Chapter 1
Evaluation of Semi-Automated Ontology Instance Migration

Maxim Davidovsky, Vadim Ermolayev, Wolf-Ekkehard Matzke, and Vyacheslav Tolok

Abstract Ontology instance migration is one of the challenges in knowledge management. It becomes even more complex in distributed settings when, for example, several autonomous agents use partial assertional knowledge in a domain that is formalized by different though semantically overlapping descriptive theories. Agents exchange instances of their ontologies when cooperate. Such an exchange is essentially the migration of the assertional part of an ontology to other ontologies owned by different agents. The paper presents our method and tool support for migrating instances between different semantically overlapping ontologies. The method is based on the use of manually coded formal rules describing the changes between the input and the output ontologies. The tool to support the process is implemented as a plug-in to Cadence ProjectNavigator software. The main contribution of the paper is in presenting the results of the evaluation of this tool. It reports about the set-up for our evaluation experiments, the metrics used for measuring the quality of instance migration, the ontologies that have been chosen as the experimental data, and the evaluation results. Evaluation results are satisfactory and suggest some directions for the future work.

1.1 Introduction

Instance migration is an important phase of ontology development and ontology management activities. Until now a large number of ontologies describing

Maxim Davidovsky · Vadim Ermolayev · Vyacheslav Tolok
Zaporozhye National University, Zaporozhye, Ukraine
e-mail: m.davidovsky@gmail.com,vadim@ermolayev.com,vyacheslav-tolok@yandex.ru

Wolf-Ekkehard Matzke
Cadence Design Systems, GmbH, Feldkirchen, Germany
e-mail: wolf@cadence.com

similar domains from different viewpoints has been developed. Therefore an effective re-use of their assertional knowledge is rational. Ontology instance re-use is also essential in ontology evolution. When a new ontology version is developed it is often necessary to transfer the instances of the previous version(s) to the newer version. The development of a newer ontology version starts with the implementation of the required changes in its TBox. Therefore the reuse of the ABox could be ensured if the elements of the ABox are accordingly transformed. Assertional parts of ontologies can contain a large quantity of instances that in turn makes manual realization of such transformations and instance transfer a laborous task. The paper briefly presents our semi-automated approach to instance migration based on the use of the formal rules describing transformations and transferring instances from a source ontology to a target one. The approach is implemented in our software prototype that has been developed as a plug-in for the Cadence Project Navigator (CPN) Framework for carrying out instance migration between the versions of PSI[1] ontologies [11], [12]. PSI project develops the methodology and the toolset for assessing, predicting, and optimizing the performance of engineering design systems in microelectronics and integrated circuits [10]. A multi-agent system (MAS) for holonic simulation of an engineering design system in the field of microelectronics is developed [20] as a part of the CPN Framework. The MAS assists in analyzing and assessing the performance of a design system as a tool for simulation experiments.

The paper focuses on the evaluation of the approach to instance migration between evolving ontologies in a distributed environment and distributed ontologies with overlapping domains. The scenario of evaluation experiment was developed according to these two use cases. The first phase of the experiment evaluates the quality of instance migration between the versions of PSI ontologies. PSI ontologies are used by cooperating software agents that simulate planning and execution of dynamic engineering design processes [20] in an industrial application domain of Microelectronics and Integrated Circuits. This use case is tailored to support the evolution of the Suite of Ontologies which assertional part is used as the common knowledge of a multi-agent system of cooperative software agents – in distributed settings. The aim of the evaluation based on the second use case is generating reproducible assessment measurements based on the publicly available set of ontologies. For that the benchmark set of ontologies of the Ontology Alignment Evaluation Initiative – (OAEI) 2009 Campaign[2] has been used. The majority of these ontologies are artificially built using the common parent ontology by injecting different sorts of changes. By that a distributed collection of similar ontologies describing the same domain is modeled.

