# X3ML Framework: An effective suite for supporting data mappings

Nikos Minadakis[1], Yannis Marketakis[1], Haridimos Kondylakis[1], Giorgos Flouris[1], Maria Theodoridou[1], Martin Doerr[1], and Gerald de Jong[2]

[1]Institute of Computer Science, FORTH-ICS, Greece
[2]Delving B.V. The Netherlands
{minadakn,marketak,kondylak,fgeo,maria,martin}@ics.forth.gr
gerald@delving.eu

**Abstract.** The aggregation of heterogeneous data from different institutions in cultural heritage and e-science has the potential to create rich data resources useful for a range of different purposes, from research to education and public interests. In this paper, we present the architecture and functionality of X3ML data exchange framework, that handles effectively and efficiently the schema mapping, URI definition and generation, and data transformation steps of the provision and aggregation process. The X3ML framework is based on the *X3ML mapping definition language* that offers the building blocks for describing both schema mappings and URI generation policies, and the *X3ML engine*, that handles the URI generation and the data transformation. The X3ML framework supports the cognitive process of mapping and it has a lot of advantages compared to other existing tools including that the schema mappings are expressed in a declarative way, and are both human and machine readable allowing domain experts to understand them, the schema matching and the URI generation policies comprise different distinct steps in the exchange workflow, and follow different life cycles. Furthermore X3ML is symmetric and potentially invertible allowing bidirectional interaction between providers and aggregator and thus supporting not only a rich aggregators' repository but also corrections and improvements in the providers' data bases.

**Keywords:** Data Mappings, Data aggregation, URI generation

## 1 Introduction

Managing heterogeneous data is a challenge for cultural heritage institutions, such as archives, libraries, and museums, but equally for research institutes of descriptive sciences such as geology, biodiversity, clinical studies etc. These institutions host and develop various collections with heterogeneous material, often described by different metadata schemas. In order to provide uniform access to heterogeneous and autonomous data sources, complex query and integration mechanisms have to be designed and implemented. There are now significant

numbers of projects that aggregate data with these purposes in mind. Besides data aggregation, memory institutions try more and more to move away from monolithic legacy collection management systems and to publish their collections using rich metadata and schemata. Similar efforts are done at geoscience and biodiversity institutions.

In order to allow data transformation and aggregation, it is required to produce mappings, to relate equivalent concepts or relationships from the source schemata to the aggregation schema, i.e. the target schema, in a way that facts described in terms of the source schema can automatically be translated into descriptions in terms of the target schema, or "enterprise model" as Calvanese et al. [6] describe it. This is the mapping definition process and the output of this task is the mapping, i.e., a collection of mapping rules.

In this paper we describe the X3ML framework, which is able to support the data aggregation process by providing mechanisms of data transformation and URI generation. Mappings are specified with the X3ML mapping definition language which is a declarative, human readable language that supports the cognitive process of a mapping. Unlike XSLT, that can only be understood by IT technicians, X3ML can be understood by non-technical people, so a domain expert is capable of testing the semantics, reading and validating the schema matching. This model carefully distinguishes between mapping information from the domain experts who know and provide the data and that created by the IT technicians who actually implement data translation and integration solutions, and serves as an interface between both.

A common problem of a schema matching and transformation process is that the IT experts do not fully understand the semantics of the schema matching and the domain experts do not understand how to use the technical solutions. For this reason, in our approach the URI generation and the schema matching processes are separated, so the schema matching can be fully performed by the domain expert and the URI generation by the IT expert, and therefore solving the bottleneck that requires that the IT expert understands the mapping. Furthermore this keeps the schema mappings between different systems harmonized since the schema mappings definitions do not change in contrast to the URIs that may change between different institutions and are independent of the semantics. XSLT and R2RML have tightly coupled the URI generation from the schema matching processes.

Our approach completely separates the definition of the schema matching from the actual execution. This is important because different processes might have different life-cycles; in particular the schema matching definition has a different life-cycle compared to the URI generation process. The former is subject to more sparse changes compared to the latter.

The remainder of this paper is organized as follows: Section 2 discusses the related work. Section 3 discusses about the background. Section 4 describes the details of the X3ML framework. Section 5 enumerates different usages of the framework and provides the evaluation results. Finally Section 6 concludes and discusses about the future directions of our work.

