Facilitating Ontology Co-evolution with Ontology Instance Migration

Mark Fischer and Juergen Dingle
Queen's University, Kingston, Ontario, Canada

Keywords: Ontology, Knowledge Base, Migration, Instance, Co-evolution, Alignment.

Abstract: An ontology is typically defined in terms of some vocabulary that is defined using other ontologies. When the vocabulary changes, it is important for a dependent ontology to evolve in a consistent manner. Automating the migration of ontologies based on changes to their vocabulary is requisite to proper adoption of ontologies for varying fields of use. Oital is a transformation language capable of automatically migrating dependent ontologies. It helps make ontologies easier to maintain as they evolve.

1 INTRODUCTION

As the information gathered and stored by computers grows, so too do the benefits of being able to formally represent that data. Ontologies are the forerunner technology being used to give information extra semantic meaning. Ontologies help limit complexity and organize information so that it can then be applied to problem solving.

Predicting how an ontology may evolve or change in the future is a non-trivial problem that may be infeasible to solve. Each alteration in an ontology may result in inconsistencies for any ontologies which import the changed ontology's vocabulary (Stojanovic et al., 2002). As ontologies become more ubiquitous, the benefits from being able to effectively and consistently keep dependant ontologies up to date increase as well.

Figure 1 demonstrates the problem and a possible solution. The original ontology evolves to become the updated ontology. Any ontologies $(I_1, I_2, ..., I_n)$ which depend on the original ontology might require a migration. The solution being suggested here is to create a transformation language which will perform the migration automatically.

Currently, this problem is often solved by handcrafting SPARQL-Update statements designed to make the necessary alterations to the dependant ontologies. Working at a different level of abstraction while creating these transformations increases difficulty as the user must be aware of how ontologies are

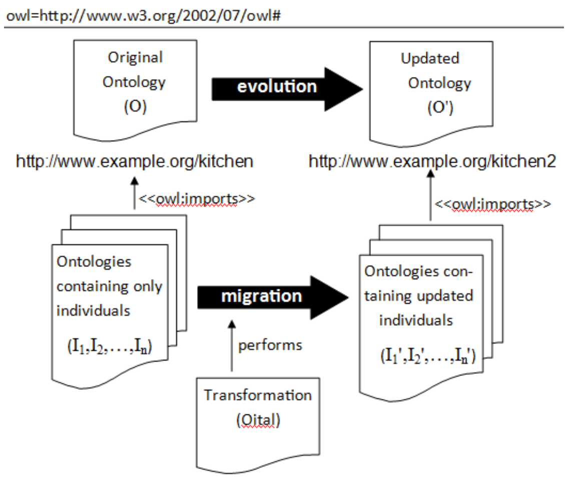
owl=http://www.w3.org/2002/07/owl#
Original Ontology (O)
evolution
Updated Ontology (O')
http://www.example.org/kitchen
http://www.example.org/kitchen2
<<owl:imports>>
<<owl:imports>>
Ontologies containing only individuals
$(I_1,I_2,...,I_n)$
migration
performs
Ontologies containing updated individuals
$(I_1',I_2',...,I_n')$
Transformation (Oital)

Figure 1: An overview of the problem setup.

serialized as RDF. Direct changes to the underlying RDF may result in extra and unwanted artifacts being left behind in the RDF graph.

The main contribution of this approach is twofold. First is the way in which it helps to automate the migration process. This is done by aiding the user in the creation of a transformation which unambiguously defines how to perform a migration for any possible dependent ontology. Second, it distinguishes itself from similar approaches through the creation of a domain specific language at the same level of abstraction as the ontology being modified.

1.1 Terminology

In OWL, the separation of class and properties (terminology) from individuals and relations (assertions) is

This work was partially funded by NSERC (Canada), as part of the NECSIS Automotive Partnership with General Motors, IBM Canada and Malina Software Corp.

expressed using *axioms* and *facts*. In relation to figure 1, I_1 through I_n are all a series of facts with an inclusion reference to O while O and O' both consist of a series of axioms (possibly with inclusion references to other ontologies).

