Creating visualizations through ontology mapping

Sean M. Falconer R. Ian Bull Lars Grammel Margaret-Anne Storey
University of Victoria
{seanf,irbull,lgrammel,mstorey}@uvic.ca

Abstract

We explore how to support the creation of customized visualizations of ontology instance data through the specification of ontology mappings. We combine technologies from the disciplines of software modeling and ontology engineering. The feasibility of our approach is demonstrated by extending an existing ontology mapping tool, CogZ*, to translate ontology mappings into software model transformation rules. The tool uses these transformations to automatically convert domain instance data into data that conforms to a model describing a visualization. After this transformation, a visualization of the domain instance data is generated.*

1 Introduction

Ontologies provide a shared and common understanding about a specific domain [7]. The members of this domain represent instances of the concepts within the ontology. For example, if "Country" is defined as a concept in an OWL (Web Ontology Language) ontology, then "Canada" and the "United States" are potential members or instances of this class. Information visualization helps people explore and understand complex information spaces like ontologies. A variety of tools to explore and navigate ontologies have been developed; Jambalaya [20], OntoRama [8], TGVizTab [1], and OntoSphere [3] are but a few of these tools. However, there has been little work investigating techniques for visualizing the instance data associated with an ontology. Also, when these types of visualizations have been developed, programmers or information visualization experts have typically created the visualization for the ontology's instance data explicitly, as in Mutton *et al.*'s work [13]. This can be a time consuming process and the resulting applications are not easily re-usable or configurable for other ontologies.

In our collaborations with the National Center of Biomedical Ontology (NCBO) [1], we are developing an information visualization toolkit. The NCBO provides an online tool called BioPortal for uploading and sharing biomedical ontologies [2]. The goal of the visualization toolkit is to provide a general means of rapidly developing and deploying ontology-specific instance data visualizations to BioPortal. These visualizations are meant to assist scientists with exploring and understanding their data.

Bull [5] developed a model-based approach to generating visualizations called Model Driven Visualization (MDV). In this work, he represented visualizations (e.g. node-link diagrams, charts) with abstract software models using the Eclipse Modeling Framework (EMF). To use these visualizations, a software data model is "mapped" or transformed to the visual model, and then the corresponding visualization can be generated by executing the transformation.

We recognized that a similar approach could potentially be used for generating visualizations of ontological instance data where the visualizations are constructed by mapping ontology concepts to view model concepts. In this paper, we explore this idea by demonstrating how to leverage existing software modeling tools along with ontology mapping tools to generate highly customized visualizations.

We begin by discussing relevant background about software modeling (Section 2). In Section 3, we describe extensions made to the CogZ mapping tool [10] to support the creation of instance data visualizations. This includes extensions to Bull's existing work by integrating his model transformation techniques into CogZ. By visually specifying mappings in CogZ from the source ontology to the target visualization ontology, data level transformations are generated that can be used to automatically create information visualizations. We demonstrate this process with a case study in Section 4. Following this, we briefly discuss related work (Section 5). Finally, we discuss our future work (Section 6) and conclusions (Section 7).

2 Background

Model Driven Engineering (MDE) is an approach to software development where software is specified, de-

[1] http://bioontology.org/

[2] http://bioportal.bioontology.org/

signed, implemented and deployed through a series of models and model transformations [18, 19]. Software models, while designed and developed to assist engineers with the process of building quality software, share a number of commonalities with ontologies. Software models typically consist of classification hierarchies. For each concept in the hierarchy, its name, properties and relationship to other concepts can be specified. These models are used to help understand the domain, test hypotheses, build prototypes and even generate working systems. Toolkits have even emerged that allow knowledge and software engineers to convert their data between ontology and software model representations. We believe by integrating the concepts from both disciplines, we can leverage strengths of each discipline and enable the rapid specification and generation of customized visualizations of ontology instance data.

