# Semantic Annotation and Conflict Analysis for Information System Integration

Henning Agt[1], Gregor Bauhoff[2], Ralf-D. Kutsche[1], Nikola Milanovic[2], and Jürgen Widiker[2]

[1] Technische Universität Berlin
{hagt,rkutsche}@cs.tu-berlin.de
[2] Model Labs GmbH
{nikola.milanovic,gregor.bauhoff,juergen.widiker,}@modellabs.de

**Abstract.** One of the main challenges in software integration is to overcome interface incompatibilities. Integration specialists are confronted with different semantic, structural, behavioral, communication and property mismatches. We developed an MDA-based theoretical framework to address these aspects by systematically modeling integration projects at the CIM, PSM and PIM levels in order to achieve automatic analysis, code generation and integration. However, small and medium enterprises (SMEs) frequently have problems to apply such methodology due to the high complexity of general purpose modeling tools like UML tools, and of ontology editors in the context of semantic enhancement of MDA-based tools. Our development of the MBIF tool set (Model-Based Integration Framework) takes care of this 'conditio-sine-qua-non' requirement in SMEs with available staff under daily business pressure. In this paper we describe data- and function-oriented semantic enrichment of integration models and the semantic conflict analysis which discovers matchings between model elements with respect to semantic process requirements. The solution of MBIFs semantic annotation and integration tool set was designed under the 'small is beautiful' philosophy.

## 1 Introduction

Integration of complex and heterogeneous distributed IT-systems is one of the major problems and cost-driving factors in the software industry today [1]. Therefore, there is an increasing need to systematically address integration in accidental architectures that have grown over time in an uncontrolled manner in heterogeneous enterprise environments. This problem was addressed in a three-year research activity started in 2007 of our BIZYCLE [3] group at Technische Universität Berlin, in collaboration with six small and medium enterprises from the region.

[3] This work is partially supported by the Bundesministerium für Bildung und Forschung BMBF under grant number (Förderkennzeichen) 03WKBB1B.

One of the main potentials of semantic methods, such as ontologies, is facilitating system integration. However, their application is neither straightforward nor possible in all cases. The common approach to software integration today are various SOA integration platforms, based on ESB engines which enact integration process flow, specified and executed in BPEL. Frameworks and languages for semantic annotation of Web services exist, but scenarios that do not use Web service wrappers, such as data-stream transfer/processing, are implemented using e.g., SQL because of performance reasons. This fact significantly limits the applicability of semantic SOA extensions. Other approaches for system integration, such as those based on schema matching [2], Extract-Transform-Load (ETL) tools, or message-oriented middleware platforms (such as WebSphere MQ, BizTalk or TIBCO) do not offer functional semantic annotation or reasoning and only rarely support rudimentary data annotation.

Furthermore, ontologies are rarely used in industrial integration practice, at least by the majority of SMEs, as they appear rather complex from the logical background, and from the appearance of existing tools. In general, for these enterprises, even classical general purpose modeling tools are hard to introduce into daily business, but their co-existence with ontology modeling in different tools clearly causes capacity and skill problems within the limited staff.

Knowledge about semantics of business data and functions thus remains in the heads of business analysts and is rarely, if ever, adequately documented and transported to developers that are implementing the integration solution. This is a constant source of errors, frustration and costly workarounds. For these reasons BIZYCLE consortium (www.bizycle.de) was formed to investigate in large-scale the potential of MDA-based software integration methodologies, including semantic concepts and ontologies, in order to develop a theoretical basis and affordable tool support and practical applicability for SMEs in arbitrary industrial domains (for the concrete project following the lines of our industrial partners: health, production/logistics, facility management, and publishing).

The essence of the BIZYCLE integration process [3–6] is in providing multi-level integration abstractions and making the process partially automatic. The levels of abstraction are the computation independent model (CIM), platform specific model (PSM) and platform independent model (PIM). The CIM level captures the integration business process. This model is refined at the PSM level where properties of interfaces realizing an integration scenario are described. All PSM models are transformed to the PIM level, which represents a common abstraction level which is used to discover integration conflicts (incompatible interfaces). Based on results of the conflict analysis, appropriate connector (mediator) model and code for the connector component are generated.