Any given terminology or axiom within an ontology may either restrict to or add to the ontology. Restrictive axioms add further clarification to the information encoded in an ontology (such as *holdsVolume exactly 1* in figure 2) while additive axioms broaden the scope of the information (such as the creation of a new class).

An axiom is defined either in terms of fundamental parts of an ontology (any axiom predefined within OWL, for example) or in terms of already defined axioms within the ontology.

An axiom's context is a set of axioms that is recursively defined. Let $Uses(a)$ be the set of all axioms that a refers to (uses) and let Pre be the set of all predefined axioms in OWL. The context a^* of a is defined to be the smallest set of axioms satisfying:

- $a \in a^*$, and
- $c \in a^*$, if $\exists b \mid (b \in a^* \wedge c \in Uses(b) \wedge c \notin Pre)$

In other words, the context relation is the reflexive, transitive closure of the $Uses$ relation over non-predefined axioms. For the sake of analysis, all predefined axioms are assumed not to change and are therefore not helpful when contrasting evolving ontologies.

What is important about an axiom's context is that it represents the smallest set of auxiliary axioms required in an ontology before the axiom can be considered semantically equivalent with another axiom regardless of the rest of the ontology. If both O and O' contain a^* then we know that the axiom a in O is not only syntactically the same, but also semantically the same as the axiom c in O'.

An inconsistent ontology is one which contains two statements that contradict one another. This paper introduces the concept of an *incongruent* ontology. An ontology is incongruent if it is inconsistent or if it contains facts which can be used to infer axioms within the ontology that are not already present. Since separating axioms from facts is common practice when designing and building ontologies (Gangemi and Presutti, 2009), an incongruent ontology is a sign that this separation of concerns has been breached.

2 METHODOLOGY

Creating the transformation that performs the instance migration can be broken down into three distinct phases which can be reused throughout the development process.

When creating a transformation, it is important to understand how the ontology in question has been changed. This can be done through a change-log created by the individuals updating the ontology, but it is still generally a good idea to have a way in which to compare the original and updated ontology. Therefore the first phase deals with comparing ontologies (e.g., O and O'). There are tools designed to display the differences between ontologies, but since the expected use case is well defined, there is a less general way of comparing which gives results tailored toward the specific problem being addressed.

The second phase is the creation of the transformation itself. It should be as simple as possible both to increase usability and to boost adoption. The language being suggested as a solution here is called Oital. The completion of this phase results in a finished and working transformation. Therefore, phase 2 relies on results from phase 1 and phrase 3. The completed Oital script should be able to be run any number of times to migrate any number of instances from O to O'.

In order to determine if part or all of the transformation that has been created is working as expected, it is important to be able to analyse the transformation. Discovering which parts of the transformation may require more work is key to quickly creating a trustworthy script.

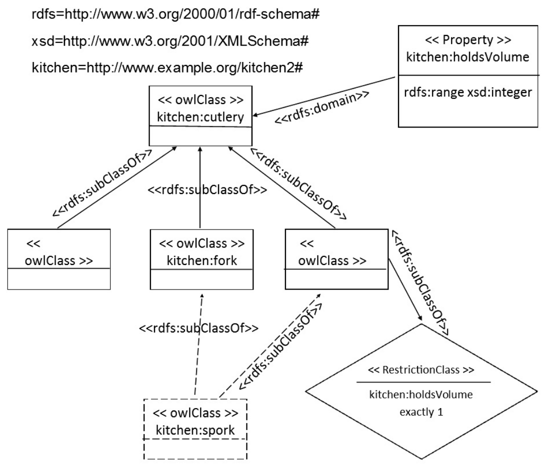

Figure 2: Kitchen example ontology and a possible evolution ontology. Solid components and their labels are from the original ontology while both solid and dashed components are from the evolved ontology.