The paper is organized as follows. Section 2 gives a brief overview of the related work. Section 3 describes typical structural change patterns. These

[1] Performance Simulation Initiative (PSI) is an R&D project of Cadence Design Systems GmbH.

[2] `http://oaei.ontologymatching.org/2009/benchmarks/bench.zip`

patterns have been derived from the analysis of the changes in the versions of PSI ontologies and remain valid for the second use case. Change patterns are used in the presented approach for the specification of instance transformation rules. It is explained how instance transformation rules are designed and used for migrating ABox elements. Section 4 presents and discusses the set-up for and the results of the evaluation of the quality of instance migration. Section 5 discusses problems that has been investigated in ontology instance migration and outlines future work.

1.2 Related Work

When an ontology evolves the structural changes are reflected in the controlled vocabulary. A new TBox version is developed by applying these changes. As a rule TBox development is carried out manually using various ontology engineering methodologies and tools ([3], [4], [13]). The next step of the development of the newer ontology version is transferring the instances from the older ABox to the newer one. Doing this manually proves being very laborious. The effort to be spent appreciably increases with the number of transferable entities. Furthermore it can be error prone.

Very similarly, if distributed ontologies are populated based on the assertional parts of the other ontologies in the same domain[3], the patterns apply for transforming the instances according to the structural differences. However in this case manual transformation is not appropriate as it has to happen at run time. Luckily, different distributed variants of a knowledge representation for the domain may be regarded as different versions of the same ontology that differ by the set of structural changes in the TBox.

The main objective of the approach under evaluation is to develop the tool for semi-automated instance migration from an older version to a newer version of the same ontology. The process of ontology instance migration starts with the analysis of the changes in the TBoxes of the ontology versions. Such an analysis is usually made manually, however there are some software tools that can help in this (e.g. [2]). As a result the rules for transforming the instances based on the patterns for the discovered structural changes are written down by the ontology engineer. These rules are coded in an XML-based[4] language and are further used for guiding the automated transfer of the instances from the older ABox to the newer one. It should be mentioned that the process of making the rules is supported by transformation rules editor that is the part of the developed plugin (Fig. 1.1). Similar approach is presented in [21]. As opposed to the presented approach, it is based on the use of the ontology instance migration scenario driving the migration

[3] For example when cooperating software agents communicate their partial knowledge to their peers in a multi-agent system

[4] Extensible Markup Language, `http://www.w3.org/XML/`

process and encoded as Python scripts. A more detailed problem statement for ontology instance migration and a survey of alternative approaches are given in [5].

A good survey of the related work is [19]. In addition, more frameworks for ontology evolution and ontology versioning need to be mentioned ([16], [15])[5] as they provide some extra bits of the required functionality for ontology instance migration.

Ontology instance migration is the problem that has not been fully researched so far. To the best of our knowledge, the literature reporting the results in this sub-field of ontology management is limited. However, looking at a broader knowledge and data change and transformation landscape is helpful. For instance it is useful taking a look at ontology translation and database transformation fields of research ([14], [1], [7]). Ontology translation approaches can be classified as: ontology extension generation; datasets translation and querying through different ontologies. Dataset translation is of particular relevance to our work.

To the best of our knowledge currently there is no a software tool allowing for effectively carrying out instance migration in a fully automated mode. The analysis of the available literature on the tool support reveals that ontology instance migration is often carried out manually, using a tool for defining the differences between the TBoxes of the source and the target ontologies (e.g. PromptDiff [18]).

Finally, the evaluation of the quality of the results of instance migration and, hence, of the efficiency of the used methods is essential. For quality measurements it is possible to adapt the metrics used in data migration [8] and schema matching [6]. These metrics originate from the information retrieval [17] domain. Our approach to evaluation is based on the use of these adapted metrics.

1.3 Instance Transformation Patterns and Rules

For correctly migrating instances between a source and target ontology the transfer process has to be explicitly and formally specified. The specification implies an (implicit) declaration of the set of transferable individuals that is achieved by the explicit indication of the set of classes which instances have to be migrated. Also it is necessary to specify the set of required transformations over the migrating individuals. Having done that, we obtain the set of transformation rules for instance migration.