In this paper we present semantic enrichment and semantic annotations of BIZYCLE models (Figure 1). Ontologies, as carriers of semantic knowledge, are represented as models. Their elements, also known as concepts, are assigned to model elements that describe integration process at the CIM (business process) and PSM/PIM (interfaces) levels. Thus, model elements are given domain-specific meaning. Afterwards, semantic annotations are processed in the model

checking procedure (we call it integration conflict analysis) and semantic dependencies (mappings) between model elements are derived. Later, they are used for identification of semantically compatible model elements and code generation. We focus here on investigating semantic compatibility, more information about the subsequent code generation can be found in [7].

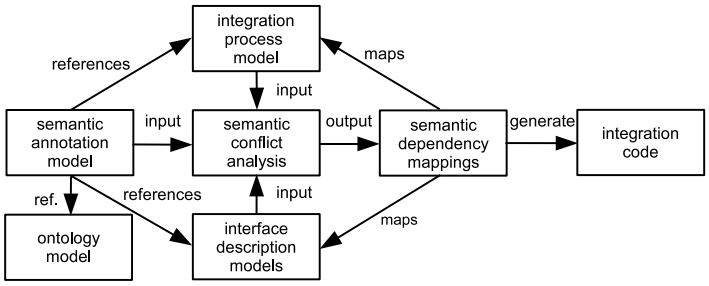


**Fig. 1.** Semantic enrichment of integration models

## 2 Related Work

System integration with respect to semantic properties relates to several research areas: semantic annotation [8], ontology application [9], information integration, schema matching [2] and service-oriented integration [10]. Types of semantic conflicts in database and information systems integration have been classified in [11, 12]. A survey of ontology-based information integration with focus on roles and development of ontologies and integration at the ontology level is given in [13]. It observed the lack of mappings and sophisticated methodologies that we address in our approach.

Semantic annotation of enterprise models has been addressed in [14]. They describe different annotation types with regard to model interoperability. This general concept is similar to our approach but does not focus on specific models nor on realization of the annotations. Detection of semantic conflicts in models in the context of version control systems is described in [15, 16] (ModelCVS project). Conflicts between model versions are addressed while we consider conflicts between different models and abstraction levels.

Many existing integration tools rely on ontologies represented in OWL [17] and on Web Service technology, such as the METEOR-S framework [10]. In such an environment annotation and analysis is done at ontology and XML language level and reasoning tools like Pellet [18], Racer [19] (OWL-based) and XSB, SWI-Prolog [20] (Prolog-based) are used. To overcome the gap between ontology space and model-driven engineering, which we also address, in [21] an ontology-aware MDE platform architecture with semantic enrichment of models is presented. In contrast to our approach it addresses conflicts between models on

different MOF-levels. Similar to our approach, semantic enrichment of models is proposed in [22]. The main goal of their OntMT tool is ontology-based creating of model transformation for interoperability between modeling languages while we focus on semantic dependency analysis and resolution of interface integration conflicts.

## 3 Semantic Annotation

Semantic annotation originates in the Semantic Web community that describes associations of ontology elements to service interfaces, documents and web resources [8, 14]. It enables machine-based processing of these artifacts. We adopted this paradigm to semantically annotate model elements of integration project models at the CIM level and interface description models at the PSM and PIM level enabling automatic annotation analysis for element matching. In this section we describe two metamodels that realize semantic annotation within our integration framework: the semantic metamodel (SMM) for expressing ontologies that contain domain knowledge of integration projects and the intermediate annotation metamodel (AMM) that links the semantic metamodel with respective metamodels of our framework, using a model weaving approach [23].