To build software using MDE, engineers first capture both domain concepts and their relationships as Platform Independent Models (PIMs). These models are transformed to one or more Platform Specific Models (PSMs), which can be executed. By developing software in this manner, the same PIM can be used on a number of different platforms.

To support this software development methodology, a number of software modeling languages have emerged. Examples of such languages include the Unified Modeling Language (UML) [16] and the Eclipse Modeling Framework (EMF) [4]. In addition to software modeling languages, transformation languages have also been designed to facilitate the transformation from one model to another, or from PIM to PSM. In the software engineering world, the Atlas Transformation Language (ATL) [12] and Query View Transformation (QVT) [15] are widely used.

Bull investigated how information visualizations can be specified as a series of platform specific models [5]. Using these models, he transformed complex software models into visualization models, facilitating the rapid construction of highly customized visualizations. Due to the overlap between software modeling and ontology development, we have started investigating if modeling technologies, and in particular Model Driven Visualization, can be applied to the process of customizing ontology visualizations.

3 The Tool

```
rule mappingRule {
 from IN : shrimpbib!User
 to OUT : distinct nodelink!GraphNode
      nodeLabel <- IN.nickname,
      color <- 'Orange'
}
```

Listing 1: Example of User to GraphNode ATL mapping rule.

COGZ is an ontology mapping perspective integrated as a plugin to the ontology management suite PROMPT [14].

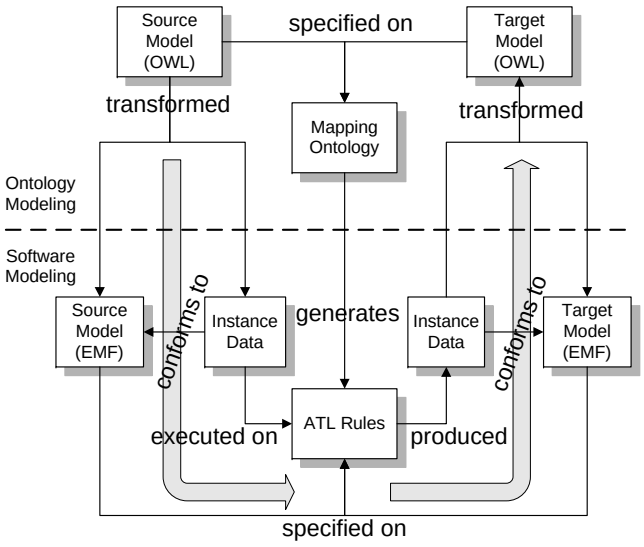

Figure 1: COGZ instance data mediation architecture.

COGZ supports a visual representation of mappings and a drag and drop interface for creating mappings. Previous research indicates that this interface helps users create mappings faster and with more accuracy than the default PROMPT view [10]. The mappings are recorded in an alignment ontology, which stores mapping relationships as instance data. To create customized visualizations of ontology instance data, we extended COGZ by combining technologies from software modeling and ontology engineering.

The extended COGZ architecture for ontology instance data mediation is shown in Fig. 1. We use the existing EMF visualization models introduced by Bull [5]. To make these usable within COGZ, we integrated the Eclipse Ontology Definition Metamodel (EODM) [3] project, which supports automatic conversion between EMF and OWL models. To support data integration between the source and target models, we developed an ATL rule writing library that converts the specified source to target mappings that are stored in the mapping ontology into transformation rules that describe how to convert data between the two models. Since ATL only works with software models, we automatically convert the source and target OWL models into EMF and use the EMF versions for data transformation. The conversions and generating process happen automatically and are hidden from the user.