## 2 Related Work

Mapping relational databases (RDB) to RDF became a quite active field the last few years. This happens as the majority of data currently published on the web are still stored in relational databases with local schemata and local identifiers. There are several solutions that fall into this category. Direct Mapping [13] maps automatically relational tables to RDF classes and attributes to RDF properties. D2R MAP [5] is a declarative language to describe mappings between relational databases and OWL/RDFS ontologies. Triplify [2] maps HTTP-URI requests onto RDB queries and translates the resulting relations into RDF statements. R2RML[1] which is a mapping language proposed by W3C in order to standardize RDB to RDF mappings.

In addition there are similar works that map CSV files to RDF. XLWrap's mapping language [11] provides conversions from CSV and spreadsheets to RDF data model. Mapping Master's M2 [14] converts data from spreadsheets into OWL statements. Vertere [2] is a conversion tool based on a templating mechanism.

Finally there are tools, that provide mappings from XML to RDF leading to mappings in the syntactic level rather than in the semantic level. Tools in this category include tools based on XSLT (i.e. Krextor [10], AstroGrid-D [3]), tools based on XPATH (i.e. Tripliser [4]) and XQUERY (i.e. XSPARQL [4]). There are also other approaches that exploit mapping technologies to publish their data as linked data. For example the Smithsonian American Art Museum used KARMA [17] to publish their data as linked data, a tool trying to automate the mapping process allowing users to adjust the generated mappings. However, there is still no clear distinction on the work of the domain and the IT experts which perplexes the whole workflow. KARMA uses R2RML model so it inherits the issue of tight coupling between the schema matching and the URI generation.

All these different approaches prove that there is no standard model to support mapping of data sources other than relational, the technologies used are too complex to be used by the domain experts and the whole workflow is not well-defined. Compared to these works our work (a) uses a simple model for defining the mappings in a way that is comprehensible and readable from the domain experts, (b) is generic because the mapping definitions are not tied to the implementation of the data transformation engine, (c) supports incremental changes of source and target schema, (d) supports customized URI generation policies and (e) promotes the collaborative work of experts with different roles on the mapping process.

[1] http://www.w3.org/TR/r2rml/
[2] https://github.com/knudmoeller/Vertere-RDF
[3] http://www.gac-grid.de/project-products/Software/XML2RDF.html
[4] http://daverog.github.io/tripliser/

## 3 Background

The main pillar of our work is the Synergy Reference Model (for short SRM)[5] which is an initiative of the CIDOC CRM Special Interest Group[6]. It is a reference model for a better practice of data provisioning and aggregation processes, primarily in the cultural heritage sector, but also for e-science. It is based on experience and evaluation of national and international information integration projects. It defines a consistent set of business processes, user roles, generic software components and open interfaces that form a harmonious whole. Currently a draft version of the model is available online[7], arenas2010semantics still being evolved and enriched. The goal of SRM is to: (a) describe the provision of data between providers and aggregators including associated data mapping components, (b) address the lack of functionality in current models (i.e. OAIS [12]) and practice, (c) incorporate the necessary knowledge and input needed from providers to create quality sustainable aggregations and, (d) define a modular architecture that can be developed and optimized by different developers with minimal inter-dependencies and without hindering integrated UI development for the different user roles involved.

SRM aims at identifying, supporting or managing the processes needed to be executed or maintained between a provider (the source) and an aggregator (the target) institution. It supports the management of data between source and target models and the delivery of transformed data at defined times, including updates. This includes a mapping definition, i.e., specification of the parameters for the data transformation process, such that complete sets of data records can automatically be transformed. A graphical representation of the data provisioning workflow is shown in Figure 1

The main steps of the data provisioning workflow are:

- **Schema matching**: Source and target schema experts (a.k.a the domain experts) define a schema matching which is documented in a schema matching definition file. This file should be human and machine readable and it is the ultimate communication mean on the semantic correctness of the mapping.
- **Instance generation specification**: In this step the URI generation and datatype conversion policies are defined for each instance of a target schema class referred to in the matching. In this step only IT experts are involved and domain experts have no interest or knowledge about it.
- **Terminology mapping**: Finally, the terminology mappings between source and target data/terms are defined. Providers may use anything from intuitive lists of uncontrolled terms up to highly structured third party thesauri.
- **Transformation**: Once the mapping definition has been finalized (and all syntax errors are resolved) the data needs to be transformed. The transformation process itself may run completely automatically. In the case where