The semantic metamodel (Figure 2) defines an abstract DSL syntax for ontology representation. The SMM provides semantic concept and predicate metaclasses to build statements (RDF-like triples) consisting of subject, predicate and object (e. g., *CustomerName IsA Name*). In the context of integration artifacts we distinguish between knowledge representation of data (`DomainObject`) and representation of functionality (`DomainFunction`). Relations between ontology concepts are modeled with predefined predicates: `IsA` (generalization), `IsInputOf`, `IsOutputOf` (data processing for functions), `Has` (containment), `IsListOf` (data sets) and `IsEquivalentTo` (equivalence). These types are later exploited by the conflict analysis algorithm to infer matching model elements. Our framework allows creation of own ontologies with an Eclipse GMF-based editor [24] or usage of existing OWL ontologies [25].

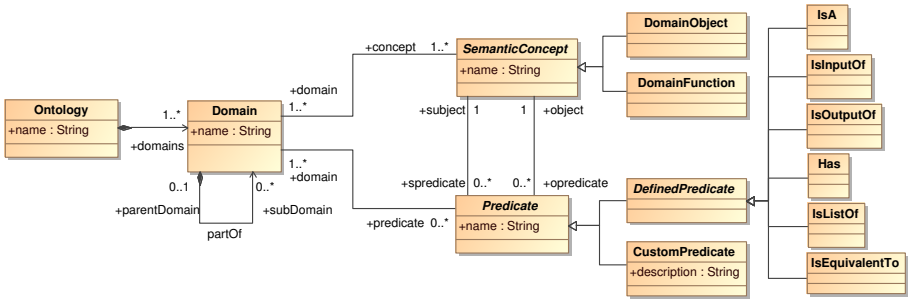


**Fig. 2.** Semantic Metamodel for conceptual modeling

The annotation metamodel establishes links between the semantic metamodel and all other metamodels of our framework (CIM-, PSM- and PIM-level metamodels), in order to annotate model elements with ontology concepts. Model elements are thus assigned domain-specific meaning. It is then possible to compare integration artifacts more precisely and identify semantically equivalent or compatible elements. The features of the model element annotation are given in Figure 3. It shows part of the annotation metamodel at the platform-specific level (providing annotations for J2EE Enterprise JavaBeans interface models). The AMM offers data-oriented and functional annotation (1) with respect to both knowledge representation types of the semantic metamodel (2). Data-oriented annotations are used to describe all model elements that represent data at different levels of abstraction (3). Functional annotations describe the meaning of computational and processing behavior (4). Logical operators link two or more annotations for combination (5).

Use of logical operators and cardinality is constrained to five annotation types: *Single representation* links one model element to one ontology concept (e. g., ZipcodeParameter → ZipcodeConcept). *Containment* links one coarse grained model element to multiple concepts (e. g., AddressParameter→{Street, Town,PostalCode}). *Composition* links multiple model elements to one coarse grained concept (e. g., {Firstname,Lastname} → Name). *Multiple and alternative representation* combine annotations with the `AND` and `OR/XOR` operators respectively (e. g., IDcolumn→((Identifier) && {FirstName,LastName}).

We currently support annotation of integration project models at the CIM level (abstract data descriptions and integration process) and platform-specific interface annotations for J2EE applications, relational database systems, SAP R/3 ERP systems, Web Services and XML structured flat files [24, 6]. All PSM level models are transformed to the PIM level (a common abstraction level with a single metamodel) to discover integration conflicts. Thus, annotations given at the PSM level are transported to the PIM level using model transformations.