The ATL rule writer library supports three types of mappings: concept to concept, data type property to data type property, and reference property to reference property mappings. We extended COGZ to support a property value editor for concept to concept mappings. This allows the user to assign values to a target concept's property for a spe-

[3] http://eclipse.org/modeling/mdt/?project=eodm

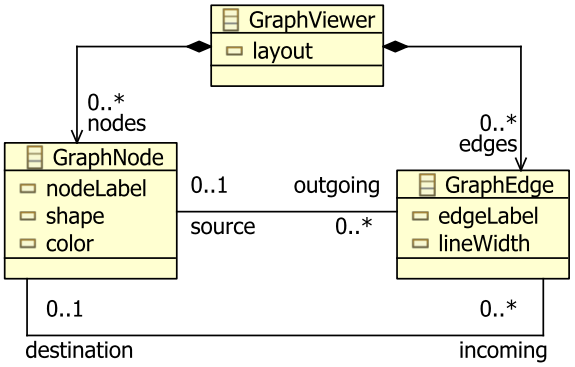

Figure 2: A simplified representation of graph viewer model.

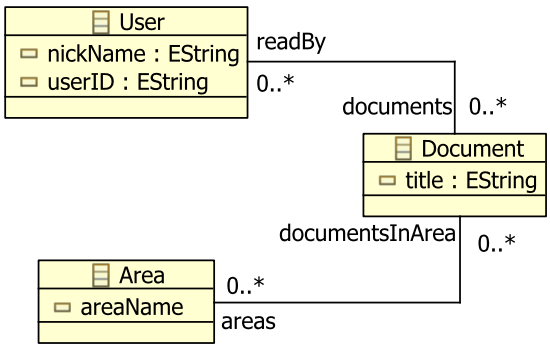

Figure 3: A simplified representation of the domain ontology.

cific mapping. This is necessary for manipulation of core visual elements. For example, assume we map a concept *User* from a source ontology to the concept *GraphNode* in a target visualization ontology. The *GraphNode* concept represents the node in a node-link graph visualization and we want all users in our ontology to be represented by a node. The *GraphNode* may have properties such as *color, size, shape, etc.* It is important that the end-user be able to customize the visualization to their needs, such as making the nodes in the graph that represent a user be colored "Orange". The property editor allows the end-user to assign this value to the property and this will automatically be associated with the corresponding mapping rule, see Listing 1.

Finally, we enhanced CogZ to allow the end-user to execute their transformation rules and generate the result as a visualization. By executing this, the instance data stored and associated with the source OWL model is converted into target model elements via the specified ATL rules. The MDV rendering engine is then used to display the visualization. This engine is capable of rendering any visualization conforming to one of the pre-defined models. Models exist for *node-link diagrams, charts, maps* and *treemaps*, among others. The simplified view of the *node-link* model is shown in Fig. 2. In the next section, we discuss a specific example demonstrating how this mapping process works.

4 Case Study

```
rule mappingRule {
 from IN : shrimpbib!Document
 to OUT : distinct nodelink!GraphEdge
    foreach( e in IN.computeCrossProduct() ) (
       destination <- e.domain,
       source <- e.range
    )
}
```

Listing 2: Document to GraphEdge ATL mapping rule.

To demonstrate the feasibility of this approach, we chose to regenerate an ontology instance data visualization discussed by Allen [2] and Bull [5]. Allen developed an ontology to describe researchers, research areas, research documents, and the relationships between these concepts (Fig. 3). She programmed a specific graph visualization that displayed researchers and the documents they had read. Bull demonstrated how to generate the same view using MDV. He converted Allen's ontology into an EMF model and then manually constructed ATL rules to transform the source model into his view model (Fig. 2). In this case study, we demonstrate how to generate a similar visualization by simply dragging and dropping mappings in CogZ between Allen's source ontology and Bull's visualization model.

The desired visualization is a graph-based view that connects researchers to their area of expertise based on the papers they have read. In the model, there is no direct association between a *User* and an *Area* (this information can only be derived by looking at the papers each researcher has read). In the following, we demonstrate how a mapping can be used to describe the desired visualization.