The process of the creation of a new ontology version starts with applying changes to the TBox. Thus comparing TBoxes of respective ontology versions we can identify a certain set of structural changes conditioning differences

[5] Also the Karlsruhe Ontology and the Semantic Web tool suite, `http://kaon.semanticweb.org/`

between the versions. In the case of ontologies having overlapping domains we first have to determine the correspondences between the entities forming the terminological parts of these ontologies. Then using the obtained mappings we can similarly determine structural changes between the ontologies.

Considering in such a way all possible kinds of changes the set of change patterns that underlie the differences between ABoxes can be defined – in particular such types of changes as the presence or absence of some property, the occurrence of a new relation or the removal of any old relation, etc. Furthermore, based on the set of such change specifications it is possible to define the set of typical transformation operations over the individuals liable to migration. By applying these to concrete ontologies and performing required operations on the instances of the source ontology the target ontology ABox can be obtained.

The analysis of the differences between various versions of PSI ontologies has been done as a part of the presented work. The analysis was carried out as follows. We compared UML diagrams of the ontological contexts [9] of the corresponding concepts and documented the differences. Discovered differences were further cross-checked in the owl[6] files of the ontologies. For realization of the respective changes between the versions the following transformation patterns have been defined: rename a concept; add a property; remove a property; add a relation; remove a relation; change the cardinality of a relation; change an ancestor; (subsumption); and move a concept to another package (ontology module). It should be noted that the operations of moving a concept to another package and subsumption relationship changes do not transform migrated instances. Obviously, referencing of some concept to another package in itself does not entail additional changes and hence will not be reflected in the instances of a concept. Similarly, the fact of referencing some concept as a subclass to another class also does not matter. Probable changes (e.g. inheritance of additional properties) these operations could entail can be described by a combination of the operations and should be discovered when determining differences between ontologies. Therefore the patterns for moving to a different package and changing an ancestor may be not considered. Instance transformation rule patterns can be of two kinds. The first is for the declaration of the set of classes which instances are liable to migration. The second represents typical transformation operations. The patterns are further instantiated in the rule sets for the particular ontologies. These transformation rules are specified by combining appropriate transformation patterns and filling in the pattern slots with concrete values.

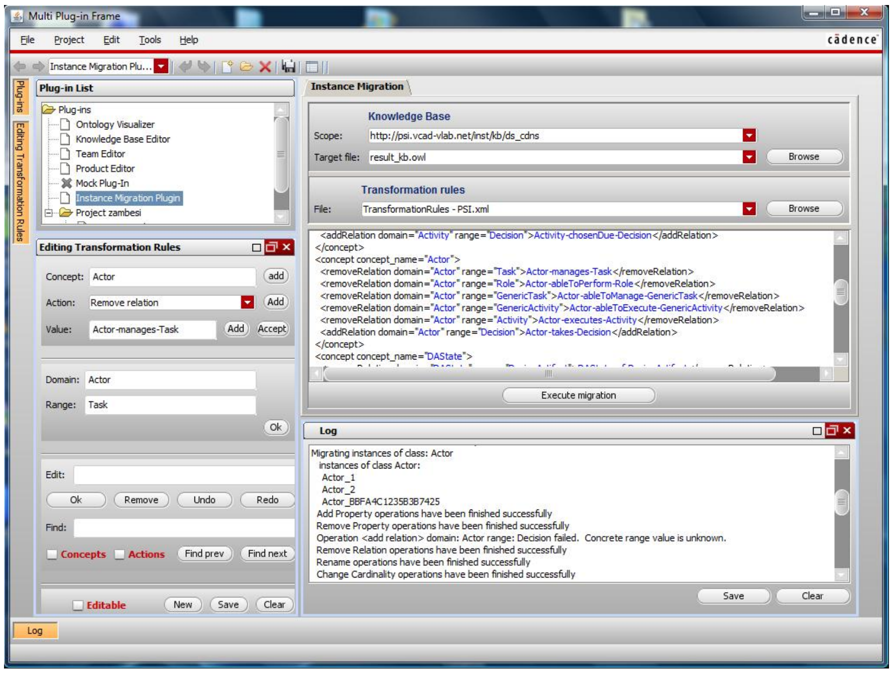
Transformation patterns are defined by means of XMLSchema[7]. The use of the patterns provides decoupling from particular ontology contexts and allows producing a set of specific transformation operations for the specific pair of ontologies. In our approach transformation rules are serialized in XML. It