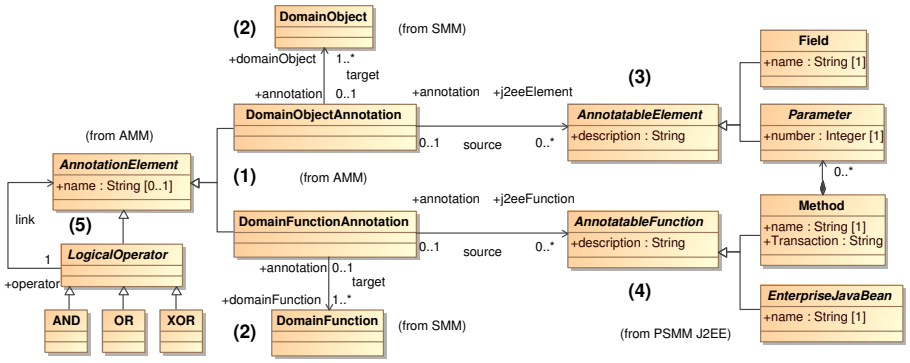


**Fig. 3.** Semantic data and function annotation (excerpt)

## 4 Semantic Conflict Analysis

In this section, semantic conflict analysis is introduced. It is an algorithm for checking semantic compatibility of model elements based on semantic annotations presented in the previous section. We describe the general approach of performing semantic matching of corresponding models by establishing equivalent or compatible model elements and point out two important features of the algorithm: **annotation comparison** based on the five annotation types and **reasoning rules** based on the defined predicates of the semantic metamodel. Furthermore we present how **semantic mappings** (matching results) are represented as models as well.

**General Approach:** Model element matching is performed at two levels: CIM-PIM, where requirements of an integration project (CIM) are matched against technical interfaces (PIM), and PIM-PIM, where semantic compatibility of the underlying systems which implement an integration scenario is investigated. We distinguish between following types of *requirements*: business object import/export requirement (the need to import/export data), business function requirement (necessary functionality) and interface element requirements (e. g., parameters of methods that have to be supplied for invocation). Business requirements are generated from definitions specified by integration specialist at the CIM level. Interface element requirements are created from interface description models at the PIM level during the conflict analysis. Results of the semantic conflict analysis are *requirement mappings*: mappings of business object requirements to the underlying interface elements, mappings of required to provided interface elements and mappings of functional requirements to interfaces. Mappings thus mark the compatible (matched) elements and are used for subsequent conflict analysis phases and for connector code generation [26]. The basic algorithm structure is given in Figure 4. Integration project models at the CIM and PIM level, annotation models and ontology are not shown in the picture, because they are used in almost all steps of the algorithm.

The process is initiated by creating business requirements (1). Data-oriented and function-oriented requirement mappings are created separately, based on given semantic annotations (2). Interfaces that fulfill business requirements are checked for required import elements that have to be supplied for invocation (3). For the import element requirements, appropriate export elements are determined (4). If all requirements are not fulfilled (mapped), the algorithm employs semantic reasoning (5) on the annotations and ontology to handle the unmapped requirements. Requirement identification and mapping are performed recursively to check whether newly generated mappings introduce additional requirements. Data mappings are cross-checked against functional mappings (6) to eliminate possible incorrect solutions in which interface elements match, but an interface provides different functionality than required at the CIM level. Furthermore, integration specialist has the option to manually map requirements that are still unfulfilled (7). If multiple solutions are found, a selection can be made or alternatives can be kept for further analysis phases.