To begin, CogZ is loaded and Allen's model is specified as our source ontology and Bull's node-link model as our target. We want to represent both users and areas as nodes and link these based on representing documents as edges. To do this, we drag mapping lines from the source concept *User* to the target concept *GraphNode*, *Area* to *GraphNode*, and finally *Document* to *GraphEdge*. We also assign several data type property mappings to configure the labels for the *GraphNode* mappings. We map the *User* property *nickname* to *GraphNode*'s *nodeLabel* property and *Area*'s *name* property to *nodeLabel*. We also assign the color "Orange" to the *User* to *GraphNode* mapping using the property editor. An ATL rule is automatically created for each concept to concept mapping, similar to Listing 1. The rule describes which source class is being mapped to a target class and which properties should be mapped between the two con-

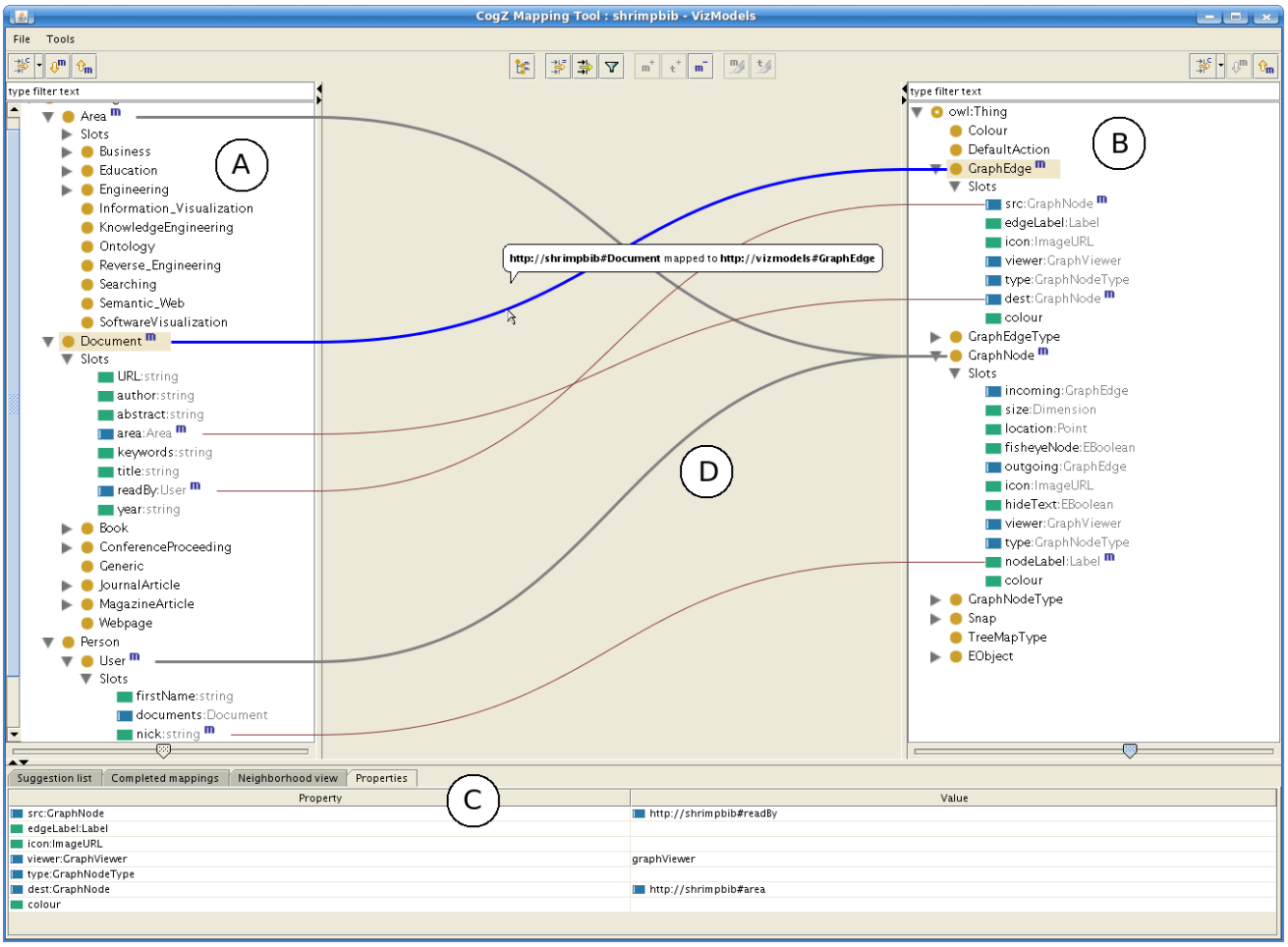

Figure 4: Mappings from domain ontology to visualization ontology. (A) shows the source domain ontology, (B) shows the target view ontology, (C) shows the property editor for a mapping, and (D) shows the visual representation of mappings. Thick arcs represent concept to concept mappings and thin arcs represent property mappings.

cepts. Each of these mappings are simple one-to-one mappings and the rule generation is straight-forward.

