Curating the Specificity of Metadata while World Models Evolve

Yannis Tzitzikas and Anastasia Analyti and Mary Kampouraki
Institute of Computer Science, FORTH-ICS, and
Computer Science Department, University of Crete, Greece
{tzitzik|analyti|mkabour}@ics.forth.gr

ABSTRACT

The main digital preservation strategies are based on metadata and in many cases Semantic Web languages, like RDF/S, are used for expressing them. However RDF/S schemas or ontologies are not static, but evolve. This evolution usually happens independently of the "metadata" (ontological instance descriptions) which are stored in the various Metadata Repositories (MRs) or Knowledge Bases (KBs). Nevertheless, it is a common practice for a MR/KB to periodically update its ontologies to their latest versions by "migrating" the available instance descriptions to the latest ontology versions. Such migrations incur gaps regarding the specificity of the migrated metadata, i.e. inability to distinguish those descriptions that should be reexamined (for possible specialization as consequence of the migration) from those for which no reexamination is justified. Consequently, there is a need for principles, techniques, and tools for managing the uncertainty incurred by such migrations, specifically techniques for (a) identifying automatically the descriptions that are candidate for specialization, (b) computing, ranking and recommending possible specializations, and (c) flexible interactive techniques for updating the available descriptions (and their candidate specializations), after the user (curator of the repository) accepts/rejects such recommendations. This problem is especially important for curated knowledge bases which have increased quality requirements (as in e-Science). In this paper we elaborate on this problem, we propose a general approach, and discuss examples and a prototype application that we have developed assuming the RFD/S framework.

1. INTRODUCTION

The main (if not all) digital preservation approaches (e.g. the OAIS-based) heavily rely on the existence and curation of metadata, and currently Semantic Web languages, like RDF/S, are increasingly used for expressing them (e.g. see [9, 8]). However ontologies change for various reasons, e.g. an ontology may need to change because it offers a richer conceptualization of the problem domain, the domain of interest has been changed, the perspective under which the domain is viewed has changed, or the user/application needs have changed, and so on.

An important observation is that this evolution happens *independently* of the ontological instance descriptions which are stored in the various Metadata Repositories (MRs) or Knowledge Bases (KBs). With the term *ontological instance description*, (for short "metadata") we refer to RDF/S [3] descriptions that classify an instance o to a class c or relate two instances o, o' with a property pr. With the term MR or KB, we refer to a stored corpus of ontological instance descriptions, which can be stored in files, in RDF/S databases (i.e. RDF triple-stores [10]), or in the rapidly growing *Linked Open Data* (LOD) cloud [2]. Due to the distributed nature of the Web and the Semantic Web, the evolution of ontologies happens independently of the ontological instance descriptions, e.g. this is the case with ontologies maintained by standardization authorities. However, it is a common practice (mainly for interoperability purposes) for a KB to periodically update its ontologies to their latest versions by "migrating" the stored instance descriptions to the latest ontology versions. This is actually inevitable since scientific terminology and vocabularies constantly evolve. Such migrations are usually not difficult (i.e. can be performed automatically without need for human intervention), because newer versions are mainly (or constructed to be) compatible with past ones. Nevertheless, they incur gaps regarding the specificity of the migrated instance descriptions, i.e. inability to distinguish those that should be reexamined (for possible specialization as consequence of the migration) from those for which no reexamination is justified. It follows that quality control is very laborious and error-prone. In this paper we introduce an approach for alleviating this problem.

Consider a corpus of instance descriptions and suppose that at certain points in time we can assert, that the available instance descriptions are the *most specific and detailed descriptions* that are possible with respect to the employed ontology. In other words, our metadata are at a good state. For instance, we can make such an assumption after explicit human (e.g. by the curator of the KB) inspection and verification [4], or in cases where the descriptions have been produced automatically by a method that is guaranteed to produce specific descriptions (e.g. by transforming curated relational data to RDF/S descriptions [14], or by automatic classification to categories each defined by sufficient and necessary conditions, etc.). We will hereafter refer to this as-

sumption by the name *maximum specificity assumption* (for short *MSA*). It is not hard to see that if the new version of the ontology is *richer* than the past one, then the corpus of the migrated instance descriptions *may no longer satisfy the MSA with respect to the new ontology.*

The ability to identify the instance descriptions that satisfy the *MSA* and those that do not, is useful in order to address questions of the form: (a) for what descriptions can we make the *MSA*? (b) what (class or property) instances should probably be reclassified (to more refined classes or properties), and (c) which are the candidate new classes or properties (refinements) of such instances? The above questions are very useful for curating a corpus of instance descriptions, i.e. for managing its specificity as the corpus (and its ontologies) evolves over time. Without special support, such tasks would be unacceptably expensive and vulnerable to omissions, for large datasets.