Table 1: Overview of which axioms require further investigation with respect to specific regions as outlined in Figure 3. Incongruent and semantically ambiguous axioms must be investigated by the user.

	Restrictive Axiom (and its context)	Additive Axiom (and its context)
Region 1	OK	Incongruent
Region 2	OK	OK
Region 3	OK	OK
Region 1 & 2	Ambiguous	Incongruent
Region 2 & 3	Incongruent	Ambiguous

2.1 Phase 1: Comparing the Original and Updated Ontologies

Since the logical formalism used by the Web Ontology Language (OWL) is a description logic, we use the structure of OWL's underlying description logic as a basis for our comparison.

This comparison focuses only on axioms within the ontologies. The goal is to extract only those axioms which may be important for the migration of individuals. Changes which do not affect possible individuals within dependant ontologies can be safely ignored.

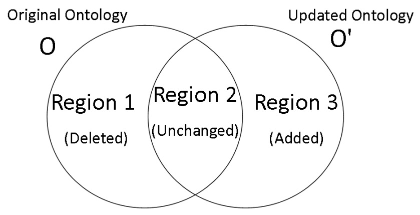

Figure 3: Venn Diagram of two ontologies. Diagram labels the three regions where an axiom may be found.

Figure 3 depicts three regions in which an axiom may exist. Region 1 describes axioms which can be found in O and not in O' (deleted axioms), region 2 describes those axioms which can be found in both O and O' (unchanged axioms), and region 3 describes axioms which cannot be found in O but can be found in O' (new or added axioms).

As shown in table 1, axioms being analysed can be labelled depending on whether they are restrictive or additive as well as which region of figure 3 they belong to. The way in which these labels are applied can be explained using a few categorisations:

1. Any axiom a whose context is entirely in region 3 cannot be the cause of any incongruencies. Region 3 represents those axioms which are entirely unique to the updated ontology. If the axiom (and its context) are new, they will not be related to any individuals within the original ontology. The same rule applies to region 2. For any a such that a^* is entirely in Region 2, a cannot cause incongruencies. Region 2 means the axiom exists in both the updated and original ontology. Those parts of the original ontologies which remain unchanged will continue to describe individuals in dependant ontologies the same way they always have.
2. Axioms in region 1 are labelled depending on whether they are additive or restrictive axioms. If the axiom is restrictive, removing the restriction will not cause any incongruencies. If the axiom is additive, then its removal may be problematic.

   For example, removing the *fork is a subclass of cutlery* axiom from the kitchen ontology in figure 2 results in any facts which state that an individual fork holds a specific volume causing the ontology to become incongruent. *cutlery*.
3. If an axiom and its context a^* exist within more than one region, that axiom is likely significant. If a^* straddles region 1 and region 2, this generally means that the part of the ontology that was removed requires the remaining parts of the ontology to be restructured a bit. If a^* straddles region 2 and region 3, this generally means that part of the new ontology that was newly created affects parts of the ontology that already existed. The addition of a restriction class is one such example. In either case, the axiom may create incongruencies or may be semantically ambiguous in terms of how affected individuals are meant to be migrated.

2.2 Phase 2: Developing Parts of the Transformation

Analysis of the original and updated ontologies does not yield enough information to be able to fully automate the co-evolution process. As will be shown, there are possible evolution paths for an ontology which require human input before a migration can be completed. A transformation is used so that all the necessary extra input can be gathered before the migration process begins.

Comparing the original and updated ontology will only reveal the differences and cannot, inherently, determine the reasoning behind the changes that were made. This meta-data is important for the creation of the transformation and the execution of the migration. It is important to understand the purpose of each change as well as how those changes are meant to affect the individuals within dependant ontologies.

When a new class or property is introduced, how this is dealt with during migration depends on the reasoning behind the introduction to the evolved ontology. If, for example, the new class or property is created from the merger of two deleted classes or properties, then individuals from the deleted elements should be migrated to the new class or given the new property.