The final mappings necessary to construct the view are the reference mappings from a *Document*'s reference to the areas it is contained in and the reference to the user's that have read that particular document. We map the *area* reference property to the *GraphEdge*'s *destination* property, while the *readBy* property is mapped to the *GraphEdge*'s *source* property. The complete mapping as represented by CogZ is displayed in Fig. 4.

The reference property mappings have different cardinality as both the *area* and *readBy* properties can have multiple values while the *destination* and *source* represent a single instance. This change in cardinality is automatically detected by our ATL rule writing engine and resolved by computing a cross-product between the users mapped to *source* and areas mapped to *destination*. An example of the rule generated for the *Document* to *GraphEdge* mapping is shown in Listing 2.

The mappings necessary to generate the view are now specified and the transformation rules can be executed. The resulting visualization is shown in Fig. 5.

This small case study has demonstrated the feasibility of using model driven visualization as an approach for customizing visualizations for ontology instance data.

5 Related Work

In this section we discuss research related to ontology instance data visualization and instance data mediation.

5.1 Ontology Instance Data Visualization

Gilson *et al.* describes an approach that uses ontology mapping techniques and probabilistic reasoning to automatically create visualizations for domain specific data from the

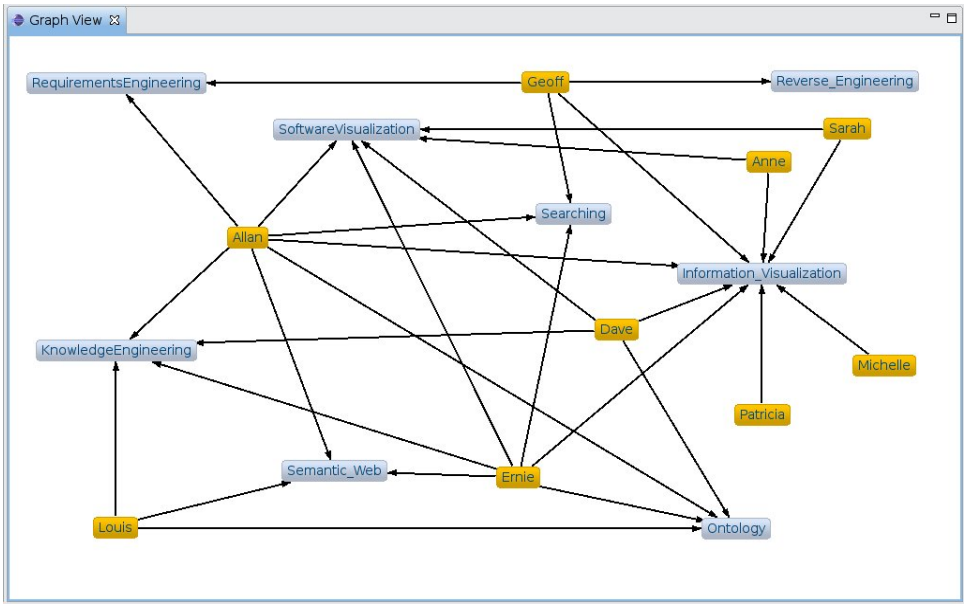

Figure 5: Visualization generated by mapping rules. Researchers are connected to their areas of expertise based on documents they have read from those areas.

