ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-1223

VMAP: A Visual Schema Mapping Tool[1]

Florin Chertes[2] and **Ingo Feinerer**[3]

Abstract. Schema mapping languages are a key technique for data engineers to perform data exchange and integration in the context of the semantic web. Visual schema mapping languages provide an extra benefit as they help to visually describe the flow of data between various sources and destinations in the Internet. We present a tool called VMAP for implementing visual schema mappings. Its focus is on standardization and modularity.

1 INTRODUCTION

The semantic web [1, 9] provides the playground for a variety of applications of intelligent systems and artificial intelligence. However, one of the main challenges for the semantic web is the transition from unstructured format of most web pages to a structure which can be semantically processed by machines. Schema mapping languages help to fill this gap as they provide techniques for mapping data between databases. They help to identify the relationships between data and provide a semantic underpinning by linking data. Recently, UMAP [2] was proposed as a middleware and layer for implementing high-level schema mapping languages with a strong focus on standardization. In this paper, we introduce the tool VMAP which implements the core functionality proposed by UMAP.

In section 2, we give a short introduction to UMAP and visual schema mappings. In section 3, we outline our demonstration scenario. Section 4 gives a succinct system overview. Section 5 gives a summary and concludes.

2 UMAP AND VISUAL SCHEMA MAPPINGS

Consider a schema mapping as depicted in Figure 1 (adapting a scenario as shown in [2, Figure 2]). It shows a mapping in the context of the administration of employees and implements a similar mapping from a high-level visual schema mapping language called CLIP [8]. This high-level mapping diagram from [2, Figure 1], not shown here, presents on the source side (left side) graphical elements that are instances of the visual language constructs. The `regEmp` is described by the visual language construct (in CLIP terminology) named *multiple element*, the `ename` or `sal` by the *single elements*, and strings or integers by *value nodes*. The mapping elements *builders* and *build nodes* acting on *multiple elements* like `regEmp` create on the target side (right side) also *multiple elements*: `employee`. The *build nodes* iterate over the source *multiple elements* mapping them to the target. The *build nodes* perform also a selection so that only those elements that satisfy a certain condition, in this case the salary greater than 11000, are mapped to the target. Further, the language construct *value mapping* maps `ename` from the source to `@name` from the target.

Figure 1 shows the structure of a generated UMAP schema mapping corresponding to this previously explained CLIP mapping. The visual language UMAP uses the standardized Unified Modeling Language (UML) [6] and Object Constraint Language (OCL) [7] to describe the schema mapping. This means the visual constructs of UMAP are classes, attributes and associations from UML augmented by OCL expressions which define the operations used to map the source to the target. The source side of the UMAP diagram presents the classes `regEmpSet` and `regEmp`. They are connected by an aggregation and have the following attributes: `ename` of type `string` and `sal` of type `int`. The target side uses similar UMAP language constructs as classes: `employeeSet` and `employee`. The schema mapping is achieved in UMAP by the help of association classes, as iterators, e.g., the class `Builder` or as atomic mapping elements, e.g., `ValueMap`. The two different association classes are connected also by an association allowing the iterator to trigger the mapping. These mapping functions, `Builder::build` and `ValueMap::map` are defined by OCL post-condition expressions. Details of such a UMAP translation are described in [2].

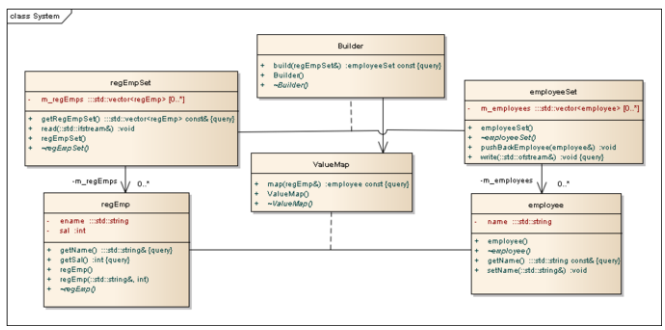

Figure 1. Visual schema mapping using UMAP

3 DEMONSTRATION SCENARIO

We present the work flow of data exchange using our VMAP tool chain from the perspective of a data engineer for the semantic web. Our reference scenarios are similar to those presented in [8]. The user of VMAP is the practitioner or researcher that already has a visual high-level schema mapping language like CLIP and the tool for his own visual language is still missing the component producing the executable from the model. More than that, if he has a complete implementation, then it is difficult to compare its performance with other similar tools. In this situation VMAP can be used as the middleware to solve these both problems.

[1] This work was supported by the Austrian Science Fund (FWF), project P25207-N23.
[2] Vienna University of Technology, Austria, email: FlorinChertes@acm.org
[3] Vienna University of Technology, Austria, email: feinerer@logic.at