Comparing O and O' depicted in figure 2 uncovers a difference for which there is more than one possible desired migration. The evolution in figure 2 shows the creation of a new class called a *spork*. There is more than one possible reason for why this class was created. Each reason requires a different action during migration.

1. The *spork* class' creation was preceded by the creation of a new utensil called a spork. Since the spork is new, none of the cutlery in the original ontology is affected and no change is required during migration.
2. The *spork* class was created because an analysis of the utensil's revealed that many of the forks were holding a volume. Since forks which hold a volume are undesirable, a new class is created for them and volume-holding forks can be migrated to this new class.

Because changes can have multiple migration paths depending on intent, user interaction will be required during the creation of the transformation. To express which sort of migration is desired, a transformation language is used. We developed a domain specific transformation language called Oital.

2.3 Phase 3: Analyzing the Transformation

Once part of the transformation has been created, analysing the transformation can help insure it is correct. Currently, the best way to test an Oital transformation is to use traditional testing methods. Migrating ontologies which are small enough that they can be checked by hand (or verified using some other method) can give the user a good sense of how well the transformation will perform.

It should be possible to run the transformation through a form of abstract interpretation which executes the transformation in an abstract fashion in order to collect specific information (Jones and Nielson, 2001). Such an analysis could, for example, answer questions about how the transformation will affect class membership.

2.4 About Oital

Oital is a transformation language designed specifically for specifying the migration of individuals who are conformant to O such that they become conformant to O'.

Currently, SPARQL-Update (Prud'hommeaux, 2013) is most commonly used to perform tasks which alter an ontology. SPARQL-Update, however, is a query-update language for RDF. Since RDF is used as serialization of the various ontology constructs, it does not inherently have knowledge of what those serializations represent. To use SPARQL-Update to effect change on an ontology requires an understanding of how ontologies are serialized as RDF. Ideally a user should be able to interact with an ontology using the same level of design abstraction as the ontology itself.

Oital works directly with classes and properties in the ontology instead of querying triples in a triple store. This abstracts away from how the ontology is serialized, or stored.

The Manchester OWL syntax was used as an inspiration for Oital's syntax. The hope is that, by using a language similar to one that has been adopted by the World Wide Web Consortium, users will have an easier time learning the new language.

2.5 Incorporating Ontology Alignment Tools

The 2014 Ontology Alignment Evaluation Initiative (OAEI) had 14 participants who competed in various ontology alignment activities(Zlatan, 2014). The work done by some of these tools to create alignment information between ontologies can be leveraged to automatically generate parts of an Oital transformation.

As illustrated in Figure 4, the process of generating Oital code from alignment information discovered by third-party tools can be integrated smoothly into the Oital development process.

An alignment is a set of correspondences which, in turn, can be represented as a 4-Tuple. A correspondence between and entity e_1 (defined in ontology O_1) and an entity e_2 (defined in ontology O_2) can be expressed as a 4-tuple (e_1, e_2, r, n) where r is a semantic relation and n is a real-valued confidence value.

Once alignment information has been extracted from a third-party tool, it can be used to generate some basic Oital code. Consider the following example correspondence which says that the class $HomoSapien$ from the *anik* ontology is equivalent to the class $Human$ from the *mamk* ontology:

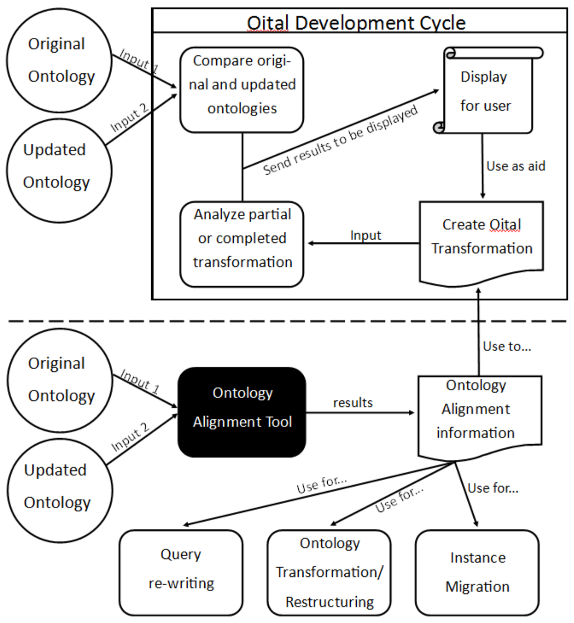

Figure 4: Visual representation of the three phases for creating an Oital transformation. Illustrates how existing ontology alignment tools may be used to generate partial Oital transformations without hindering the usual development cycle of a transformation.

$$(anik:HomoSapien, mamk:Human, \equiv, 1)$$

This correspondence can be used to generate an Oital action taking all instances which are members of anik:HomoSapien and migrating them to mamk:Human. Similar conversions can be done with any other relevant alignment information discovered this way.

While a bit trickier, in theory, multiple third party tools can be used in parallel to increase the likelihood of discovering extra correspondences this way.

3 RELATED WORK

While little work has been done focusing on instance migration in this way specifically, there are technologies already in use which can be used in a similar capacity.

As discussed in section 2.5, ontology alignment tools do some of the work required for instance migration. Furthermore, RDF querying technologies such as SPAQRL and ontology querying technologies such as SeRQL can be re-purposed as well. Since this task is so fundamental to the evolution and development of ontologies, it is important to be able to simplify and streamline instance migration as much as possible.

Ruiz et. al. (Ruiz et al., 2011) researched the importance logic-based semantics of ontologies. Three principles are discussed which should be adhered to when trying to align.

1. Consistency principle: the alignment should not lead to unsatisfiable classes in the integrated ontology
2. Locality principle: correspondences should link entities that have similar neighborhoods
3. Conservativity principle: the alignments should not introduce alterations in the classification of the input ontologies

The notion of an incongruent ontology during migration is an altered approach to ensuring these principles are adhered to.

Fahad et al. (Fahad et al., 2011) propose a way of improving upon state-of-the-art ontology mapping and merging systems by lessening the amount of required human intervention. Merging creates a mapping so that queries used on the updated ontology can return results from the original ontology.

Kondylakis and Plexousakis (Kondylakis and Plexousakis, 2013) have work which tries to sidestep the work of developing correspondences between merging ontologies by altering the queries given to series of evolving ontologies. A mapping is used to extend queries against the ontologies so that the ontologies are considered aligned.

Stojanovic (Stojanovic et al., 2002) proposed a way of facilitating evolution (as well as migration) in a semi-automated manner. Stojanovic (Stojanovic et al., 2002) partially automates the migration process by monitoring an ontology for changes and then taking a user through a series of resolution points for which an evolution strategy can be chosen. The deletion of a class C, for example, may prompt the user to deal with a series of repercussions such as what to do with orphaned sub-concepts of C, what to do with properties of C, what to do with inherited properties of C, what to do with the properties whose range is C, and so on. Instead of dealing with batch analysis of changed ontologies as the approach described in this paper does, Stojanovic (Stojanovic et al., 2002) follows the changes as they happen and provides more context and intent to the changes being performed.

Much of the work done in this field, such as (Stojanovic et al., 2002) relies on being able to access dependent ontologies ($I_1, I_2, ..., I_n$ in Figure 1). Some also depend on user involvement at the time of evolution or migration which our approach does not require.

3.1 Software Model Co-evolution and Database Migration

The migration problem, as presented in this paper, closely resembles the co-evolution problem found in model driven development (MDD) work (Cicchetti et al., 2008). Instead of dependent ontologies, MDD seeks to co-evolve models and meta-models. When a meta-model evolves, it becomes helpful to have tools which help facilitate the migration of the underlying models to the newly evolved meta-model.