[6] `http://www.w3.org/2004/OWL/` - Web Ontology Language

[7] `http://www.w3.org/XML/Schema` - XML Schema

allows building convenient and yet computationally efficient formal descriptions for the transformations.

At the step of formulating the transformation rules for a particular migration case the names of the particular classes and the set of transformation operations that should be executed over their individuals are specified in the corresponding xml file. A root element "concepts" in the xml schema file is defined for that. This root element contains the descriptions of the classes and operations for the particular pair of ontologies. It is formalized as a set of "concept" elements with respective properties that contain as their values the names of classes which instances are liable to migration. The tag <concept> contains the embedded tags that correspond to operations which may also have attributes (for example "add property" tag has "data type" attribute for the added property and "value" attribute that contains the value of the property). The following types of transformation operation patterns are defined: "Add relation" (with optional indication of the domain and range of the corresponding object property); "Remove relation" (also with an option of indicating the respective domain and range); "Change cardinality" (with the indication of a property which cardinality should be changed), "Add property" (with the indication of the data type and value of the added property), "Remove property" and "Rename". The software prototype provides the transformation rule editor (Fig. 1.1).



```
  <addRelation domain="Activity" range="Decision">Activity-chosenDue-Decision</addRelation>
</concept>
<concept concept_name="Actor">
  <removeRelation domain="Actor" range="Task">Actor-manages-Task</removeRelation>
  <removeRelation domain="Actor" range="Role">Actor-ableToPerform-Role</removeRelation>
  <removeRelation domain="Actor" range="GenericTask">Actor-ableToManage-GenericTask</removeRelation>
  <removeRelation domain="Actor" range="GenericActivity">Actor-ableToExecute-GenericActivity</removeRelation>
  <removeRelation domain="Actor" range="Activity">Actor-executes-Activity</removeRelation>
  <addRelation domain="Actor" range="Decision">Actor-takes-Decision</addRelation>
</concept>
<concept concept_name="DAState">
```

```
Migrating instances of class: Actor
  instances of class Actor:
   Actor_1
   Actor_2
   Actor_BBFA4C1235B3B7425
 Add Property operations have been finished successfully
 Remove Property operations have been finished successfully
 Operation <add relation> domain: Actor range: Decision failed.  Concrete range value is unknown.
 Remove Relation operations have been finished successfully
 Rename operations have been finished successfully
 Change Cardinality operations have been finished successfully
```

Fig. 1.1 Instance Migration Plugin user interface.

Let us assume for an example that we do the migration of instances of the concept “Actor” (PSI Actor ontology versions). First, we locate its context in the documentation of changes and identify the following changes in comparison to the previous version of the ontology: (i) the relations with “Task”, “Role”, “GenericTask”, “GenericActivity” and “Activity” concepts are removed; and (ii) the relation with concept “Decision” is added. Hence, the following operations on the instances have to be performed: “Remove a relation” and “Add a relation”. Therefore these operations are created in the editor and written down to the transformation rules file as follows (see also Fig. 1.1):

```
<concept concept\_name="Actor">
  <removeRelation domain="Actor" range="Task">
    Actor-manages-Task</removeRelation>
  <removeRelation domain="Actor" range="Role">
    Actor-ableToPerform-Role</removeRelation>
  <removeRelation domain="Actor" range="GenericTask">
    Actor-ableToManage-GenericTask</removeRelation>
  <removeRelation domain="Actor" range="GenericActivity">
    Actor-ableToExecute-GenericActivity</removeRelation>
  <removeRelation domain="Actor" range="Activity">
    Actor-executes-Activity</removeRelation>
  <addRelation domain="Actor" range="Decision">
    Actor-takes-Decision</addRelation>
</concept>
```