web [11]. Three ontologies (domain, visual representation, and semantic bridging ontology) are used to capture expert knowledge. The data from the web was mapped to the domain ontology, and the semantic bridging ontology determined how the domain ontology was mapped to the visual representation ontology. Using those mappings and visualization toolkits, a set of rated visualizations was calculated. The mappings were represented by the semantic bridging ontology, which the authors assumed to be created by experts. In comparison with our approach, we use a semi-automatic mapping procedure to create mappings between the domain ontology and the visualization ontology.

Cammarano *et al.* addressed the problem of visualizing semi-structured, potentially incomplete ontology data by automatically mapping it to a visualization [6]. Their algorithm relied on a visualization specification and a set of ontology entities. The algorithm first generated and ranked relevant general paths. Second, data instances for the attributes and each ontology entity were searched using these paths and some additional heuristics. In contrast, our approach is semi-automatic and enables humans to create and control the mapping from the ontology to the visualization.

5.2 Instance Data Mediation

There has been a variety of research on transforming ontology instance data from a source to target ontology based on mappings. As of yet there is no standard available. Euzenat *et al.* proposed using SPARQL queries to transform RDF triples between ontologies [9]. This approach is attractive as it works directly with the concepts and relationships of the ontologies and SPARQL syntax is designed for extracting and translating data. However, as Euzenat *et al.* indicate, SPARQL is missing a number of needed transformation constructs to fully support ontology data mediation.

Another promising approach is described by Scharffe *et al.* [17]. The authors discuss the design and requirements of a mapping language for ontology mediation. The language they propose uses an XML syntax, and a Java API has been provided for parsing and serializing the methods for mapping. Again, like the previously mentioned approach, the design of this language is still preliminary and it does not support many required features for ontology mediation, such as functional mappings (e.g. splitting and merging).

Due to these limitations, we chose to rely on existing work from software modeling for transforming/integrating data from our domain models to our visualization models. We chose ATL for constructing our transformation rules as this was successfully used in the Model Driven Visualization work that we are extending and libraries exist for automatically executing ATL rules.

6 Future Work

Although we are able to create instance data visualizations using the current CogZ extensions, there are cases that are currently not supported without manual manipulation of the ATL rules. For example, there is no visual support within CogZ to concatenate or split property values, however, these are supported within ATL. We plan to

continue to enhance support for different types of mappings both within CogZ and the ATL rule writing engine we have developed. Also, further extensions are necessary for handling other types of cardinality changes during mapping. Our current ATL rule writing library supports automatic detection when mapping a many-to-many relation to a one-to-one, but other cardinality differences may occur.

Also, moving forward, we would like to generate web-based visualizations. Currently, the supported visualizations are only available as a native Java application. We are working on developing a web-based version of CogZ to integrate with the NCBO's BioPortal application and we plan to investigate supporting instance data visualizations for the web within this system.

Another opportunity for future work is with the construction of a mapping algorithm tailored to data integration. Prompt's default algorithm does not perform well for this type of mapping scenario. Prompt relies primarily on lexical similarities between the models, however, with this particular application, it is unlikely that the source model will have concept names lexicographically similar to the view model. We believe a more appropriate algorithm would put a higher priority on data type comparison once some concept to concept mappings are manually generated.

