Towards an Ontology-based Approach for Heterogeneous Model Matching

Mahmoud EL HAMLAOUI[1, 2], Cassia TROJAHN[1], Sophie EBERSOLD[1] and Bernard COULETTE[1]

[1] IRIT Laboratory, MACAO Team & MELODI Team
{firtname.lastname}@irit.fr
[2] ENSIAS, SIME Laboratory, IMS Team, University of Mohamed V Souissi

Abstract. The overall goal of our approach is to relate models - of a given domain - that are designed by different actors in different Domain Specific Languages, and thus are heterogeneous. Instead of building a single global model, we propose to organize the different source models as a network of models, which provides a global view of the system through a correspondence model. This latter, conform to a correspondence meta-model is built via a manual matching process. In this paper we explore the possibility of representing models as ontologies and take advantage of an automated process to match them.in order to enhance the automation of the matching process.

Keywords: DSL, heterogeneity, correspondence, matching, refine, ontology

1 Introduction

Models manipulated by different actors within the design process of complex systems are by nature heterogeneous. One common way to describe such models is to use Domain Specific Languages (DSLs) [21]. Matching these models is one of the key tasks for ensuring the consistency of the overall system. However, due to the specific way these models are represented, correspondences between heterogeneous models are generally established manually by the domain expert [3]. Approaches for automating the matching process usually consist in transforming the input models to other representations, such as tree models [7] or directed labeled graphs [1], before the matching step itself. In this paper, we propose to take advantage of ontologies, which provide a formal representation of the knowledge about a given domain [19], and automated ontology matching approaches [12], in order to automate and optimize the model matching process. Our proposal extends a previous work on matching models through a correspondence model [13]. Correspondences link two or more elements belonging to different models. They are first established at meta-model level, and are then instantiated (refined) at model-level. We distinguish domain independent relationships (e.g., “dependency”, “generalization”, “aggregation”, and “similarity”) from domain specific ones. In our previous proposal, the first step of the matching process consists in automatically refining a set of correspondences at model-level, from corre-

spondences established by the domain expert at meta-model level. The second step consists in filtering out potential incorrect correspondences from the set of instantiated ones. In [3], we have proposed to refine the correspondences from meta-model level to model level in a semi-automatic way, the expert being in charge of filtering (with the help of a tool) the correspondences that have been generated at model level. In this paper, we propose to exploit ontology-based matching approaches for enhancing the automation of the matching process at model level. We illustrate our approach with a case study extracted from a Conference Management System (CMS). We focus in this paper on the similarity relationship between two model elements.

The rest of this paper is organized as follows. Section 2 introduces the case study that was chosen to illustrate our approach. Section 3 presents our correspondence meta-model and the matching process between meta-models and models. Section 4 discusses how correspondences at model level can be established through refinement of correspondences at meta-model level. Section 5 presents how we use ontologies in our approach. Section 6 evaluates the performance of our proposal. Section 7 compares our proposal to related work, and finally Section 8 concludes and draws perspectives.

2 Running example: Conference Management System (CMS)

CMS is a software system aiming to automate functions needed in the management of a scientific conference: call for papers, paper submission, paper assignment for evaluation, notification of the final decision, registration, etc. Even if it is not really a complex system, the CMS has been chosen because it is firstly well known to most of researchers; secondly, it is relevant because it involves different actors, working with different points of view. We consider that there are three business domains in the CMS, covering various aspects of modelling. Each business domain is described by a dedicated model, conform to a dedicated meta-model, and is manipulated by actors with specific roles: (a) *Software Architect*: responsible for the conference management design; he creates a model – called *softwareDesign* – expressed through a specific software design meta-model; (b) *Database Administrator*: responsible for storing data; he creates a model – called *persistence* – expressed through a persistence meta-model; (c) *Process Engineer* : responsible for the way to conduct a CMS; he creates a model – called *businessProcess* – expressed through a business process meta-model. Due to lack of space the CMS meta-models are not described in this paper. Fig.1, Fig.2 and Fig.3 show fragments of the partial models of CMS.