[5] by the time of writing this, there is a draft version at `http://www.cidoc-crm.org/docs/SRM_v0.1.pdf`

[6] `http://www.cidoc-crm.org/who_we_are.html`

[7] `http://www.cidoc-crm.org/official_release_cidoc.html`

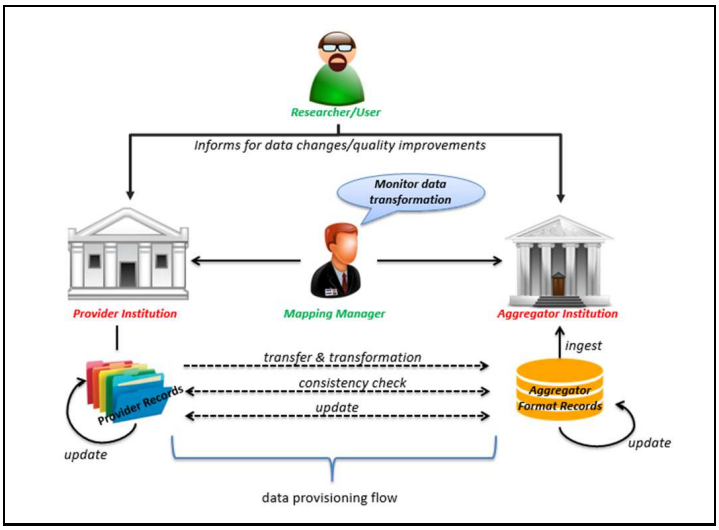


**Fig. 1.** The data provisioning workflow

any issues arise, the aggregator can resolve them on a temporary or permanent basis but it is also possible that these records are send back to the provider for further analysis and resolution. The final result is a set of valid target records.

- **Ingestion**: Once records are transformed, an automated translation for source terms using a terminology map follow. The transformed records will then, be ingested into the target system.
- **Change detection**: After the ingestion of the records all changes that may affect the consistency of provider and aggregator data are monitored. SRM describes 18-20 different updating and transformation reasons and is the only framework at the moment which takes the maintance into account.

## 4 The X3ML framework

The X3ML framework comprises the X3ML Mapping Definition Language and the X3ML Engine. Below we will describe them.

### 4.1 X3ML Mapping Definition Language

The X3ML mapping definition language is an XML based language which describes schema mappings in such a way that they can be collaboratively created and discussed by experts. The X3ML language was designed on the basis of work that started in FORTH in 2006 [9] and emphasizes on establishing a standardized mapping description which lends itself to collaboration and the building of a mapping memory to accumulate knowledge and experience. It was adapted primarily to be more according to the DRY principle (avoiding repetition) and to be more explicit in its contract with the URI Generating process.

X3ML separates schema mapping from the concern of generating proper URIs so that different expertise can be applied to these two very different responsibilities.

**Schema matching:** Schema matching is performed by domain experts who need to be concerned only with the correct interpretation of the source schema. The structure of X3ML is quite easy to understand consisting of:

- **a header** that contains basic information (title, description, contact persons), the source and target schemata and sample records
- **a series of mappings** each containing a domain (the main entity that is being mapped) and a number of links which consist of a path and a range. Each link describes the relation (path) of the domain entity to the corresponding range entity.

The basic mapping scheme and the corresponding XML structure is shown in Figure 2. Each entity-relation-entity of the source schema is mapped individually to the target schema and can be seen as self-explanatory, context independent proposition. An X3ML structure consists of:

- the mapping between the source domain and the target domain
- the mapping between the source range and the target range
- the proper source path
- the proper target path
- the mapping between source path and target path