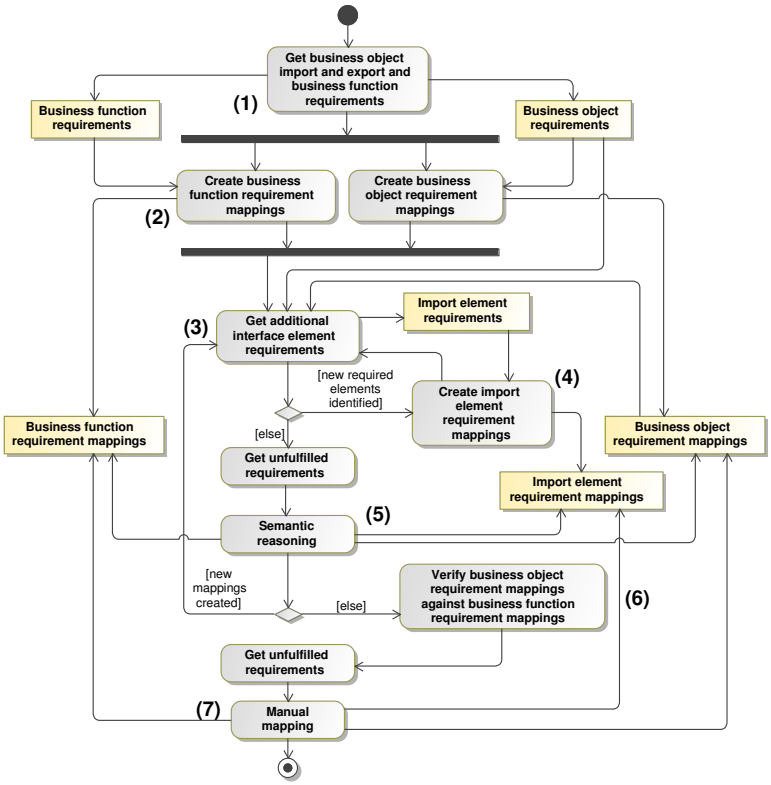


**Fig. 4.** Semantic Conflict Analysis Algorithm Overview

The algorithm is capable of checking several types of semantic conflicts, such as checking transfer of simple or structured data objects, performing interface selection and call order, integrating connector functions, and supporting multiple semantic annotations, and simultaneously checking functional and data annotations. The entire algorithm description, as well as exemplary scenarios that demonstrate these capabilities can be found in [27]. Here we focus on two algorithm features: *annotation comparison* that matches model elements based on the five annotation types (section 3) and *reasoning rules* to deduce compatible model elements if comparison is insufficient. The first feature is used in activities (2,4) from Figure 4, and the second is used in the semantic reasoning activity (5).

**Annotation Comparison:** Model elements are compared based on their annotations rather than names, as in schema matching approaches. In order to create appropriate requirement mappings for a given source element, target elements are searched that either have the same or semantically equivalent annotation. Figure 5 illustrates annotation types and how source (filled squares) and target (white squares) elements can be matched in relation to the ontology concepts (ellipses). We will exemplary describe analysis of the containment an-

notation type (this is the fifth column in Figure 5, where containment annotation is the source).

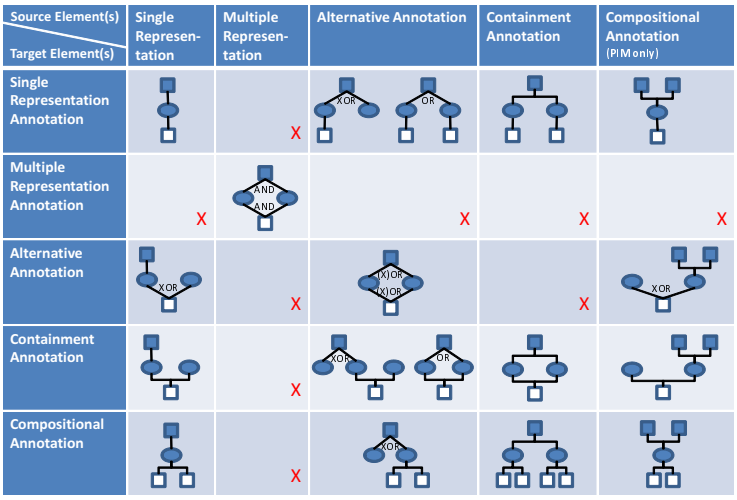


**Fig. 5.** Model element matching depending on annotation types

Let us assume in our example that customer data has to be transferred between two systems. At the CIM level an integration specialist would include an *Address* object to be transfered, with the containment annotation *Address*→ {*Name,Street,Town,PostalCode*}. The algorithm tries to find appropriate elements of interface description models at the PIM level that fulfill requirements expressed with above containment annotation. A direct match with an equal containment annotation is obvious, e. g., assume that *AddressParameter* → {*Name,Street,Town,PostalCode*} is an import parameter at the PIM level, thus being a direct match for the business object (same annotation: import requirement fulfilled). Alternative annotation does not directly match, because containment demands the presence of all elements, neither does the multiple representation annotation, because it implies polymorphy. Single representation matches containment if a set of exporting PIM interface elements can be found that has a union match and compositional annotation matches containment if multiple PIM elements are annotated together with an equal set of ontology concepts. The example of a composite match are three exporting interface elements: *CustNameParam*→*Name*, *StreetParam*→*Street* and *TownParam*→ {*PostalCode, Town*}. The semantic conflict analysis algorithms also supports mixed and nested combinations (only partially shown in the matrix).