1.4 Evaluation

The implemented software prototype has been evaluated with respect to the quality and completeness of the executed instance migration. Schematically the experimental set-up is represented in Fig. 1.2). The results of the migration have been compared to the available target ABoxes for quality and completeness measurement.

The experiment has been run in two phases. Within the first phase the evaluation has been done using the excerpt of the PSI Suite of Ontologies v.2.0 [11] and v.2.2 [12]. The objective of this phase was revealing possible errors at migration run time. The objective of the second phase was to obtain statistically representative and reproducible results. Therefore a broader set of publicly available test case ontologies has been used. The bigger quantity of used ontologies allowed receiving statistically more precise evaluation measurements using various metrics.

Precision and *Recall* metrics have been adopted from Information Retrieval [17] and further adapted for measuring the quality and completeness of ontology instance migration. In our case *Precision (P)* is the fraction of mi-

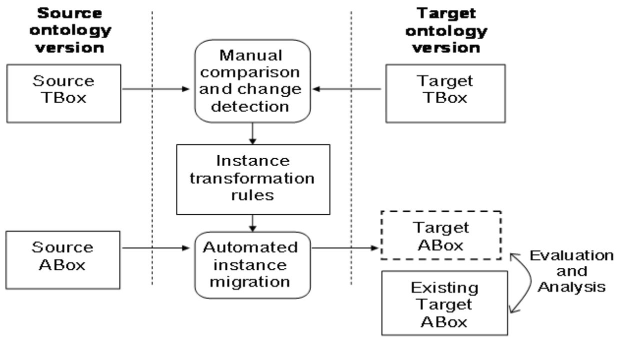


Fig. 1.2 The set-up of the evaluation experiments

grated individuals that are relevant. *Recall(R)* is the fraction of relevant individuals that are migrated. Let's examine the respective contingency Table 1.1. In the terms of this contingency table $P = tp/(tp+fp)$, $R = tp/(tp+fn)$. The effectiveness of the migration tool can also be measured using the Accuracy metric. In our case *Accuracy (A)* is defined as $A = (tp+tn)/(tp+fp+fn+tn)$. An ideal migration outcome is when *Precision*=*Recall*=1. However neither

Table 1.1 Instance migration contingency table.

	relevant	nonrelevant
migrated	true positives (tp)	false positives (fp)
not migrated	false negatives (fn)	true negatives (tn)

Precision nor *Recall* separately does not provide a complete picture of the correctness of the obtained results. For that the *F measure* could be of interest as it brings *Precision* into correlation with *Recall* as a weighted harmonic mean of both: $F = 1/(\alpha(1/R) + (1-\alpha)(1/R)) = ((\beta^2+1)PR/(\beta^2 P + R))$, where $\beta^2 = (1-\alpha)/\alpha$, $\alpha \in [0,1]$. If both precision and recall are of equal importance they can be equally weighted in F by having $\alpha = 1/2$ or$\beta = 1$. This case is the one of a *balanced F measure* $F_{\beta=1} = 2PR/(P+R)$.

Within the first evaluation phase (the excerpt of the PSI Suite of Ontologies) the total number of instances was 1890. The source for the second phase of the evaluation was the set of the test ontologies of the OAEI 2009. The choice was motivated by the public accessibility of the test set that allows cross-evaluation. 40 ontologies have been chosen which in our opinion were the most appropriable for the instance migration evaluation. The total number of instances was 2060. The contingencies for both evaluation phases are given in Table 1.2 and the results of the evaluation experiments are summarized in Table 1.3.