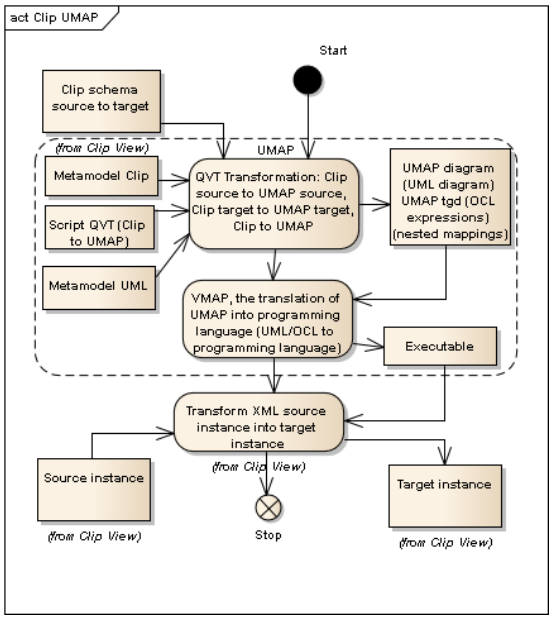

Figure 2. VMAP used as middleware to implement UMAP

As presented in Figure 2, the first activity is the Query View Transformation (QVT) [5] of a schema mapping scenario using a high-level language, in our particular case CLIP, to a scenario using UMAP as schema mapping language. This activity has four inputs, presented on the left side and one output, on the right side. The first of the inputs is the user's schema mapping scenario. The next two inputs are artifacts that must be provided by the user of the VMAP tool: the formal meta-model of the user's schema mapping language, thus the proposed modeling language has a well defined syntax and is called a formal modeling language and the QVT scripts, transforming user's scenario to UMAP. QVT is hereby an OMG standard that provides a methodology for model transformation. The techniques necessary to construct the QVT script translating a user's model to UMAP are e.g. described in [3, 4]. The fourth and the last of the inputs is the UMAP meta-model, which is a subset of the UML meta-model and is a part of VMAP. The output is the schema mapping scenario translated to UMAP.

The next activity presented in the Figure 2 corresponds to VMAP, the translation of a UMAP scenario into an executable. The whole responsibility of the implementation is taken by VMAP. Similar schema mapping languages equipped with VMAP could easily compare their properties because their implementation uses only different instances of VMAP. Another advantage of using VMAP as a middleware are the standards on which VMAP is based. UMAP uses OCL, another OMG standard, to define the schema mappings. VMAP translates the OCL expressions to a particular programming language as follows. Using a UML modeling tool first it exports the model to XMI, an XML file containing the model, second it parses it to extract the OCL expressions and compiles them to the target language. Finally it exports the skeleton of the mapping functions from the UML modeling tool and fills them with the functionality extracted from the OCL expressions. Using a build environment specific for a programming language, VMAP triggers the compilation of the generated source code, thus the executable is produced. The executable reads the source instance from a database, that in the case of our demonstration scenario is an XML file, maps it to the target instance and writes it to a database, another XML file, achieving the mapping scenario.

4 SYSTEM OVERVIEW

The system using VMAP consists of two important parts. First, the model translating part with the following components: UMAP meta-model, high-level schema mapping language meta-model, QVT scripts, the mapping scenario in the high-level modeling language and the translated scenario to UMAP. Second, the executable creating part with the VMAP tool chain compiling the OCL expressions to a programming language and the executable mapping the source instance to the target instance.

The operating system employed is Microsoft Windows but no specific property is used, with the consequence that Linux is an alternative choice. The UML modeling tool creating the function skeletons from the UMAP model and triggering VMAP is Enterprise Architect from Sparx Systems. VMAP does not have any particular dependency on this modeling tool so similar tools could be employed. VMAP uses the Microsoft Visual C++ build environment to create the executables but another build environment based on Linux could replace it.

5 CONCLUSION

UMAP is a recent layer for schema mapping languages with a strong emphasis on industry standards. It can be used as a visual mapping language itself or as middleware, having expressive power similar to CLIP [2] or other comparable high-level visual schema mapping languages.

The tool VMAP provides the reference implementation for UMAP and enables a data engineer to semantically relate different data sources and destinations in the Internet. This is useful for improving the current state of the semantic web as it allows data providers to describe the relationships between their databases. Our demonstration shows that it is easy to create mappings and that VMAP is capable of compiling these visual representations to executable code. Consequently, a declarative visual specification can be automatically transformed to a mapping program. Our VMAP tool demonstration shows the application of such an intelligent system to improve data exchange in the semantic web.

REFERENCES

[1] Tim Berners-Lee, James Hendler, and Ora Lassila, 'The semantic web', *Scientific American*, 29–37, (2001).
[2] Florin Chertes and Ingo Feinerer, 'UMAP: A universal layer for schema mapping languages', in *Proceedings of DEXA2013, August 26–29, 2013*, eds., Hendrik Decker, Lenka Lhotská, Sebastian Link, Josef Basl, and A Min Tjoa, volume 8056 of *LNCS*, pp. 349–363. Springer-Verlag, (2013).
[3] Siegfried Nolte, *QVT—Relations Language*, Springer, 2009.
[4] Siegfried Nolte, *QVT—Operational Mappings*, Springer, 2010.
[5] OMG, *MOF 2.0 Query, View, and Transformation*, 2011. `http://www.omg.org`.
[6] OMG, *Unified Modeling Language Superstructure 2.4.1*, 2011. `http://www.omg.org`.
[7] OMG, *Object Constraint Language 2.3.1*, 2012. `http://www.omg.org`.
[8] Alessandro Raffio, Daniele Braga, Stefano Ceri, Paolo Papotti, and Mauricio A. Hernández, 'Clip: a visual language for explicit schema mappings', in *Proceedings of ICDE2008*, pp. 30–39, (2008).
[9] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall, 'The semantic web revisited', *IEEE Intelligent Systems*, **21**(3), 96–101, (2006).