The problem occurs in various domains, including Digital Libraries (e.g. as the *Library of Congress Subject Headings LCSH* evolves), in Biomedicine/Bioinformatics (*Gene Ontology*), in e-Government (*oeGOV Ontologies*), etc. Figure 1 sketches some small and indicative examples of ontology evolution. Our work can aid the curation of structured knowledge, i.e. of digital content that is structured according to a structurally object-oriented model, like RDF/S. For instance, the datasets published in LOD fall into this category. For other kinds of content (e.g. documents, audiovisual material, etc), our work can aid the curation of their metadata. For instance consider the *Dublin Core*[1] metadata schema. In many of its elements (attributes) it is suggested to use values coming from controlled (but evolving over time) vocabularies. For instance, this is the case for the attributes *subject* (for describing the topic of the resource), *language* (where it is recommended to use a controlled vocabulary), *coverage* (for describing the spatial or temporal topic of the resource), and *format* (where the use of MIME types are suggested). Furthermore, various *subproperties* for the metadata element *relation* have been proposed in various contexts[2]. As another case, consider annotations/tags of images (e.g. medical images) or entire datasets using elements from an evolving (e.g. medical) ontology, or provenance metadata (e.g. provenance trails of 3D models) that involve artifacts (e.g. photos) and actors (e.g. photo cameras) identified by URIs and described by various metadata from evolving ontologies. Also note that CIDOC CRM which is an ISO standard for the cultural domain, consists of 86 classes and 137 properties, while its extension for digital objects, CRMdig [15], currently contains 31 classes and 70 properties. In general we can say that RDF/S is currently the "lingua franca" for metadata representation and exchange, and this is the reason why in this work we use it as representation framework. Furthermore our work could be used in cases where the *information object* of an information carrier (of any kind), as described in [6], is expressed using RDF/S.

We will explain the main idea of our approach using a small example.

EXAMPLE 1. *Consider an e-commerce portal that sells various kinds of products, and suppose the metadata that are shown in the left part of Figure 2. Suppose a car* `c1` *that has been classified under the class* `Car`*, and a person* `p1` *that has been classified under the class* `Person`*, defined in an ontology* `Ont1`*, and suppose that both classes have no subclasses. Assume that for the current set of instance descriptions according to* `Ont1` *the MSA holds (i.e. they are complete with respect to specificity). We can infer, from this knowledge, that* `c1` is not *a* `Person` *and* `p1` is not *a* `Car`*. Let* `Ont2` *be a new version of that ontology which, among other, defines the subclasses of the classes* `Car` *and* `Person`*, shown at Figure 2 (right). All subclasses of* `Car` *are* possible classes *for* `c1`*.* `Adult` is not *a possible class for* `c1`*, since* `c1` *was not a person according to* `Ont1`*. Analogously, none of the subclasses of* `Car` *is a possible class for* `p1`*, since* `p1` *was not a car according to* `Ont1`*. Moreover, notice that* `Ont1` *defines a property* `owns` *and suppose that* (`p1` `owns` `c1`) *is an instance description. Also notice that* `Ont2` *defines a subproperty* `sells` *of* `owns` *between* `Person` *and* `Car`*. This property will be prompted as a* possible specialization of the association *between* `p1` *and* `c1`*.*

The computation of possible refinements in the general case can be complex since we can have conflicts among (a) new positive knowledge inferable from the instance descriptions and the new schema, (b) new "negative" information inferable from the past negative instance descriptions and the new schema, and (c) the previously computed possible instance descriptions (possible refinements). In fact, our approach resolves such conflicts by considering that (a) has higher priority than (b), and (b) has higher priority than (c). In addition, it should be possible to update correctly the set of possibilities, at scenarios with several *successive* instance migrations interwoven with several (positive or negative) user feedbacks. Finally, another challenge is to reduce the information that has to be kept to support this scenario, specifically to avoid having to keep negative information of any kind, and to devise compact representations for the possibilities.

We could say that from a more general perspective, our work contributes in enriching the lifecycle of Semantic Web data with quality management, appropriate for scenarios where ontologies evolve frequently and independently from instance descriptions. As a consequence, this allows adopting iterative and agile ontology modeling approaches, appropriate for open environments like Linked Open Data. Note that though there are several works and approaches for dealing with the validity of data during migration in the context of RDF/S (e.g. [11, 7, 13]), there is no work for managing their specificity and quality while ontologies evolve.

The rest of this paper is organized as follows. Section 2 proposes a process model for managing the specificity of metadata, and discusses (mainly through examples) the principles of our approach. Section 3 describes the prototype application that we have developed which is publicly available. Finally, Section 4 concludes the paper and identifies issues for further research.
A thorough elaboration of the problem (that includes formal definitions, algorithms, complexity and experimental results) is available in a technical report submitted for journal publication.

[1] http://dublincore.org/documents/dces/
[2] E.g. at EDM (Europeana Data Model).

