Integrating an ontology into a software system

Leila Zemmouchi-Ghomari, Abdessamed Réda Ghomari, Lamia Adjir, Lynda Belaala
University of Science and Technology, Houari Boumediene, Algeria
lzemmouchi-ghomari@usthb.dz, a_ghomari@esi.dz, adjirlamia@gmail.com, belaalalynda92@gmail.com

DOI:10.20470/jsi.v8i2.306

Abstract: Ontologies are recognized as formal knowledge representations facilitating sharing, interoperability, and cooperation between people and heterogeneous systems. However, several questions remain unanswered in relation to ontology usage in real world software systems, such as how to achieve their concrete integration in software systems and to what extent are they really beneficial to these systems? The aim of this paper is twofold. On one hand, we propose a generic approach that integrates an ontology into a software system and on the other hand we demonstrate its feasibility and usefulness by carrying out a real world case study; namely, a competency location system for a computer science school.

Key words: ontology, software system, relational databases, integration

1. Introduction
Ontologies are machine-readable and consensual resources designed to promote sharing, interoperability, and reuse (Noy & McGuinness 2001).

Given that ontologies are expressed in languages derived from description logic, these resources have a potentially evolutionary pattern through reasoners (Abburu 2012), which enable the deduction of new facts via defining axioms and their population with instances. Derived new facts can be verified in terms of consistency (Stuckenschmidt 2008). If the check is positive, they can be added to the database scheme. This enables the continuous enrichment of knowledge bases and is unlike static structures such as relational databases.

The advantages of ontologies are known and theoretically sound with regard to enterprise information systems (Badsi et al. 2015; Oberle 2013); however, there is still ongoing research on integrating them into software systems and evaluating their impact on these systems (Kusy 2013).

This paper proposes a generic approach to the integration of an ontology into a software system. The testing of this integration’s feasibility and usefulness was based on a real world case study.

The remainder of this paper is organized as follows. Section 2 provides a brief description of ontologies. Section 3 presents a few typical situations in the context of ontology integration in software systems. Section 4 describes the proposed integration approach throughout its different phases. Section 5 describes the case study through which the concrete application of the guidelines presented in the previous section was carried out. Section 6 summarizes the contributions of this study and highlights its limitations. Finally, Section 7 concludes the paper and presents insights gained from this study.

2. Ontology description
One of the most consensual ontology definitions is that an ontology is a formal, explicit specification of a shared conceptualization (Gruber 1993), it provides a machine readable, standardized representation of a part of the real world.

Ontology objectives consist in promoting: share, reuse and communication between human and artificial agents (Milton et al. 2010).

Ontologies, as formal representations, enable: automatic interpretation of the domain model, inferences via reasoners to derive new facts and querying via formal query languages.

Ontologies allow, as stated by Oberle (2013, pp. 1), an interesting combination of technology features that are not permitted by other technologies, “There exists a plethora of technologies that offer the
features of conceptual modelling but only ontologies combine this feature with web compliance, formality and reasoning possibilities”.

3. Ontology integration in software systems

Ontology usage in a database context is generally related to integration of heterogeneous databases (Dou & LePendu 2006; Song et al. 2013). Information integration can be defined as combining data from different sources and providing a unified view of these data. In the context of the current work this view can be assimilated to an ontology. In fact, ontologies focus on meaning; they can link applications that use different storage systems. Furthermore, Beydoun (2014) proposed the development of an ontology to formally represent information systems requirements.

Another usage is related to Ontology-based data access (OBDA): In this scenario, an ontology is linked to a source database and plays the role of an intermediate layer between the user and the data (Spanos et al 2012).

However, there are several obstacles that limit broad ontology application in the industry (Kusy 2013), such as: the lack of clear guidelines to develop ontologies, ontology development and integration is cost effective and not well documented, in addition to the lack of suitable ontologies that covers enterprise information systems domain.

4. Proposed integration approach

We assume that there is a software system and that it is in a production phase (functional). The ontology to be integrated has already been selected or developed. The ontology, which we intend to integrate into the system, is not static and has to be continuously enriched from relevant sources (ontology libraries, documents, etc.). Every time new knowledge, related to the scope of the system, is discovered, the ontology must be updated. The objective is to cover the maximum knowledge favorable to the better functioning of the system.

The approach is composed of nine phases (Fig. 1). We explain each phase and provide some tools to perform required tasks automatically or at least semi-automatically whenever possible.

Phase 1: conversion of the database scheme to the relational model. The relational model proposed by Edgar F. Codd in 1970 is a simple data model supporting data independence; it is still the most widely used model¹.