3 Meta-model and Model Matching Process

Meta-model for representing correspondences (MMC). Establishing correspondences between models is a way to ensure and to maintain consistency of the whole system. Our approach consists in analyzing input (meta)-models in order to identify correspondences that exist among them. These correspondences are then stored into a model of correspondences (M1C) which is conform to a meta-model of correspond-

ences (MMC). It expresses links among meta-elements of distinct meta-models as well as links among elements of distinct models. MMC is composed of a generic part common to all the domains, and of a specific part which depends on the specific domain modeled (Fig.4). The main meta-classes of this correspondence model are : (a) *CorrespondenceModel:* represents all the correspondences established among at least two (meta)-elements belonging to different (meta)-models; (b) *Correspondence*: composed of a relationship and its extremities, which represent elements from input models; (c) *Relationship*: abstract meta-class that defines relationships among (meta-) elements of different (meta-)models and allows for defining n-ary correspondences connecting more than two elements at once. Its definition is done through specialization of ”Relationship”, by introducing two sub-meta-classes: “DomainIndependentRelationship” and “DomainSpecificRelationship”; (d) *DomainIndependentRelationship*: abstract meta-class that represents the generic relationships that may exist in different domains; and (e) *DomainSpecificRelationship*: abstract meta-class representing relationships among models in a specific domain. New relationships are specified by specialization of this meta-class according to the studied domain. The other meta-classes will be described progressively in subsequent sections.

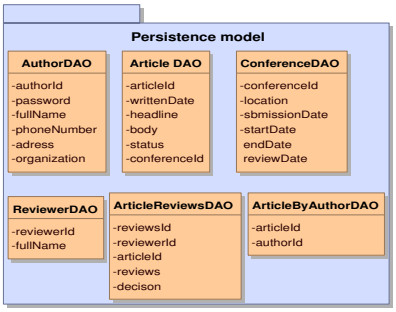

Fig. 1. Extract of the persistence model

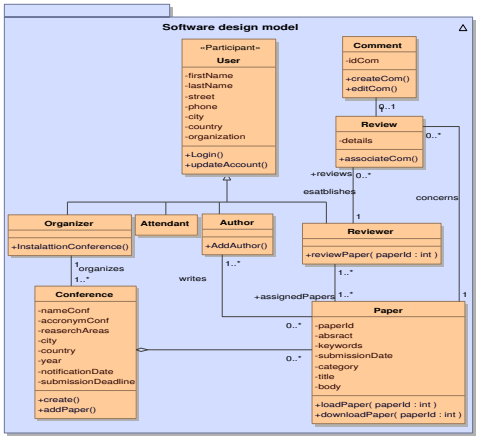

Fig. 2. Extract of the software design model

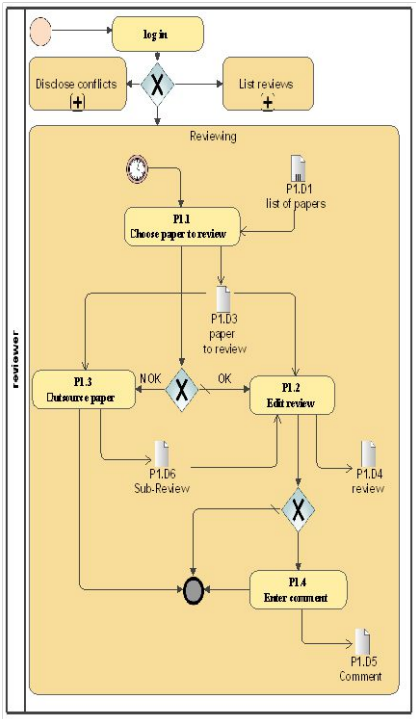

Fig. 3. Extract of the business process model

Global matching process. The correspondence model (M1C) is created via a “matching process”. This process, described in [13], involves two actors, namely, a domain

(design) expert who can be seen as an orchestrator of the system, and a tool to support the automated phases. The process takes as input the various models, their respective meta-models and the kernel of the meta-model of correspondences (the generic part). It begins by identifying correspondences between meta-elements so as to produce an intermediate correspondences model called M2C. Correspondences stored in M2C are called High Level Correspondences (HLCs). HLCs are then refined in order to produce Low Level Correspondences (LLCs) at model level.