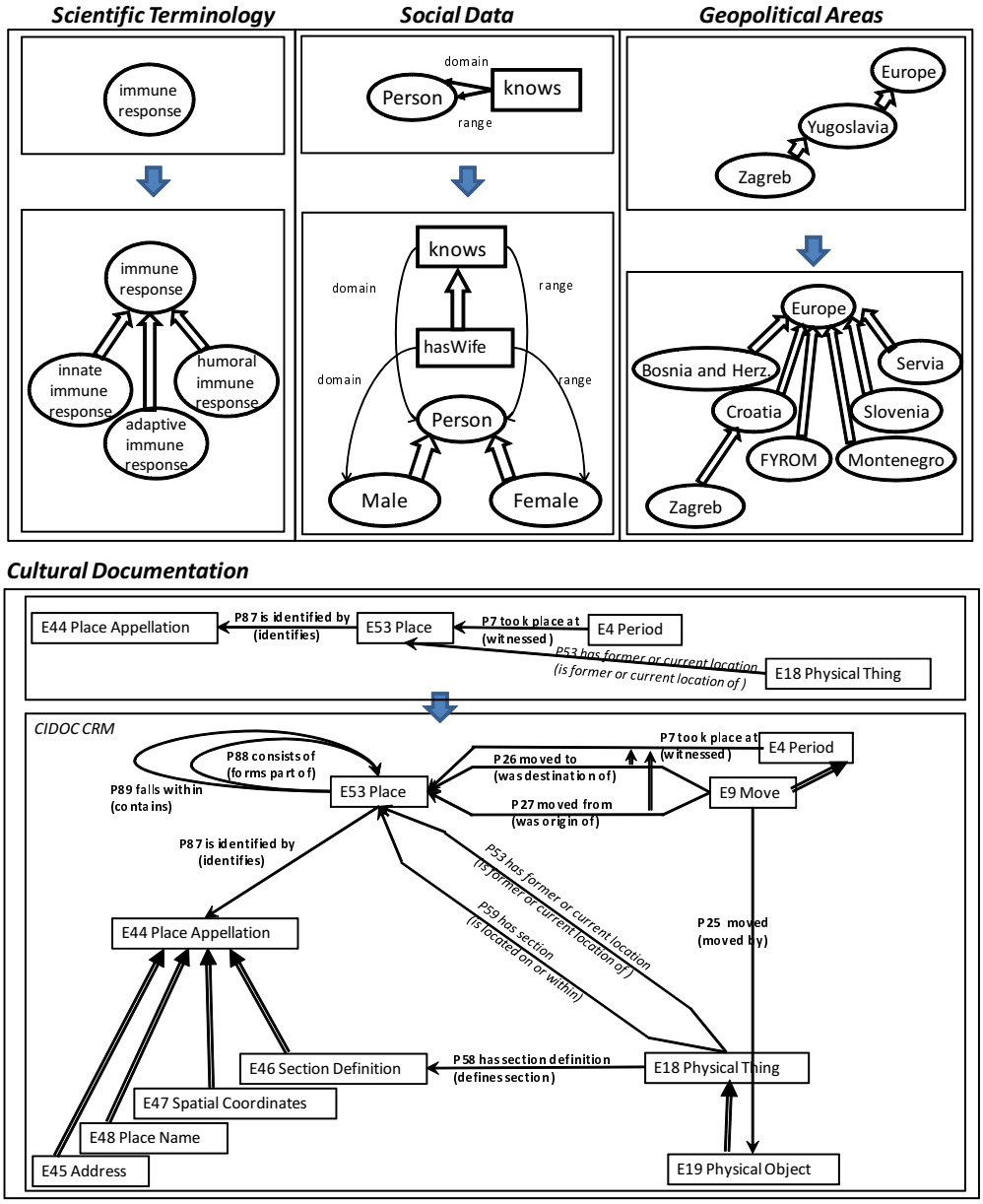

Figure 1: Examples of ontology evolution from various application domains

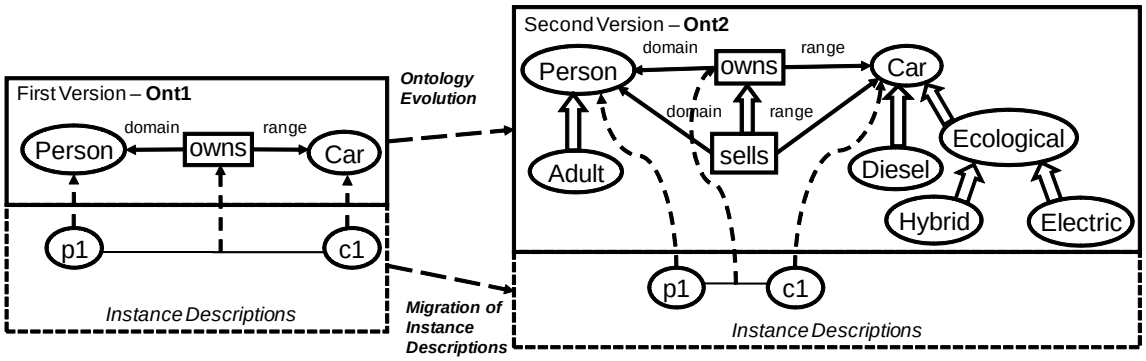

Figure 2: Introductory example

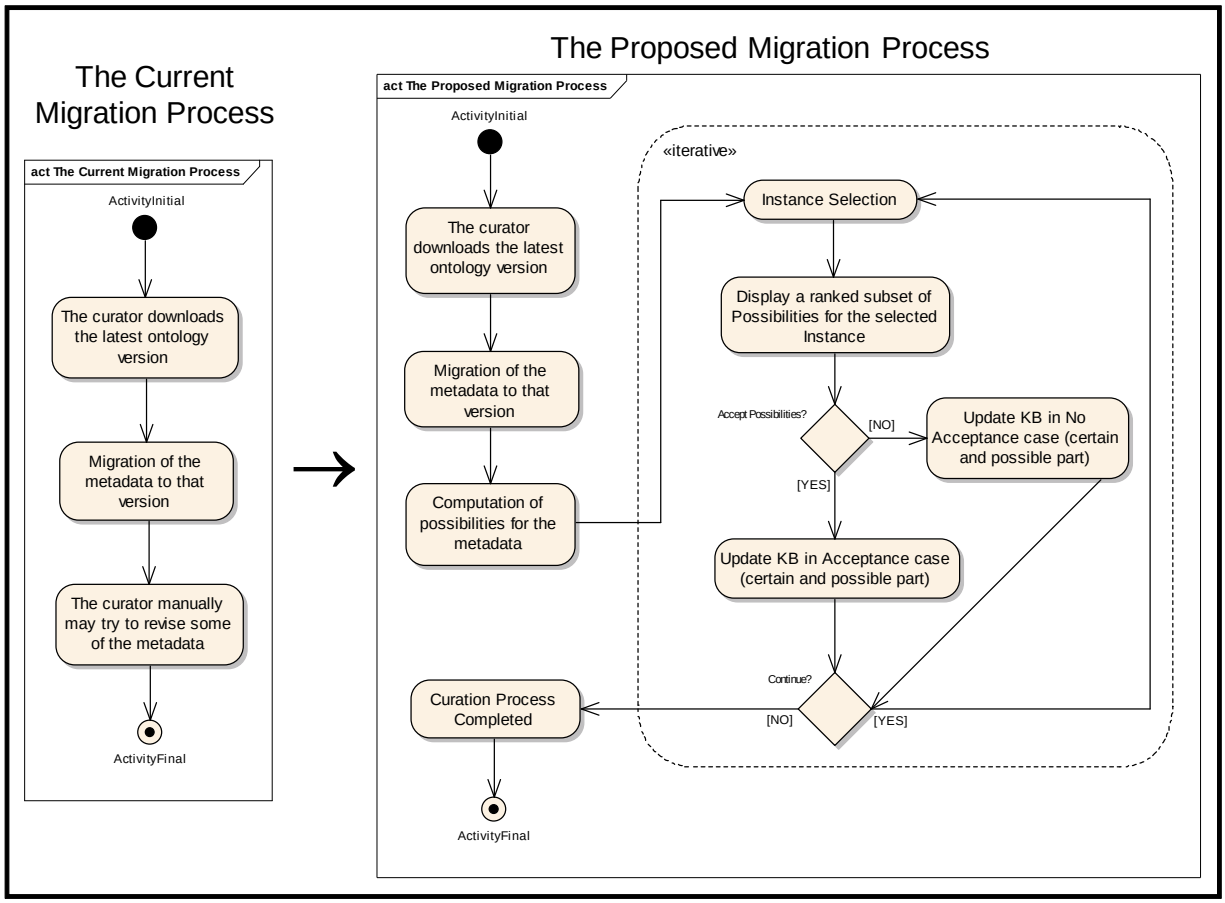

Figure 3: Current and Proposed Migration Process

2. THE APPROACH

2.1 The Life-Cycle

Apart from identifying the information that could be further specialized (as we discussed just before), we would like to *aid* making it as specific as possible. Therefore, we should support flexible and *interactive* processes for managing the computed possibilities, where the user will be able to either *accept* or *reject* the computed *recommendations*, and eventually update the knowledge base reaching to a state where the *MSA* holds (at least for those resources for which this is possible). The *ranking* of possibilities is important for designing user-friendly interaction schemes, since we may have a high number of recommendations. Essentially, we propose a process like the one sketched in the right part of Figure 3. Specifically, assume that the user selects some instances then the system displays ranked all or some of the possible instance descriptions for the selected instances. The user accepts or rejects these instance descriptions and the system updates appropriately the KB and its possible part. Note that the possible part of the KB is stored explicitly and separately. In our toy example, this means that we can rank the possible classes for `c1`, so that if the user is prompted to select a possible class for `c1`, then `Diesel` and `Ecological` will be the first classes to be displayed. If the user rejects the class `Ecological`, then all its subclasses will be rejected from the possible classes (and this reduces the effort required for reaching a state where the *MSA* holds).

2.2 Foundations and Examples

For reasons of space here we describe only the main points of the theory (the reader can refer to the technical report for the details) and provide some indicative examples.

For expressing (actually bounding) the uncertainty regarding the specificity of a description caused by its migration to a new schema, we introduce the notion of *possible instance triples*. To capture the various application-specific assumptions about the specificity of the descriptions of a KB, we introduce the notion of TFP-partition (True-False-Possible partition). We denote the TFP-partition of a KB K by a triple (of the form $\mathcal{C}_i(K)$, M_K, P_K), the first being a set of positive instance triples (explicitly stated or inferrable), the second is a set of negative instance triples, and the last is a set of *possible* instance triples.