Phase 2: ontology scheme conversion to the relational model based on the principles explained in (Astrova et al. 2007; Vysniauskas & Nemuraite 2009), which are summarized in the following tables (Tab. 1 & Tab. 2):

Tab. 1 Correspondences between ontology elements and UML class diagram elements

<table>
<thead>
<tr>
<th>Ontology Element</th>
<th>UML Class Diagram Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept, Class</td>
<td>Class</td>
</tr>
<tr>
<td>subclass, subtype</td>
<td>Specialization-Generalization</td>
</tr>
<tr>
<td>objectProperty(part-of, has-part)</td>
<td>Aggregation/Composition</td>
</tr>
<tr>
<td>Other objectProperty</td>
<td>Association</td>
</tr>
<tr>
<td>DataTypeProperty</td>
<td>Attribute</td>
</tr>
<tr>
<td>Individual</td>
<td>Instance</td>
</tr>
</tbody>
</table>

Tab. 2 Correspondences between OWL Axioms and UML constraints

<table>
<thead>
<tr>
<th>Ontology Axioms</th>
<th>UML Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllValuesFrom</td>
<td>Specializes</td>
</tr>
<tr>
<td>SomeValuesFrom</td>
<td>Refines</td>
</tr>
<tr>
<td>DisjointWith</td>
<td>Disjoint</td>
</tr>
<tr>
<td>UnionOf</td>
<td>Cover</td>
</tr>
<tr>
<td>Cardinalities</td>
<td>Multiplicity</td>
</tr>
</tbody>
</table>

**Fig. 1 Ontology Integration Approach Scheme**
There exist several techniques for enabling the conversion of ontologies to relational databases (Gali et al. 2004), such as OWL2ToRDB\(^2\); however, researchers agree on the fact that there is a drawback throughout this conversion, which is the loss of expressivity. This is attributed to description logics based on ontology languages in addition to the fact that mappings between the two structural elements are not straightforward (Martinez-Cruz et al. 2012). OWL2ToRDB combines the direct mapping of ontology classes, properties and instances to the database schema with representative axioms and restrictions in meta-tables. The objective of this conversion is to obtain two structures represented by the same formalism in order to enable their fusion (phase 3).

**Phase 3**: Model fusion or merging as a result of the two previous phases. The result is a new and enriched relational model. This phase requires the participation of ontology developers and database designers in order to identify equivalences and overlaps between the elements of the two structures. They have to be aware of synonyms and homonyms in order to avoid the misinterpreting the semantics of the two models.

**Phase 4**: Proposition of new functionalities, based on the new scheme, in order to update the application with more functionalities, such as update, search and visualization of integrated data, and new associations.

**Phase 5**: Implementation and test of the system’s new version. The implementation concerns both the database and the application. Tests are carried out in order to detect if there are bugs or malfunctioning for the purpose of overcoming these failures.

**Phase 6**: Periodically, mapping is carried out between the relational database (RDB) scheme and the ontology scheme in preparation of phase 7. This mapping can be assimilated to intersecting the ontology scheme and the last version of the database scheme. The intention is to highlight common classes and properties in order to populate the ontology only with instances related to these common elements.

**Phase 7**: Periodically, the ontology is fed with RDB instances belonging to the set of elements resulting from the previous phase. Several protégé plugins enable the performance of this task, such as: Data Master\(^3\), RDOTE\(^4\) and Ontop\(^5\).

**Phase 8**: A reasoner is applied to the ontology; hence, new facts are inferred. The reasoner is a software used to derive new facts from existing ontologies (Abdur 2012). Some of the popular reasoners developed in the last few years are: Pellet\(^6\), RACER\(^7\), FACT++\(^8\) and Hermit\(^9\); they are also available as plugins for widely used ontology editors such as protégé. This deduction capability is possible due to description logics, which constitute the underlying basis of ontology languages, and in particular to the web ontology language OWL\(^10\), recommended by W3C since 2009. The most noticeable deduction is the inference of a subsumption hierarchy for the classes described in the ontology.

**Phase 9**: System maintenance. Inferred facts are verified by ontology developers and domain experts using the explanation workbench protégé plugin (Horridge et al. 2009), which allows them to:

- Map between inferred facts and facts at the origin of these inferences
- Detect incoherencies and contradictions
- Provide more information in order to decide whether the ontology is satisfying all stakeholders
- Precisely determine the causes of undesirable facts

---

\(^2\) https://protegewiki.stanford.edu/wiki/OWL2ToRDB
\(^3\) https://protegewiki.stanford.edu/wiki/DataMaster
\(^4\) https://sourceforge.net/projects/rdote/
\(^5\) https://protegewiki.stanford.edu/wiki/Ontop
\(^6\) https://github.com/stardog-union/pellet
\(^7\) http://www.ifis.uni-luebeck.de/~moeller/racer/
\(^8\) http://owl.man.ac.uk/factplusplus/
\(^9\) http://www.hermit-reasoner.com/
\(^10\) https://www.w3.org/TR/owl2-primer/
In the case of positive assessment, the new facts are converted and the relational model of the database is updated. Then the process continues from phase 4. Note that each new fact enriches the database further.