7 Conclusion

We extended the CogZ ontology mapping tool to support the generation and customization of ontology instance data visualizations. These extensions involved combining techniques from both software modeling and ontology modeling. We extended work from Model Driven Visualization by incorporating existing EMF visualization models as our target OWL view models. We integrated CogZ with the MDV toolkit for executing ATL rules and generating visualizations. We demonstrated the feasibility of this approach through a case study where we recreated a previously developed ontology instance data visualization simply by specifying mappings from a source to target ontology.

We believe this approach is very promising for the specification and generation of visualizations. This approach gives us the generality we require for the visualization toolkit we are developing for BioPortal. Potentially, mappings that specify a visualization could be shared in BioPortal and end-users could modify the mappings and property values to create their own custom visualizations.

References

[1] H. Alani. Tgviztab: An ontology visualisation extension for protg. In *Knowledge Capture (K-Cap'03), Workshop on Visualization Information in Knowledge Engineering*, 2003.

[2] M. M. Allen. Empirical evaluation of a visualization tool for knowledge engineering. Master's thesis, University of Victoria, 2003.

[3] A. Bosca, D. Bomino, and P. Pellegrino. Ontosphere: more than a 3d ontology visualization tool. In *SWAP, the 2nd Italian Semantic Web Workshop*.

[4] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. J. Grose. *Eclipse Modeling Framework*. Addison Wesley, 2003.

[5] R. I. Bull. *Model Driven Visualization: Towards a Model Driven Engineering Approach for Information Visualizations*. PhD thesis, University of Victoria, 2008.

[6] M. Cammarano, X. L. Dong, and J. Talbot. Visualization of heterogeneous data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1200–1207, 2007.

[7] Y. Ding and S. Foo. Ontology research and development. part 2 - a review of ontology mapping and evolving. *Journal of Information Science*, 28(5):375–388, 2002.

[8] P. W. Eklund, N. Roberts, and S. P. Green. Ontorama: Browsing an rdf ontology using a hyperbolic-like browser. In *The First International Symposium on CyberWorlds*, 2002.

[9] J. Euzenat, A. Polleres, and F. Scharffe. Processing ontology alignments with sparql. *cisis*, 0:913–917, 2008.

[10] S. M. Falconer, T. Yamauchi, and M.-A. Storey. Cogz: Evaluating cognitive support for semi-automatic terminology mapping. Under review, CHI'09.

[11] O. Gilson, N. Silva, P. W. Grant, and M. Chen. From web data to visualization via ontology mapping. In *Proceedings Eurographics / IEEE VGTC Symposium on Visualization*, 2008.

[12] F. Jouault and I. Kurtev. Transforming Models with ATL. *Lecture Notes in Computer Science, Springer Berlin / Heidelberg*, 3844:128–138, 2006.

[13] P. Mutton and J. Golbeck. Visualization of semantic metadata and ontologies. In *IV '03: Proceedings of the Seventh International Conference on Information Visualization*, page 300, Washington, DC, USA, 2003. IEEE Computer Society.

[14] N. F. Noy and M. A. Musen. The PROMPT suite: Interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.

[15] OMG/QVT. MOF – Query / View / Transformation (QVT), 2007. available at http://www.omg.org.

[16] OMG/UML. Unified Modeling Language (UML), version 2.1.1, (Formal/2007-02-03), 2007. available at http://www.omg.org.

[17] F. Scharffe and J. de Bruijn. A language to specify mappings between ontologies. In *IEEE SITIS'05*, 2005.

[18] B. Selic. The Pragmatics of Model-Driven Development. *IEEE Software*, 20(5):19–25, 2003.

[19] B. Selic. Model-Driven Development: Its Essence and Opportunities. In *Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 313–319, 2006.

[20] M.-A. Storey, N. F. Noy, M. Musen, C. Best, R. Fergerson, and N. Ernst. Jambalaya: an interactive environment for exploring ontologies. In *IUI '02*, pages 239–239, 2002.