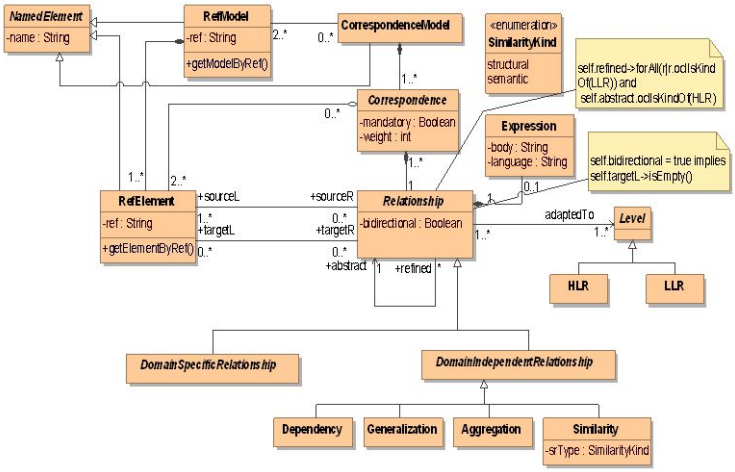

Fig. 4. Overview of the generic part of the meta-model of correspondences (MMC)

4 Defining correspondences on the CMS case study

HLCs stored in the M2C model are defined only once during the modelling cycle. Fig.5 shows examples of HLC correspondences using 5 types of relationships. For example, a correspondence encompasses a link between the meta-element “Task” on one side and the meta-element “Operation” on the other side through a “Contribution” relationship. A “Similarity” is defined between the meta-elements “Property” and “Column”, but also between the meta-elements “Table”, “Entity” on one side and “Table” and “DataObject” on the other side. The last type of relationship, called “Aggregation”, is used to relate the meta-elements “Property” and “Column”.

HLCs are exploited to generate LLCs stored in the M1C. Actually, LLCs are obtained by refinement of HLCs. In software and system engineering, refinement is a classical way to reuse [9] [10] [4]. It can be seen as a way of crossing different levels of abstraction with the purpose of adding details when passing from a given level to a more concrete one. According to [8], even though refinement is a key concept in MDA (Model Driven Architecture), it is loosely defined, and open to misinterpretation. The refine notion has also been defined in UML [6] as a stereotype for “Abstraction” - a directed relation from an element to another one stating that the dependent element (concrete) depends on the other one (abstract). A transition from HLC to

LLC is similar to a transformation of a PIM (platform independent model) into a PSM (platform specific model) in the context of MDA.
In this work we consider *refinement* as a transformation between HLCs and LLCs. This is done by projecting abstract correspondences on the concrete level. The principle consists in defining a HLC and then to reuse it each time it is needed at the model level. Cx=[Similarity(Entity, Table)], Cy=[Similarity(Paper, ArticleDAO)] are respectively examples of a HLC and the corresponding refined LLCs.
Thus, we define Refine (denoted by Rf) as follows: Ci Rf Cj, where Ci and Cj are two correspondences, iff Ci is defined from Cj which means that Ci is an upgrading of Cj by adding details in order to precise the correspondence. Fig.6 describes some LLCs created by refining HLCs presented in Fig.5.