In the case of negative assessment, the correction of the ontology conceptual schema is performed and the process continues from phase 2.

The types and timing of Interactions between the ontology and the database are summarized in Fig. 2.

![Fig 2. Ontology-Database interactions](image-url)

5. Case Study
The real world case study of the current work concerns a competence localization system called “ESI Clever Network” developed at ESI (Ecole Nationale Supérieure d'Informatique) a computer science higher school at Algiers, Algeria.

5.1 Description
According to Trichet (2003, pp. 1), « A competency is the effect of combining and implementing Resources in a specific Context for reaching an Objective ». We use the terms “competency”, “competence” or “skill” interchangeably.

A competency location system is aimed to identify and to organize the areas of knowledge and the expert profiles. In addition, a competence location system contributes to increasing productivity, improving the quality of work and to optimizing personnel management by reducing the impact of mobility.

The objective of the system is to match people profiles with given job descriptions or training sessions. A well-defined common understanding of each competence needs to be developed and enforced across various departments or organizations (Schmidt & Kunzmann 2007).

This system is based on the principle of “assisted declarative”, that is, each participant declares freely the competencies he wants to communicate.

However, the declaration of competences using free text and tags (keywords) presents drawbacks because system users may introduce ambiguities and inconsistencies as long as the vocabulary used is not standardized (Oberle 2013).

Two fundamental problems characterize our competence location system

1. The skill model has to be represented in a way that allows all stakeholders to understand the described skills homogeneously by means of formal representations such as ontologies.

2. The assessment of skills strength has to be approved broadly. For example, measuring programming skills or soft skills, like: leadership, presentation, negotiating techniques and communication, is really challenging (Dorn & Pichlmair 2007).
Promising solutions to these challenges are ontology-based approaches (Biesalski & Abecker 2005; Laukkanen & Helin 2004). So, in order to avoid ambiguities and inconsistencies, An application ontology known as ECAO\(^{11}\) (Esi Competency Application Ontology) has been developed (Zemmouchi-Ghomari et al. 2014) that represents, in a generic and reusable way, semantics of the competence domain and provides a common and consensual understanding to share the knowledge of the domain (Fig. 3). It is expressed in French Language.

![Fig. 3 Key concepts of ECAO Ontology](image)

5.2 Application of the approach

5.2.1 Phase 1: conversion of the RDB to the relational model

There exist approaches that derive automatically relational tables from designed models (Rafe et al. 2011). Most of the transformations rules are straightforward (class-> table, attribute ->column, association -> foreign key, etc.) except for inheritance for which there are three possible ways: (1) Each class hierarchy has a single corresponding table that contains all the inherited attributes for all elements - this table is therefore the union of every class in the hierarchy. (2) Each class in the hierarchy has a corresponding table of only the attributes accessible by that class (including inherited attributes). (3) Each generation in the class hierarchy has a table containing only that generation's actual attributes. We opted for the third strategy since it is easier to maintain.

5.2.2 Phase 2: conversion of the ontology to the relational model

We used Protégé UML Backend\(^{12}\): a Protégé plugin that exports an ontology to a UML Class Diagram, it works only with protégé 3.4, not later versions. Then, we converted the resulted diagram into a relational model as performed in phase 1.

5.2.3 Phase 3: merge of the two relational models

Since the two models have been designed by different stakeholders, it is mandatory that they work jointly to discuss in detail what differentiate and what brings together the two structures concerned by the merge. This collaborative task is not trivial because it implies to find a compromise between the different points of view of the designers. Nevertheless, the benefit of this phase is that the output is a model that covers aspects related to the knowledge domain and hence to a broader scope than the field of study (frontiers of the software system).

We found 10 common relations (from 44), such as: Member relation in the database is equivalent to Expert relation in the ontology, however 20 relations pertain exclusively to the ontology which shows the broader ontology coverage of the domain.

---

\(^{11}\)https://sourceforge.net/projects/competenyapplicationontology

\(^{12}\)https://protegewiki.stanford.edu/wiki/UML_Backend
5.2.4 Phase 4: proposition of new functionalities

Since the database has almost doubled in terms of number of relations, it is obvious that several opportunities for adding features are available. For example:

- Visualize commonalities with regard to other members since profiles’ members have been enriched through the competence ontology.
- Visualize clouds of tags with regard to declared instances of specific competences by the members knowing that competences are organized in a tree structure according to the ontology structure.
- Visualize all members having some targeted competences

More values are available to some attributes thanks to the competence ontology, such as:

- Current professional situation: nothing, worker, Open-ended contract, Looking for a job, student, at the end of the studies, project manager, trainee, and apprentice.
- Availability: nothing, active search, internship search, in single standby, open to opportunities, unavailable.