We view the *migration* of a set of instance triples to a new schema S' as a *transition* between two TFP-partitions, i.e. $(\mathcal{C}_i(K), M_K, P_K) \rightsquigarrow (\mathcal{C}_i(K'), M_{K'}, P_{K'})$. Note that the new schema S' can be *backwards* or *non-backwards compatible* with the current schema S. Schema S' is *backwards compatible* with S, if the closure of S (based on the standard inference rules of RDF/S) is subset of the closure of S'.

The transition between two TFP-partitions, is governed by few *postulates* which are very general (i.e. RDF/S independent). We adopt two postulates for the case of backwards compatible, and an additional one (third) for the case of

non-backwards compatible schema evolution.

Specifically the first postulate (P1) gives priority to the positive knowledge inferrable from the instance triples and the new schema, and it is consistent with (and reminiscent of) the principle "Recent knowledge prevails the old one" (also, called "Principle of Success" [1] and "Primacy of New Information" [5]).

The second postulate (P2) states that past negative information cannot become possible, meaning that past negative information is preserved as long as it does not contradict with the new positive knowledge.

The last postulate (P3), which is needed only when the new schema is not backwards compatible with the old schema, states and those instance triples that were previously positive, but according to the new schema are not, should be considered in the new TFP-partition as negative (not possible).

Based on the above postulates, a small set of *derivation rules* are defined for carrying out a transition for the case of RDF/S. It is important to note that transitions between TFP-partitions can be defined without having to keep any negative information (i.e. the "M" part of a TFP-partition). Instead only the certain and the possible part of the KB has to be kept, reaching to what we call *extended KB* (*eKB*). A further compression of the possible part of the eKB is feasible and suitable for large data sets. Specifically a compact (interval-based) representation of the set of possible instance triples is possible. However the important point is that if the curation process is followed and the curator accepts/rejects the migration-related uncertainties, then the possible part of the KB becomes empty, i.e. no extra storage is required.

Figure 4 illustrates two migrations. The initial schema (at left) contains only one class `Person`. The KB contains only one instance triple, stating that `John` is a `Person`. In the second schema (at the middle) we can see that the class `Person` has been extended with five subclasses. During the migration all these classes are considered as possible classes for `John`. In the figure they are enclosed by a dashed rectangle and the natural numbers indicate their ranking. Now suppose that the system suggests as possible classes for `John` only those with rank 1, i.e. the class `Student` and the class `Employee`. If we suppose that the curator rejects them, then at the right of the figure we can see the new KB. Notice that the set of possible instance triples becomes empty.

Figure 5 illustrates a variation of the previous scenario, where we assume that the system suggests to the curator only three (of the five) possible classes for `John`, namely the classes `Student`, `PostGraduate`, and `Employee`. Here we assume that the curator decides to accept the recommendation `PostGraduate`. At the right diagram we can see the new state of the KB. The set of new possible instance triples contain only that `John` could be `PhD_Student`.

Figure 6 shows an example of a migration to a non backwards compatible schema (notice that one subclassOf relationship has been deleted). The left diagram shows the possible classes for `John` (result of past migrations). At the bottom of the figure we can see the TFP-partition of these KBs. Note that `John` is no longer `Permanent Employee` due to (P3). Also note that the previously possible instance triple `(John, type, Full-time Permanent Employee)` has been removed and does not belong to $P_{K'}$ because `(John, type, Permanent Employee)` is now negative.

The previous examples involved only classes. Properties are analogously treated. An example is shown at Fig 7.

For reasons of completeness, here we describe the rules that determine how the possibilities after a migration are defined. Suppose we are in the context of a transition $(\mathcal{C}_i(K), M_K, P_K) \rightsquigarrow (\mathcal{C}_i(K'), M_{K'}, P_{K'})$. It follows from the postulates, that for a new class c' (i.e. a class that was not element of S), it holds that: $(o\ type\ c')$ should be placed at $P_{K'}$ iff:
(i) $(o\ type\ c') \notin \mathcal{C}_i(K')$, and
(ii) for all not new (i.e. in S) classes c that are superclasses of c' it holds $(o\ type\ c) \in (\mathcal{C}_i(K') \cup P_K)$.
Analogously, for a new property pr' (i.e. a property that was not element of S), it holds that: the triple $(o\ pr'\ o')$ should be placed at $P_{K'}$ iff:
(i) the triple $(o\ pr'\ o')$ is valid to add, i.e. it respects the domain and range constraints,
(ii) $(o\ pr'\ o') \notin \mathcal{C}_i(K')$, and
(iii) for all not new (i.e. in S) properties pr that are superproperties of pr', it holds $(o\ pr\ o') \in (\mathcal{C}_i(K') \cup P_K)$.
Regarding deletions, $P_{K'}$ will not contain the instance triples of P_K that their "supertriples" involving old classes or old properties do not belong to $\mathcal{C}_i(K') \cup P_K$. The rest of the instance triples in P_K are transferred to $P_{K'}$.

3. THE PROTOTYPE

We have implemented a proof-of-concept prototype, called `RIMQA` (**R**DF **I**nstance **M**igration **Q**uality **A**ssistant)[3], supporting the entire lifecycle process. Some screendumps are shown at Figure 8.