Table 1.2 The contingency table for different ontology sets.

		Relevance	
Test case		relevant	nonrelevant
PSI ontologies	migrated	360	2
	not migrated	48	1480
OAEI ontologies	migrated	1475	7
	not migrated	309	269

Table 1.3 The summary of the results of the evaluation experiments.

	Measures			
Test case	Precision	Recall	Accuracy	Balanced F-measure
PSI ontologies	0.994475138	0.881632653	0.973373303	0.93466032
OAEI ontologies	0.995276653	0.826793722	0.846601942	0.90324556

1.5 Discussion, Conclusions and Future Work

A number of problems caused by various reasons have been faced in the reported research and implementation work that could be generally described as the lack of the necessary information for transforming all the relevant instances in a correct way. Instance migration problem is therefore substantially under-defined.

The provision of an explicit, correct and complete mapping of a source TBox to a target TBox is inevitably not sufficient for devising the complete and correct transformation of the corresponding assertional parts. A characteristic example is the addition of a new relation to a concept in a target ontology. The values of the respective object property shall be added to the instances of the respective class. However we can not know the exact set of individuals that have to be related. Therefore such a property is not added to all instances in our approach.

A similar collision arises when the cardinality of a relation is changed. We cannot know which particular instances have to be related to the individual having the property with the changed cardinality. However it is known and could be specified in advance that the set of particular instances of a given class has to have the relation to another specific individual. Our approach allows specifying such an a priori knowledge explicitly in the transformation rules. Moreover, at migration run time the log of the migration problems is collected and recorded.

Another sort of migration problems arises because not all required OWL constructs are considered at proper time. A good example case is when the two classes (A and B) in the source ontology are defined as equivalent but they are not equivalent in the target ontology. Based on the equivalence, if the instances of A have been migrated then the instances of B have to be migrated as well. However, non-equivalence in the target ontology may cause collisions at particular individuals. Our approach assumes that all collisions of these sorts have to be explicitly resolved when the changes are discovered and the transformation rules are specified.

Despite that our evaluation experiments proved that the software prototype performs migration correctly and completely if the problems mentioned above are absent or the corresponding collisions are correctly resolved in the transformation rules.

Yet another kind of problems is caused by the imperfection of the transformation procedure, especially at manual steps. If a mistake is made at the step of structural change discovery it will propagate to the transformation rules and further to the automated migration. Moreover, the evaluation metrics for correctness and completeness of the automated migration done by the software are not fully accurate. Fortunately, they can only lower the measures. Our experiments show that even if there were mistakes at manual steps, the results are still quite good. More to mention, the chosen metrics are not invariant to the nature of the test data. For example, *Accuracy* increases if the number of the individuals liable to migration is considerably less than the total number of instances which however does not testify the accuracy of the method. *Precision* and *Recall* do not provide exact reflection on migration quality. High *Precision* can be obtained at the expense of *Recall* by migrating only proper individuals but minimizing the number of migrated instances. Similarly, high *Recall* can be achieved by transferring all relevant instances that inevitably minimize *Precision.*

In a summary it may be stated that the software prototype carries out instance migration correctly on the test cases used in the evaluation experiments and demonstrates sufficiently good results in terms of correctness and completeness. Simplicity, transparency, and validity may be mentioned as the highlights of the developed approach. The method and the prototype software tool allow conveniently creating, editing, and processing instance migration rules. It also reduces the probability of errors at automated processing steps, which is proven by the high value of *Precision* measurements. The lowlights are insufficient flexibility and strong correlation with the correctness of manual specification of transformation rules. The lowlights can be mitigated by offering a refined tool support for manual steps. For instance providing the means for semi-automated structural change detection in the pair of ontologies may lower the chance for manual mistakes at the preparatory steps. Subsequently, automated generation of transformation rules will increase the quality of the proposed approach. Both refinements of the current release of the software prototype are planned for the future work.