5.2.5 Phase 5: implementation and test of the new version of the system

The application was implemented in PHP/MySQL in addition to the framework Joomla which enable the usage of the Model View Control (MVC) architecture.

This phase consists in updating the application with new functionalities according to the previous phase, and updating the database in order to add the new relations and attributes.

Some screenshots of the application interface are presented underneath (Fig. 4, Fig. 5 & Fig. 6).

Fig. 4 Competences’ Selection and Rate
5.2.6 Phase 6 and 7: mapping between the two relational models and feed the ontology with RDB instances

These two phases have been performed jointly using Ontop (Bagosi et al. 2014), an open-source Ontology Based Data Access tool. Data is accessed through a conceptual layer expressed in ontology language and the data is stored in a relational database.

Ontology population with RDB instances is performed via 4 steps:

1. Connect protégé to the DBMS (in our case MySQL)
2. Add the mapping composed of an ID, a source (SQL query that retrieves data from the database) and a target (template that shows how to generate OWL Assertions in syntax very close to Turtle). Here is an example of a mapping of the table member to the concept of expert in the ontology: all attributes of member are passed to attributes of the concept expert (Fig. 7).
3. Verification of the result of the mapping: by running a SPARQL query and synchronizing the reasoner Ontop, we obtained mapped data as illustrated by Fig. 8:

4. Import mapped data from the menu of Ontop as illustrated by Fig. 9.

13 https://www.w3.org/TR/sparql11-query/
5.2.7 Phase 8 and 9: Application of the reasoner and Verification of new facts

After applying the reasoner Pellet, several new facts at the instance level (ABox, Assertional Box) and at the scheme level (TBox, Terminological Box) have been added to the ontology.

For example, before applying the reasoner 'C-Sharp' was only an individual of the concept "programming language". Due to the fact that the concept "programming language" is a sub class of Technical competence and technical competence is a sub class of competence, C-Sharp became of type competence (see Fig. 10).

Fig. 10 Inferred facts after reasoner application at the ABox level

Another inferred fact at the schema level is related to the sub property relationship between: “Has Degree”\(^{14}\), sub property of “has Competence”\(^{15}\). The derived fact is that every individual that has as object property “has Degree” is also considered as having a competence: “has Competence”, as illustrated by Fig. 11.

Fig. 11 Inferred facts after reasoner application at the TBox level

\(^{14}\) aDiplome in French

\(^{15}\) aCompetence in French
We are convinced that these new facts can enhance significantly system search and display capabilities.

An ontology can be considered as "inconsistent": if there is no possible interpretation of the Ontology, such as: an instance belonging to two disjoint classes or type errors (Stuckenschmidt 2008). According to this definition, all derived facts have been verified and found consistent.

6. Discussion

The integration approach was designed to be generic and was successfully applied to the case study since it provided the target system with more semantics and functionalities on an ongoing basis; however, this approach has several limitations.

The approach is quite lengthy because it is composed of nine phases, which are mostly manual, even, if tools, which can facilitate these operations, are provided at each stage. Since phases are mostly semi-automatic, the participation of several stakeholders is required, namely, database designers, ontology developers and eventually domain experts.

Note that even if the approach is not trivial, the process has to be executed periodically and that added value is valuable since the database scheme is continuously being enriched with new facts due to the underlying ontology.

Another limitation is related to the selection and/or the development of an adequate ontology for integration into the target software system. Often, for example, available ontologies do not fit perfectly with the specific requirements of corporate-scale software systems. Hence, it is common for ontology developers to adapt and extend these ontologies, which requires additional efforts.

Moreover, the approach should be executed without the disjointed use of different tools in each phase; an application integrating the functionalities required for the completion of the approach would be a much better solution.

7. Conclusion

This paper proposed a generic approach to the integration of an ontology into a software system. Additionally, it discussed a case-study assessment of this integration’s feasibility and usefulness.

However, it is not trivial to evaluate the impact of such an integration precisely, based on a single case study and in a short period of time (one round). As the software system matures, more constraints will be defined (as management rules) in addition to the continuous ontology update and its population with database instances.

More experiments with the proposed approach are required for drawing solid conclusions on the validity of the proposed approach and on measuring the impact of integrating an ontology into a software system. Further efforts are required for providing semi-automatic support to some phases, when the approach is validated.

Yet, we are confident that ontologies have a significant role in enhancing the overall performance of software systems, provided that they are integrated correctly.

References


Horridge, M. Parsia, B. & Sattler, U., 2009: *The OWL Explanation Workbench: A toolkit for working with justifications for entailments in OWL ontologies*. School of Computer Science, The University of Manchester, Manchester, United Kingdom


JEL Classification: C80, M15