The user selects the source ontology (.rdfs file) and a file that contains instance descriptions (.rdf file) with respect to that ontology. The latter file could be the result of applying an export operation over the system that manages the metadata of an archive. Subsequently, the user selects the destination ontology (.rdfs file), which is a subsequent version of that ontology and optionally the user selects a file with possible instance descriptions (.rdf[4] file) derived from a previous migration with respect to the source ontology and one of its previous versions. The system then automatically migrates the instance descriptions from the source to the destination ontology. Then, it computes the possible instance triples.

After that, if the user presses the "Start Curation" button, the curation process starts. If the user selects the "Statistics" menu, he can see the most indicative statistics about the source and the destination ontology, i.e. (a) the number of original classes, properties, (explicit) schema triples, and instance triples in both ontologies, and (b) the number of added classes and properties, and the number of added and

[3]The tool is available at `http://www.ics.forth.gr/isl/RIMQA/`.
[4]Note that we use the RDF format in order to store possibilities, as they are instance triples.

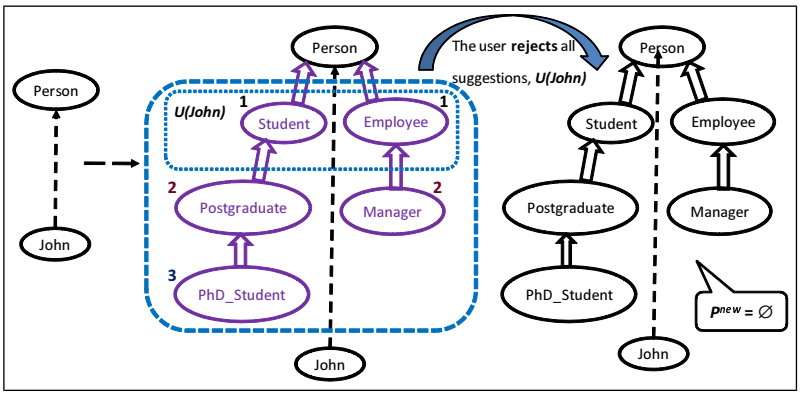

Figure 4: One migration and rejection of the computed recommendations

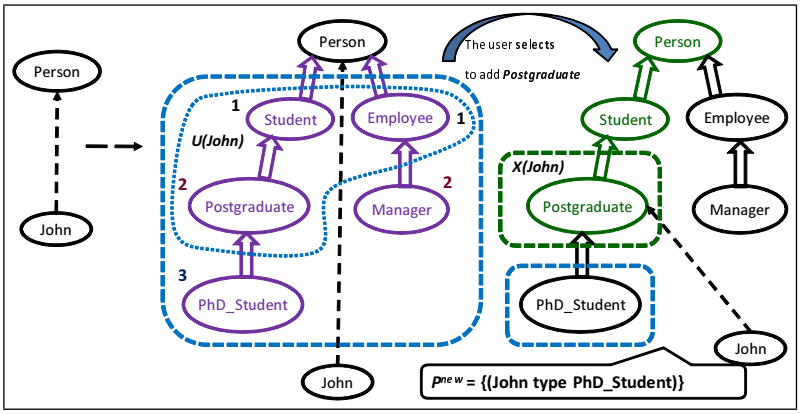

Figure 5: One migration and acceptance of some recommendations

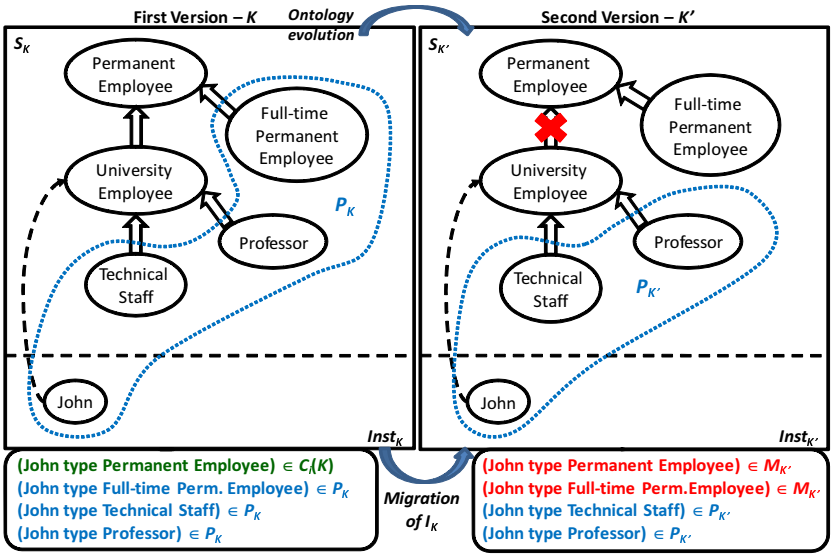

Figure 6: Migration to a non backwards compatible schema

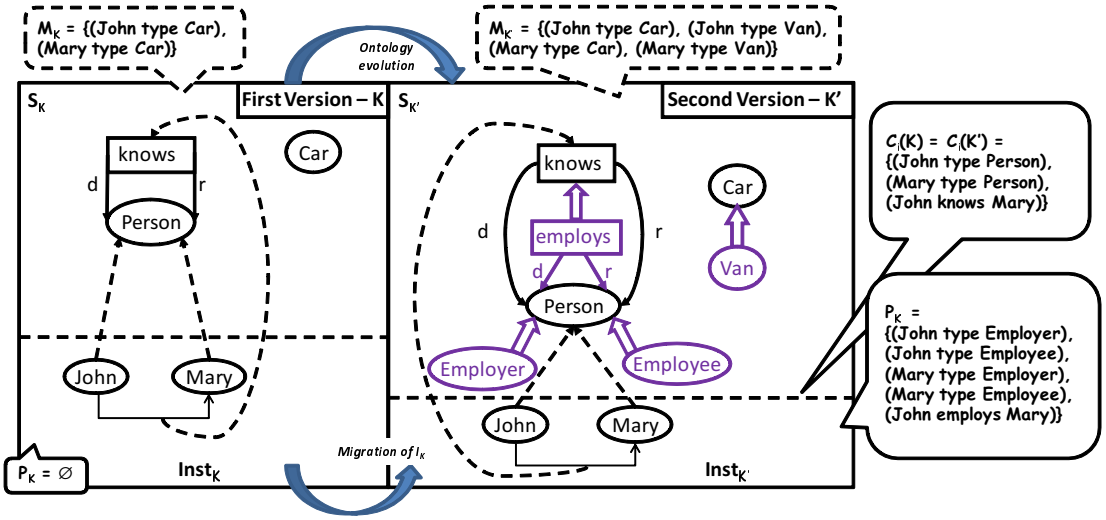

Figure 7: Examples with properties

deleted (explicit and inferred) schema triples in the destination ontology. The user can also get information about the possibilities of the source and the destination ontology, e.g. the number of possible class instance triples and possible property instance triples in both ontologies.

To curate the resulting descriptions ("Curate" menu), `RIMQA` allows the user to inspect all possible class and property instance triples. Regarding class instance triples, all possible class instance triples are listed and the user is able to add (by pressing the "Accept" button) one or more possible class instance triples to the certain part of the extended KB (*eKB*). Subsequently, the selected possible class instance triples and all their supertriples are added to the certain part of the *eKB* and they are removed from the multiple choice list and from the possible part of the *eKB*. The user can also remove (by pressing the "Reject" button) one or more possible class instance triples from the possible part of the *eKB*. Subsequently, the selected possible class instance triples and all their subtriples are removed from the multiple choice list and from the possible part of the *eKB*. After that, the user selects to save the new certain and possible part of the *eKB* (by pressing the "Save eKB" button).