**Reasoning Rules:** The second algorithm feature we describe are reasoning rules which are applied to semantic annotations, ontology concepts and predicates, in case that annotation comparison from previous section is insufficient. The rules utilize the defined predicates of the semantic metamodel (Section 3).

Six rules are defined for deducing semantic relationships. Consider the following example: an import parameter *NameParam* annotated with *NameParam*→*Name* has to be provided for interface invocation. Rule 1 tries to locate all domain functions $F_1, ...F_n$ and their respective implementations, whose outputs are annotated with $Name$ `IsOutputOf` $F_1, ...F_n$. Rule 2 finds all interface elements $I_1, ..., I_n$ whose export elements are annotated with $\iota_1, ...\iota_n$ `IsListOf` $Name$, and performs list iteration operation on $I_1, ...I_n$. Rule 3 considers specialization and finds all interface elements $I_1, ...I_n$ whose export elements are annotated with $\iota_1, ...\iota_n$ `IsA` $Name$. Rule 4 evaluates containment within ontology and processes all sets of domain objects referenced with $Name$ `Has` $\iota_1...\iota_n$. If for each $\iota_i$ an exporting interface element $I_i$ can be found, $NameParam$ is composed of $I_i$. Rule 5 is the equivalent to rule 4, except that it uses functional instead of object composition. Finally, rule 6 takes equivalence information into account ($\iota_1...\iota_n$ `IsEquivalentTo` $Name$).

**Semantic Mappings:** The results of the semantic conflict analysis algorithm are kept in a conflict analysis model with *requirement mappings* that link the abstract data and function definitions of the integration process (CIM level) to semantically compatible concrete methods and parameters that actually provide and require the data and functionality (PIM level). They also connect matching provided and required interface parts. Recall the above-mentioned example of the address object at the CIM level with annotation *Address*→{*Name, Street,Town,PostalCode*}. Mappings then connect *Address* to a matching interface that exports such an address and to an import interface, respectively. Both interfaces are linked as well. The mappings make sure that integration requirements are met by the underlying systems and compatibility of the system's interfaces is established. Mappings that have been deduced by reasoning rules usually result in data manipulation (e. g., aggregating a first and last name). That information is stored along with the mappings. The conflict analysis model and referenced integration models are further transformed into an executable form and finally result in a connector. Connector generation and execution is out of scope of this paper and can be found in [26, 7].

## 5 Implementation of the Semantic Conflict Analysis

Semantic conflict analysis is implemented as part of the BIZYCLE Model-Based Integration Framework (MBIF) based on Eclipse. Our implementation consists of the following components: annotation editor, Prolog engine, Prolog rule base, model to Prolog transformation, conflict analysis controller and mapping editor (Figure 6). The *annotation editor* is implemented as Eclipse properties view to manage ontology references, create annotations and edit annotation models using the Eclipse Modeling Framework API. We use SWI-Prolog [20] as reasoning *engine*, that offers a Java API for integration into our framework. All structural features of the metamodels (classes, associations, generalizations, etc.) and major part of the algorithm constitute the *Prolog rule base*. To enable semantic analysis on models, all Ecore model artifacts are transformed to Prolog facts

by model-to-code transformation and build the knowledge base for reasoning. The *conflict analysis controller* operates the algorithm phases and processes the Prolog engine's results to create mapping models. Finally, the *mapping editor* is implemented as Eclipse plug-in to edit resulting mapping models and to manually add mappings that could not be determined automatically.