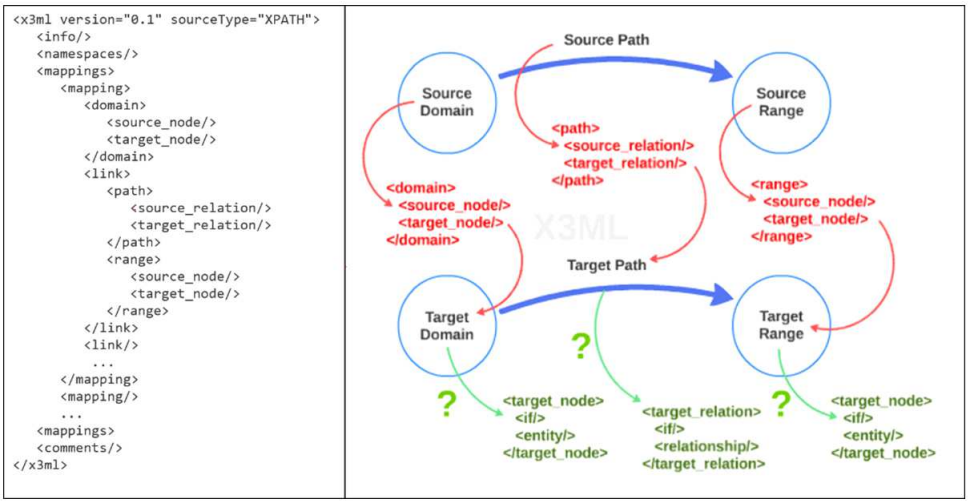


**Fig. 2.** The structure of an X3ML mapping

The X3ML mapping definition language supports 1:N mappings and uses the following special constructs:

- **intermediate nodes** used to represent the mapping of a simple source path to a complex target path (a sequence of path-{entity-path}).

- **constant expression nodes** used to assign constant attributes (e.g. a constant type) to an entity.
- **conditional statements** within the target node and target relation support checks for existence and equality of values and can be combined into boolean expressions.
- **"Same as" variable** used to identify a specific node instance for a given input record that is generated once but is used in a number of locations in the mapping.
- **Join operator (==)** used in the source path to denote relational database joins
- **info and comment blocks** throughout the mapping specification bridge the gap between human author and machine executor.

The tools that are currently used to produce the X3ML mapping definition are restricted to consuming XML input records. [8] As a result, XPath is used to specify the source elements and paths which are evaluated within the context of the source domain. There is ongoing work for an extended version that will support RDF input (see Section 4.5).

**URI generation policy:** The definition of the URI generation policy follows the schema matching and is performed usually by an IT expert who must ensure that the generated URIs match certain criteria such as consistency and uniqueness. A set of predefined URIs (UUIDs, Lietrals) and templates are available but any URI generating function can be implemented and incorporated in the system. In the X3ML definition, the target domain and range contain the functions that generate URIs or literals.

The result of the schema matching and URI generation policy steps is a complete X3ML mapping definition file that will be fed to the X3ML engine for the transformation of the data.

Figure 3 shows how a simple relational database entry that specifies the weight of a coin is mapped and expressed with respect to the CIDOC CRM schema[7]. The XML structure for the mappings of this example can be found online[9].

### 4.2 X3ML Engine

The X3ML engine realizes the transformation of the source records to the target format. The engine takes as input the source data (currently in the form of an XML document), the description of the mappings in the X3ML mapping definition file and the URI generation policy file and is responsible for transforming the XML document into a valid RDF document which is equivalent with the XML input, with respect to the given mappings and policy. The engine has been originally implemented in the context of the CultureBrokers project co-funded by the Swedish Arts Council and the British Museum.

[8] http://www.ics.forth.gr/isl/3M/

[9] `http://139.91.183.44/x3mlEditor/ViewPublished?type=Mapping&id=1`

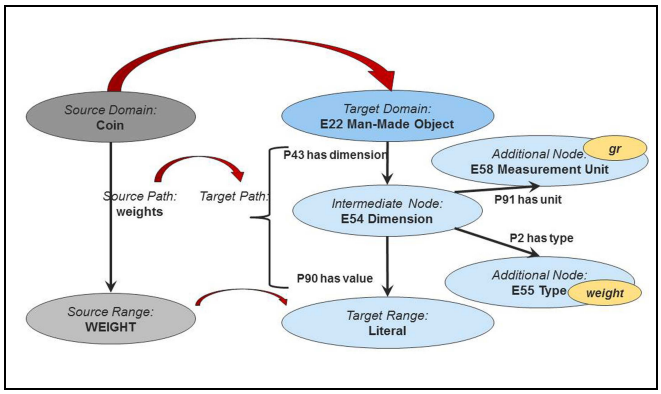


**Fig. 3.** Mapping relational db data to CIDOC CRM

### 4.3 Design, Architecture and Implementation

The X3ML Engine has been designed with respect to the following design principles:

- *Simplicity.* It is easier to create complicated things than it is to find the simplicity in something that would otherwise be complex. One important way to achieve simplicity and clarity is by carefully naming things so that their meaning is as obvious as possible to the naked eye.
- *Transparency.* The most important feature of X3ML is its general application to mapping creation and execution and hopefully its longevity. People must be able to easily understand how it works. The cleanerthe core design of this engine and X3ML language, and the clearer its documentation, the more readily it will get traction and become the basis for future mappings.
- *Re-use of Standards and Technologies.* The best way to build a new software module is to carefully choose its dependencies, and keeping them as small as possible. Building on top of proven technologies is the quickest way to a dependable result.
- *Facilitating Instance Matching.* This involves extracting semantic information with the intent of generating correct instance URIs.

Figure 4 depicts the main components of the engine. The *Input Reader* component is responsible for reading the input data (currently we support XML documents, however as we describe later in Section 4.5 more formats will be supported using proper extensions). The *X3ML Parser* component is responsible for reading and manipulating the X3ML mapping definitions. The component *RDF Writer* outputs the transformed data into RDF format. The *Instance Generator* component produces the URIs and the labels based on the descriptions that exist in the mappings and finally the *Controller* component coordinates the entire process.

The X3ML engine framework has been implemented in Java, producing a single artifact in the form of a JAR file which contains the engine software. For supporting the functionality of the main components we exploited a set of

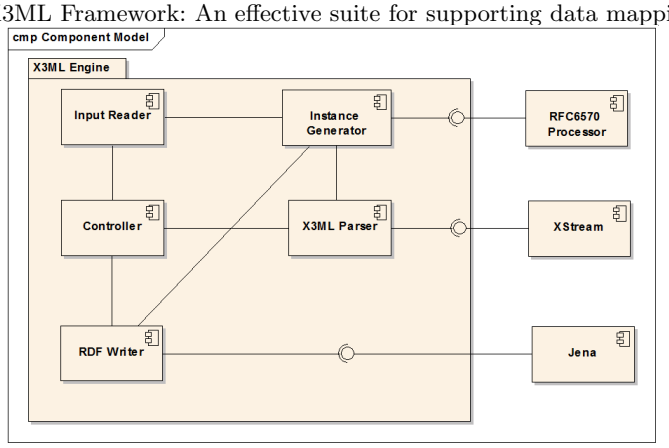


**Fig. 4.** The main components of X3ML Engine

third-party software libraries. For instance we used XStream[10] for parsing XML-based documents, Handy URI Templates[11] to support the generation of valid URIs and Jena[12] for building the RDF output. The source code the the X3ML engine framework is available under the Apache license and can be found at `https://github.com/delving/x3ml`.

### 4.4 Functionality

The X3ML Engine takes as input source XML records and generates RDF triples consisting of subject, predicate, and object. The subject and the object are "values", generally consisting of URIs, but objects can also be labels or literal values.

The generation of values (URIs, or literals) is being handled by the Instance Generator component. The following block shows two configurations; for generating (a) URIs and (b) label values.

```
<instance_generator name="[gen-name]">
    <arg name="[arg-name]" type="[arg-type]">[arg-value]</arg>
    ...
</instance_generator>
<label_generator name="[gen-name]">
    <arg name="[arg-name]" type="[arg-type]">[arg-value]</arg>
    <arg name="language" type="constant">[language-code]</arg>
    ...
</label_generator>
```

For each entity there must exist one `instance_generator` and any number of subsequent `label_generator` blocks. The argument type allows for choosing between xpath and constant and there is a special argument type called position which gives the value generator access to the index position of the source node within its context. The argument with the name `language` defines the language tag of the generated value. If it is empty then it is implied that the generated

[10] `http://x-stream.github.io/`
[11] `https://github.com/damnhandy/Handy-URI-Templates`
[12] `https://jena.apache.org/`

value will not have it (i.e. in the case of number values). The engine provides default implementations for producing: (a) URIs, (b) UUIDs, (c) literal values and (d) constant values.

The Instance Generator component is configured through an XML file (which is given as input in the X3ML Engine). When URIs are to be generated on the basis of source record content, it is wise to leverage existing standards and reuse the associated implementations. For template-based URI generation there is available the RFC 6570 [8] standard. So, the component uses an existing implementation library as described in Section 4.3. Whenever the required URIs or labels cannot be generated by the default generators, the simple templates, or the URI templates, it is always possible to insert a special generator in the form of a class implementing the *InstanceGenerator* component interfaces.