If the user selects to save the *eKB* (by pressing the "Save eKB" button), we store the new instance triples, i.e. the certain part of the *eKB*, in a .rdf file (called "newCertainModel.rdf") and the new possible instance triples, i.e the possible part of the *eKB*, in a different .rdf file ("newPossibleModel.rdf").

4. CONCLUDING REMARKS

The rapid evolution of ontologies requires principles, techniques, and tools for managing the quality of the migrated descriptions, as well as flexible interactive methods for managing the incurred uncertainty. To the best of our knowledge this is the first work that exploits ontology schema evolution for managing the specificity of instance descriptions. According to our opinion this is key issue for the preservation of scientific data, i.e. for e-Science.

Since the ultimate objective is not just the identification of possibilities, but to aid making the instance descriptions as specific as possible, we proposed a specificity *lifecycle management process* that *ranks* the possible instance triples, prompts to the user a subset of the possible instance triples and we show how the extended KB should be *updated* when the user *accepts* or *rejects* some of them. To investigate the feasibility of our approach, we designed and developed a prototype system.

There are several issues for future research. One interesting direction is to generalize our approach to the *XSD*[5]-typed literal values [12] of property instance triples. Such extension would allow reasoning about the *accuracy* of the migrated descriptions over linearly ordered domains (e.g. as consequence of migrating 32-bit floating numbers to a 64-bit representation). Finally, testing and evaluation of the approach with actual curators is worth doing.

Acknowledgements

Work done in the context of NoE *APARSEN* (Alliance Permanent Access to the Records of Science in Europe, FP7, Proj. No 269977, 2011-2014).

5. REFERENCES

[1] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *Journal of Symbolic Logic*, 50(2):510–530, 1985.

[2] C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[3] D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, February 2004. Available at `http://www.w3.org/TR/rdf-schema/`.

[4] Peter Buneman, James Cheney, Wang Chiew Tan, and Stijn Vansummeren. Curated Databases. In *27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS-2008)*, pages 1–12, 2008.

