) Relational Databases as Semantic Web Endpoints. The semantic web: research and applications. Lect Notes Comput Sc. 5554:929-933
44. Koutsomitropoulos DA, Domenech RB, Solomou GD (2011) A Structured Semantic Query Interface for Reasoning-Based Search and Retrieval. The semantic web: research and applications. Lect Notes Comput Sc. 6643:17-31
45. Buil-Aranda C, Corcho O, Krause A (2009) Robust Service-Based Semantic Querying to Distributed Heterogeneous Databases. In: Proceedings of 20th International Workshop on Database and Expert Systems Application, DEXA '09. pp:74-78
46. Motika B, Horrocks I, Sattler U (2009) Bridging the gap between OWL and relational databases. Web Semantics: Science, Services and Agents on the World Wide Web. 7(2):74–89
47. Bizer C, Heath T, Berners-Lee T. (2009) Linked Data – The Story So Far. International Journal for Semantic Web and Information Systems. 5(3):22
48. Guarino N (1997) Understanding, building and using ontologies. Int J Hum-Comput St. 46 (2–3):293–310.
49. Antoniou G, van Harmelen F (2009) Web Ontology Language: OWL. In: Handbook on ontologies. International Handbooks on Information Systems, Part 1: 91-110
50. Horrocks I, Patel-Schneider PF, Boley H, Tabet S, Grosof B, Dean M (2004) SWRL: A Semantic Web Rule Language - Combining OWL and RuleML. W3C Member Submission 21 May 2004. http://http://www.w3.org/Submission/SWRL/
51. Ensan F, Du W (2011) A knowledge encapsulation approach to ontology modularization. Knowl Inf Syst. 26 (2): 249-283.
52. Eshuis R, Grefen P (2008) Constructing customized process views. Data Knowl Eng. 64:419-438.
53. Zdravković M, Trajanović M, Stojković M, Vitković N, Mišić D (2012) A case of using the Semantic Interoperability Framework for custom orthopedic implants manufacturing. Annual Reviews in Control. 36 (2): 318-326.

54. Yahia E, Lezoche M, Aubry A, Panetto H (2012) Semantics enactment for interoperability assessment in enterprise information systems. Annual Reviews in Control 36 (1): 101–117.