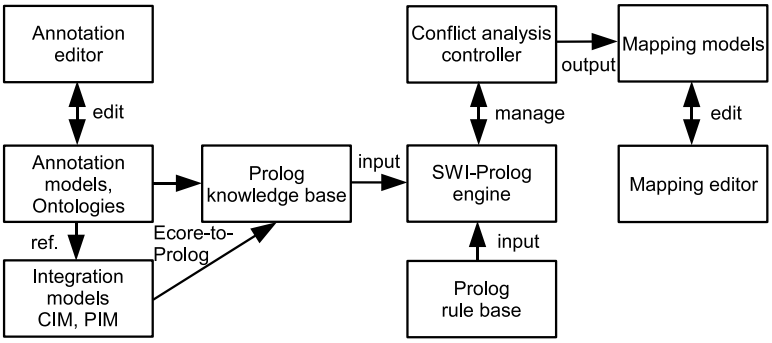


**Fig. 6.** Semantic Conflict Analysis implementation

## 6 Conclusion

In this paper, the BIZYCLE approach to semantic enrichment of model elements was presented, describing business processes (CIM level) and technical interfaces (PSM and PIM levels) which realize software integration. Semantic annotation is only the first step in the integration conflict analysis process, other parts being structural, behavioral, property and communication analysis. Together, they form a comprehensive MDA-based and tool-supported software integration process. Our prototypical MBIF tool suite has entered industrial validation phase. Based on the feedback from BIZYCLE industrial partners, several additional features about the proposed semantic enrichment of software integration process can be made.

Even the market leaders in software integration practice have very low exposure to formal modeling of semantic knowledge using e.g., ontologies (RDF, OWL-S or SA-WSDL). Domain specific knowledge is mostly captured informally and transferred using text files and unstructured free-form diagrams. Furthermore, common ontology tools are very complex, require – in the economy of SMEs – an unaffordable learning curve. As ontology tools and (mainly UML based) general purpose modeling tools today are not operated within one software development framework, there is no convenient way of code generation. Hence, a clear benefit of integrating ontologies into software integration process is difficult to identify.

The compact DSL presented in this work overcomes these problems, and meanwhile has been accepted by integration domain experts in BIZYCLE related

companies and the industrial domains mentioned: health, production/logistics, facility management, and publishing. It was evaluated by them as overall improvement when compared with existing tools. The ability to add semantic annotations at different abstraction levels improved communication within integration teams, as well as provided formal basis for matching requirement specification (e.g., at the CIM level) and integration solutions (e.g., at the PSM/PIM level). Code generation methods (see [7]) improved the perception of benefits achieved by using MDA. It was possible to derive quantitative improvement factors which showed that capturing semantic annotations at the business process level and transferring them all the way to the code level improves not only the speed but also quality of complex integration projects by preventing costly semantic mismatches much earlier in design/modeling phase. Finally, the possibility to annotate legacy systems for which we provide metamodeling support was very important, as it allowed to extend the reach of the Semantic Web technologies which are mostly restricted to Web service-based applications.

However, several problems were also identified. With large models, e.g., complex ERP data models containing many thousand nodes, automatic semantic annotation becomes a must. We are currently working on a tool which can automatically extract semantic annotations for the Oracle, SAP and Web Service platforms to unify them under our ontology metamodel, and will start to set up a theoretical basis for that under the new project frame of BIZWARE, starting in summer 2010. We experienced limited use of metadata standards (such as HL7 or Onix) and lack of their transfer to standardized domain ontologies in all sectors in which our partners work. We focus on further research in (semi-) automatic ontology generation, ontology merging, and on the integration of metamodel knowledge and ontology knowledge representations [28]. Another problem to be addressed in our follow-up project BIZWARE will be support for the cross-sector exchange or combination of semantic knowledge, using a new approach towards orthogonal combination of domain-specific languages (DSLs).

## References


1. Pulier, E., Taylor, H.: Understanding Enterprise SOA. Manning (2006)
2. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. VLDB Journal **10**(4) (Jan 2001) 334–350
3. Kutsche, R., Milanovic, N.: (Meta-)Models, Tools and Infrastructures for Business Application Integration. In: UNISCON 2008, Springer Verlag (2008)
4. Kutsche, R., Milanovic, N., Bauhoff, G., Baum, T., Cartsburg, M., Kumpe, D., Widiker, J.: BIZYCLE: Model-based Interoperability Platform for Software and Data Integration. In: Proceedings of the MDTPI at ECMDA. (2008)
5. Milanovic, N., Kutsche, R., Baum, T., Cartsburg, M., Elmasgunes, H., Pohl, M., Widiker, J.: Model & Metamodel, Metadata and Document Repository for Software and Data Integration. In: Proc. ACM/IEEE 11th MODELS. (2008)
6. Agt, H., Bauhoff, G., Cartsburg, M., Kumpe, D., Kutsche, R., Milanovic, N.: Metamodeling Foundation for Software and Data Integration. In: Proc. 8th International Conference on Information Systems Technology and Applications (ISTA). (2009)