### 4.5 Configuration/Extensibility

As already discussed the current version of the X3ML Engine, takes as input the source data in the form of an XML document. One extension (which is currently under development) is to support other types of input. To this direction we have started working on supporting RDF input. This requires several modifications in the design and implementation of the engine. More importantly the basic construct that we use for reading the source data will be an RDF model (i.e. Jena, Sesame), so instead of XPATH we will be able to use SPARQL [16]. Furthermore we will enhance the Instance Generator component since we will be able to carry the URIs from the source data to the target data if needed.

One apparent advantage of this approach is that the framework will support input and output of the same format. This sparked the light to investigate another direction; that of invertible X3ML mappings. In an invertible X3ML mapping, one can identify, in a unique manner, (and consequently regenerate) the data in the source dataset that led to the creation of each piece of data in the target dataset. Based on this idea, below we formalize the notion of invertibility, by trying to identify how X3ML maps the source data to the target data.

In particular, we view an X3ML mapping as an association between a "pattern" (say $P_s$) in the source dataset with a "pattern" (say $P_t$) in the target dataset. This association essentially describes what to put in the target dataset ($P_t$) whenever $P_s$ is encountered in the source dataset. Formally, we model $P_s$ and $P_t$ as SPARQL graph patterns [15, 1] so an X3ML mapping $m$ is just a pair ($P_s$,$P_t$) of SPARQL graph patterns.

Then, given a set of X3ML mappings (say $M$), we say that $M$ is invertible if and only if we can guarantee that whenever a pattern (say $P_t$) is found in the target dataset, we can identify in a unique manner the pattern $P_s$ that generated it (i.e., caused its inclusion in the dataset). To determine that, we look at each $P_t$ in $M$ (and its corresponding $P_s$), and identify those mappings that can potentially lead to the same triples to be generated from different source triples.

## 5 Evaluation and Usage

The X3ML engine is being exploited by several European projects. Specifically, the ARIADNE project[13] initiated several mapping activities using X3ML Engine, to convert existing schemata of archaeological data to CIDOC CRM and its extension suite. The partners of ARIADNE project had extensively used X3ML for the definition of mappings from various categories of databases, including archeological museums, buildings, ancient Roman coins, and more. The ResearchSpace project[14] is developing a collaborative environment for humanities and cultural heritage research. The project has been using X3ML for the mapping and transformation of the Rijksmuseum, the British Museum and the Yale Center for British Art (YCBA) data. Specifically for the case of the Rijksumuseum domain experts from both Rijksumuseum and the British Museaum were able to succesfully map and transform their data without the assistance of any IT expert. X3ML engine is also being exploited by the transformation services of the Greek national implementation of the European LifeWatch [3] infrastructure for biodiversity to transform biodiversity metadata/data such as Darwin Core formats to a CIDOC CRM family semantic models.

To evaluate[15] the performance of the x3ml engine we used an XML input and a X3ML mapping example coming from the ARIADNE Project as a base to produce synthetic data that was provided as input to the X3ML engine. Three X3ML mapping files were created containing 10,100 and 1000 mappings and 4 XML input files containing 10,100,1000 and 10000 records. Figure 5 displays the evaluation results. We can observe that the the overall time depends on both the number of mappings and the size of the input. For example, as we can see from the evaluation results, the time required for data transformation is approximately one second when the size of the input is low (10 records) even if the mappings are many(from 10 to 1000). As the size of the input increases however, the overall time that is required increases as well. Note, that the total number of output records is the total number of input records multiplied with the number of mappings (i.e. 10 input records with 10 mappings will produce 100 output records). Concluding, we can see that are the execution time is affected equally by the number of the mappings and the records, and it is related with the number of the links that are created during the transformation process.

## 6 Conclusion and Future Work

This paper presents a novel framework for the management of the core processes needed to create, maintain and manage mapping relationships between different data sources. The main pillars of this work is the Synergy Reference Model and the X3ML specification language. Based on these works we designed and implemented the X3ML Engine framework; a tool that supports the transformation

[13] `http://www.ariadne-infrastructure.eu/`
[14] `http://www.researchspace.org/`
[15] The experiments were run in a PC with an i7 processor and 8GB of RAM.