The correspondence between data migration due to database-schema evolution and instance migration due to ontology evolution is strong. The many similarities between the two make much of the extensive research done on schema evolution relevant to the work proposed here.

Curino et al. (Curino et al., 2013), Kondylakis and Plexousakis (Kondylakis and Plexousakis, 2013), Fahad et al. (Fahad et al., 2011), and Stojanovic (Stojanovic et al., 2002) all discuss creating mappings between schema, using query rewriting, and keeping track of atomic schema changes. Each aims to make use of these activities to achieve the task of data migration while still allowing legacy queries and without shutting down the affected database where possible.

4 CONCLUSIONS

In this paper, we have described an approach, language, and tool to facilitate the development of transformations that migrate individuals from an original ontology to an updated one.

The approach is based on 1) differencing the ontologies, 2) transformation development using a novel, domain-specific language, and 3) analysis. Finally, ontology alignment tools can be leveraged to aid some of the development of an Oital transformation.

Vital to the efficient use of an ontology is the ability to easily effect change. Ontologies evolve and change as they mature. Tooling designed to facilitate such change is a major step toward increasing the adoption of ontologies.

Current approaches to solving this problem are often laborious, require detailed knowledge about how ontologies are serialized and stored, or are prone to error. Encouraging the users of a vocabulary to migrate their ontologies to an updated version is made significantly easier if that migration is given to them is an easy to use fashion. Oital seeks to provide the facilities to make this task manageable.

This approach has been used to facilitate instance migration for two different case studies. The first took an ontology encoding of the UML 2.1.1 and UML 2.4.1 specifications and migrated some example models. The second was to perform instance migration for different versions of the DBpedia ontologies. DBpedia is an effort to extract structured information from Wikipedia and make it accessible (DBpedia, 2015).

Future work includes improving tooling and approaches for testing and verifying Oital transformations. Integrating ontology alignment tools with the Oital development process to automatically generate parts of the transformation is under development as well.

REFERENCES

Cicchetti, A., Di Ruscio, D., Eramo, R., and Pierantonio, A. (2008). Automating Co-evolution in Model-Driven Engineering. In *EDOC '08,*, pages 222–231.

Curino, C., Hyun, J. M., Deutsch, A., and Zaniolo, C. (2013). Automating the database schema evolution process. *VLDB*, 22(1):73–98.

DBpedia (2015). DBpedia. http://wiki.dbpedia.org/. [Online; accessed 4-May-2015].

Fahad, M., Moalla, N., and Bouras, A. (2011). Towards Ensuring Satisfiability of Merged Ontology. *Procedia Computer Science*, 4:2216–2225.

Gangemi, A. and Presutti, V. (2009). Ontology Design Patterns. *Handbook on Ontologies*, pages 221–243.

Jones, N. and Nielson, F. (2001). Abstract interpretation: A Semantics-based Tool for Program Analysis. *Semantic Modeling, Handbook of Logic in Computer Science*, 4:527–635.

Kondylakis, H. and Plexousakis, D. (2013). Ontology Evolution without Tears. *Web Semantics: Science, Services and Agents on the World Wide Web*, 19:42–58.

Prud'hommeaux, E. (2013). SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/. [Online; accessed 20-Feb-2014].

Ruiz, J. E., Grau, B., Horrocks, I., and Berlana, R. (2011). Logic-based assessment of the compatibility of UMLS ontology sources. *Biomedical Semantics 2*.

Stojanovic, L., Maedche, A., Motik, B., and Stojanovic, N. (2002). User-driven Ontology Evolution Management. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, pages 285–300.

Zlatan, D. e. a. (2014). Results of the ontology alignment evaluation initiative 2014. *Proceedings of the 9th International Workshop on Ontology Matching Collocated with the 13th International Semantic Web Conference*.