[5]XML Schema Definition

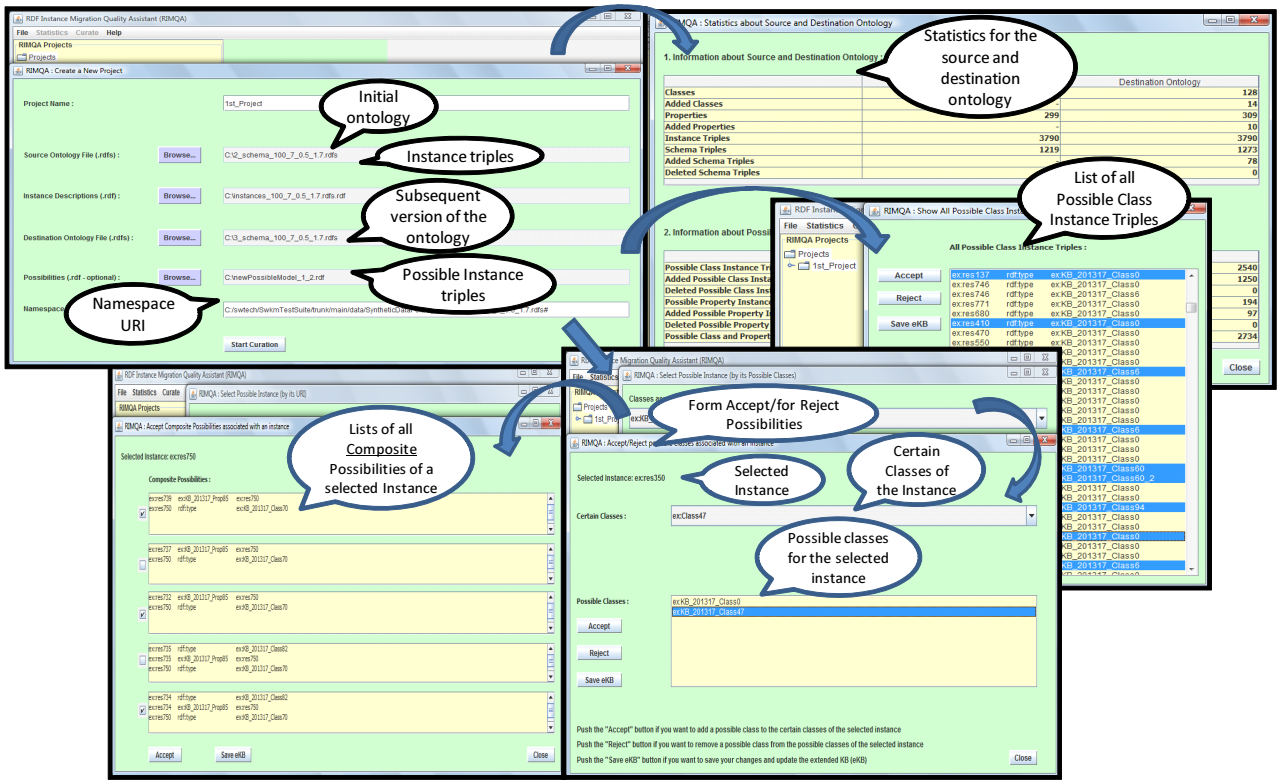

Figure 8: Some screenshots of the GUI of RIMQA

[5] M. Dalal. Investigations into a Theory of Knowledge Base Revision: Preliminary Report. In *7th National Conference on Artificial Intelligence (AAAI-1988)*, pages 475–479, 1988.

[6] M. Doerr and Y. Tzitzikas. Information carriers and identification of information objects: An ontological approach. *Arxiv preprint arXiv:1201.0385*, 2012.

[7] G. Konstantinidis, G. Flouris, G. Antoniou, and V. Christophides. A Formal Approach for RDF/S Ontology Evolution. In *18th European Conference on Artificial Intelligence (ECAI-2008)*, pages 70–74, Patras, Greece, July 2008.

[8] Y. Marketakis, M. Tzanakis, and Y. Tzitzikas. PreScan: Towards Automating the Preservation of Digital Objects. In *Procs of the Intern. Conf. on Management of Emergent Digital Ecosystems MEDES'2009*, Lyon, France, October, 2009.

[9] Y. Marketakis and Y. Tzitzikas. Dependency Management for Digital Preservation using Semantic Web technologies. *International Journal on Digital Libraries*, 10(4), 2009.

[10] Thomas Neumann and Gerhard Weikum. x-RDF-3X: Fast Querying, High Update Rates, and Consistency for RDF Databases. *Proceedings of the VLDB Endowment (PVLDB)*, 3(1):256–263, 2010.

[11] Natalya Fridman Noy and Michel C. A. Klein. Ontology Evolution: Not the Same as Schema Evolution. *Knowledge and Information Systems*, 6(4):428–440, 2004.

[12] David Peterson, Shudi (Sandy) Gao, Ashok Malhotra, C. M. Sperberg-McQueen, and Henry S. Thompson. W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, W3C Working Draft 3, December 2009. Available at `http://www.w3.org/TR/xmlschema11-2/`.

[13] Li Qin and Vijayalakshmi Atluri. Evaluating the validity of data instances against ontology evolution over the Semantic Web. *Information & Software Technology*, 51(1):83–97, 2009.

[14] Satya S. Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Soren Auer, Juan Sequeda, and Ahmed Ezzat. A Survey of Current Approaches for Mapping of Relational Databases to RDF, 2009. Report by the W3C RDB2RDF Incubator Group. Available at `http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf`.

[15] M. Theodoridou, Y. Tzitzikas, M. Doerr, Y. Marketakis, and V. Melessanakis. Modeling and Querying Provenance by Extending CIDOC CRM. *J. Distributed and Parallel Databases (Special Issue: Provenance in Scientific Databases)*, 2010.