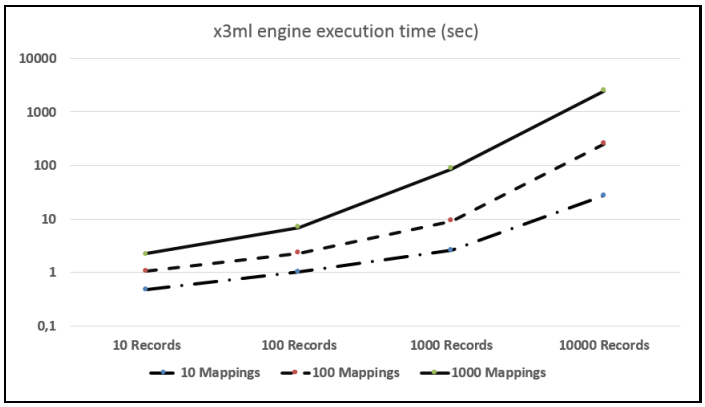


**Fig. 5.** x3ml Engine Evaluation Results

process and the generation of URIs and values. This tool is characterized by its scalability in terms of number of providers, consistent mappings and related end up processes. We demonstrated some of our experiences on using the aforementioned framework and discuss about the evaluation results. In future we plan to continue working on the extended version of the framework that will support different types on input (i.e. RDF documents) and investigate the invertible X3ML mappings functionality.

### Acknowledgement

This work was partially supported by the project *PARTHENOS* (H2020 Research Infrastructures, 2015-2019), the project *ARIADNE* (FP7 Research Infrastructures, 2013-2017), and the *LifeWatch Greece* project (National Strategic Reference Framework, 2012-2015).

## References

1. M. Arenas, C. Gutierrez, and J. Pérez. *On the Semantics of SPARQL*. Springer, 2010.
2. S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller. Triplify: light-weight linked data publication from relational databases. In *Proceedings of the 18th international conference on World wide web*, pages 621–630. ACM, 2009.
3. A. Basset and W. Los. Biodiversity e-science: Lifewatch, the european infrastructure on biodiversity and ecosystem research. *Plant Biosystems-An International Journal Dealing with all Aspects of Plant Biology*, 146(4):780–782, 2012.
4. S. Bischof, S. Decker, T. Krennwallner, N. Lopes, and A. Polleres. Mapping between rdf and xml with xsparql. *Journal on Data Semantics*, 1(3):147–185, 2012.
5. C. Bizer. D2r map-a database to rdf mapping language. 2003.
6. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *KR*, pages 2–13, 1998.
7. M. Doerr. The cidoc conceptual reference module: an ontological approach to semantic interoperability of metadata. *AI magazine*, 24(3):75, 2003.

8. J. Gregorio, R. Fielding, M. Hadley, M. Nottingham, and D. Orchard. Rfc 6570: Uri template. *Internet Engineering Task Force (IETF) Request for Comments*, 2012.
9. D. P. Haridimos Kondylakis, Martin Doerr. Mapping language for information integration. *Technical Report ICS-FORTH*, 385, 2006.
10. C. Lange. Krextor-an extensible framework for contributing content math to the web of data. In *Intelligent Computer Mathematics*, pages 304–306. Springer, 2011.
11. A. Langegger and W. Wöß. *XLWrap–querying and integrating arbitrary spreadsheets with SPARQL*. Springer, 2009.
12. B. Lavoie. Meeting the challenges of digital preservation: The oais reference model. *OCLC Newsletter*, 243:26–30, 2000.
13. T. B. Lee. Relational databases on the semantic web. *Design Issues (published on the Web)*, 1998.
14. M. J. O?Connor, C. Halaschek-Wiener, and M. A. Musen. Mapping master: A flexible approach for mapping spreadsheets to owl. In *The Semantic Web–ISWC 2010*, pages 194–208. Springer, 2010.
15. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of sparql. In *International semantic web conference*, volume 4273, pages 30–43. Springer, 2006.
16. E. Prud?Hommeaux, A. Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.
17. P. Szekely, C. A. Knoblock, F. Yang, X. Zhu, E. E. Fink, R. Allen, and G. Goodlander. Connecting the smithsonian american art museum to the linked data cloud. In *The Semantic Web: Semantics and Big Data*, pages 593–607. Springer, 2013.