Finally it has to be stated that, given the limits of the currently available technologies, it is impossible to completely automate the whole process of ontology instance migration. Therefore, the reduction of the proportion of the manual work is the only feasible way of increasing the performance and quality of results in this important activity of ontology engineering. This observation becomes even more valid in distributed settings. The development of a collaborative platform for ontology engineering teams working on distributed ontologies is one more planned activity for the future work.

References

1. Chalupsky, H.: Ontomorph: A translation system for symbolic logic. In: In Proc. Int'l. Con. on Principles of Knowledge Representation and Reasoning, pp. 471–482 (2000)
2. Copylov, A., Ermolayev, V., et al: Ontology revision visual analysis. Tech. rep., VCAD EMEA Cadence Design Systems, GmbH (2009)
3. Corcho, O., et al: Methodologies, tools and languages for building ontologies: where is their meeting point? Data & Knowledge Engineering **46**(1), 41–64 (2003)
4. Corcho, O., et al: Ontological Engineering: Principles, Methods, Tools and Languages. Ontologies for Software Engineering and Software Technology. Springer Berlin Heidelberg (2006)
5. Davidovsky, M., Ermolayev, V.: Ontology instance migration. psi. Tech. rep., VCAD EMEA Cadence Design Systems, GmbH (2009)
6. Do, H., et al: Comparison of schema matching evaluations. LNCS **2593** (2009)
7. Dou, D., et al: Ontology translation on the semantic web. The Journal on Data Semantics pp. 35–57 (2005)
8. Drumm, C., et al: Quickmig - automatic schema matching for data migration projects. In: CIKM (2007)
9. Ermolayev, V., et al: A strategy for automated meaning negotiation in distributed information retrieval. In: 4th Int. Semantic Web Conference ISWC'05 (2005)
10. Ermolayev, V., et al: An upper level ontological model for engineering design performance domain. In: ER '08: Proc. of the 27th International Conf. on Conceptual Modeling, pp. 98–113. Springer-Verlag, Berlin, Heidelberg (2008)
11. Ermolayev, V., et al: Performance simulation initiative. the suite of ontologies v.2.0. reference specification. Tech. rep., VCAD EMEA Cadence Design Systems, GmbH (2009)
12. Ermolayev, V., et al: Performance simulation initiative. the suite of ontologies v.2.2. reference specification. Tech. rep., VCAD EMEA Cadence Design Systems, GmbH (2009)
13. Fernandez-Lopez, M., et al: A survey on methodologies for developing, maintaining, evaluating and reengineering ontologies. Tech. Rep. D1.4, OntoWeb Project (2002)
14. Gruber, T.: Ontolingua: A translation approach to providing portable ontology specifcations. Knowledge Acquisition **5**, 199–220 (1993)
15. Klein, M., et al: Ontology versioning and change detection on the web. In: EKAW-2002, pp. 197–212 (2002)
16. Maedche, A., et al: Managing multiple and distributed ontologies on the semanticweb. VLDB **12**, 286–302 (2003)
17. Manning ChD, e.a.: Introduction to Information Retrieval. Cambridge University Press, NY (2008)
18. Noy, N.F., Musen, M.: Promptdiff: A fixed-point algorithm for comparing ontology versions. In: AAAI/IAAI, pp. 744–750 (2002)
19. Serafini, L., Tamilin, A.: Instance migration in heterogeneous ontology environments. LNCS **4825** (2007)
20. Sohnius, R., Jentzsch, E., Matzke, W.: Holonic simulation of a design system for performance analysis. In: HoloMAS '07: Proc. of the 3rd international conference on Industrial Applications of Holonic and Multi-Agent Systems, pp. 447–454. Springer-Verlag, Berlin, Heidelberg (2007)
21. Vladimirov, V., Ermolayev, V., et al: Automated instance migration between evolving ontologies. Tech. rep., VCAD EMEA Cadence Design Systems, GmbH (2007)