7. Milanovic, N., Cartsburg, M., Kutsche, R., Widiker, J., Kschonsak, F.: Model-based Interoperability of Heterogeneous Information Systems: An Industrial Case Study. In: Proceedings of the ECMDA. (2009)
8. Reeve, L., Han, H.: Survey of semantic annotation platforms. In: Proceedings of the 2005 ACM symposium on Applied computing, NY, USA, ACM (2005)
9. Uschold, M., Grüninger, M.: Ontologies: Principles, methods and applications. In: Knowledge Engineering Review. (1996) 93–155
10. Patil, A., Oundhakar, S., Sheth, A., Verma, K.: Meteor-s web service annotation framework. In: Proceedings of WWW '04. (2004)
11. Naiman, C.F., Ouksel, A.M.: A classification of semantic conflicts in heterogeneous database systems. In: Selected papers of the workshop on Information technologies and systems. (1995)
12. Goh, C.H.: Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems. PhD thesis, Massachusetts Institute of Technology (1997)
13. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-based integration of information - a survey of existing approaches. (2001) 108–117
14. Boudjlida, N., Panetto, H.: Annotation of enterprise models for interoperability purposes. In: Proceedings of the IWAISE 2008. (2008)
15. Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., Schwinger, W., Wimmer, M.: Lifting metamodels to ontologies: A step to the semantic integration of modeling languages. In: MoDELS. (2006) 528–542
16. Reiter, T., Altmanninger, K., Kotsis, G., Schwinger, W., Bergmayr, A.: Models in conflict - detection of semantic conflicts in model-based development. In: Proceedings of the MDEIS 2007. (2007) 29–40
17. Li, C., Ling, T.W.: OWL-Based Semantic Conflicts Detection and Resolution for Data Interoperability. In: ER (Workshops). (2004)
18. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semant. **5**(2) (2007) 51–53
19. Haarslev, V., Mller, R.: Racer: A Core Inference Engine for the Semantic Web. In: In 2nd International Workshop on Evaluation of Ontology-based Tools. (2003)
20. SWI-Prolog (2008) `http://www.swi-prolog.org/`.
21. Zivkovic, S., Murzek, M., Kühn, H.: Bringing ontology awareness into model driven engineering platforms. In: TWOMD. (2008) 47–54
22. Roser, S., Bauer, B.: Automatic generation and evolution of model transformations using ontology engineering space. (2008)
23. Fabro, M.D.D., Bézivin, J., Jouault, F., E.Breton, Gueltas, G.: AMW: a generic model weaver. In: Proceedings of IDM05. (2005)
24. Agt, H., Bauhoff, G., Kumpe, D., Kutsche, R., Milanovic, N., Shtelma, M., Widiker, J.: Metamodels and Transformations for Software- and Data-Integration. Technical Report 2010/2, Technische Universität Berlin (2010)
25. Brömme, T.: Analyse und Verarbeitung von semantischen Annotationen für Web Services. Diploma Thesis, Technische Universität Berlin (2010) (in German).
26. Shtelma, M., Cartsburg, M., Milanovic, N.: Executable domain specific language for message-based system integration. In: Proc. MoDELS. (2009)
27. Agt, H., Widiker, J., Bauhoff, G., Milanovic, N., Kutsche, R.: Model-based Semantic Conflict Analysis for Software- and Data-integration Scenarios. Technical Report 2009/7, Berlin University of Technology (2009)
28. Blomqvist, E.: Pattern Ranking for Semi-automatic Ontology Construction. In: Proc. of the 23rd Annual ACM Symposium on Applied Computing. (2008)