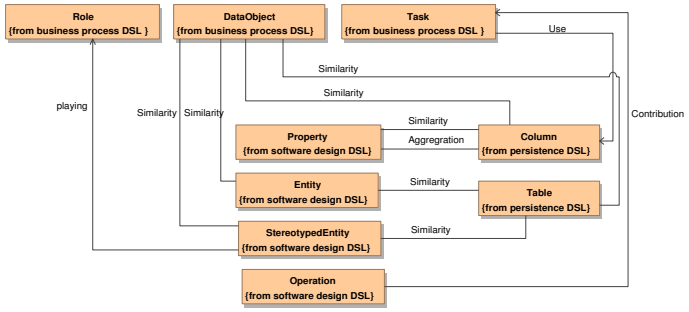

Fig. 5. Example of HLCs from CMS system

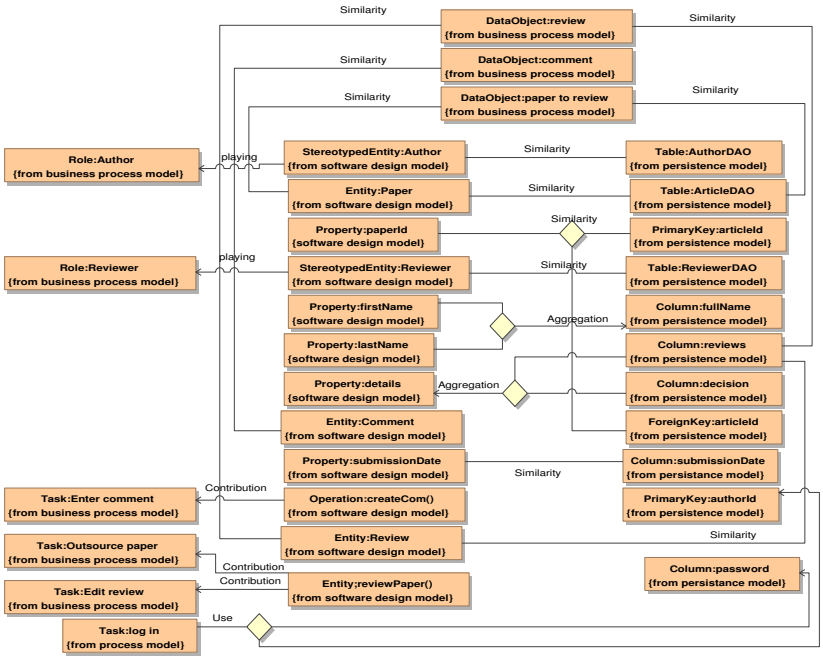

Fig. 6. Example of LLCs from CMS system

5 Ontology based approach

Ontologies are an explicit, formal specification of a shared conceptualization of a domain of interest [19]. They provide a model of the concepts of a domain and how these concepts are related to each other. In our proposal, we first automatically transform a set of Ecore (meta-)models into OWL ontologies and then we exploit ontology matching strategies in order to filter out potential incorrect LLCs.

The transformation process is carried out with the help of TwoUse (Transforming and Weaving Ontologies and UML in Software Engineering)[5]. In this process, each meta-class is represented as an OWL concept and the meta-properties are represented as OWL object properties (relating two objects) or OWL data properties (relating an object to a data value). We refer to these elements as terminological components (TBox) of ontologies. For the models, the instances of meta-models are represented as OWL individuals. Each individual is related to a type (meta-class) and is described by a set of OWL object and data properties, which are defined at meta-model level. We refer to the instances as assertional components (ABox) of ontologies.

The correspondence model (Fig.5) is also represented as an OWL ontology (TBox), where HLCs and LLCs are represented as OWL individuals (ABox). While HLCs are manually established by the domain expert before the transformation step, LLCs will be automatically generated within our matching process.

The first step of the matching process consists in automatically generating a set of OWL LLCs from the set of OWL HLCs (refining process). From each OWL HLC relating a source and a target elements (OWL individuals), one OWL LLC is automatically created. The second step consists in automatically filtering out potential incorrect correspondences from the initial set of OWL LLCs, with the help of ontology matching strategies. For each source and target elements ("RefElement" in Fig.6) of a LLC, we retrieve the value of the data property describing them (i.e., following the correspondence model, the "name" data property of a "RefElement" describes this element at model level, e.g., "conferenceId"). These values are then compared using different syntactic metrics. Correspondences that do not match using these metrics are removed from the initial set of LLCs.

In fact, MMC serves as a contextual basis for the matching process at model level, limiting the generation of a correspondence to elements whose type participates in a HLC. Even if the contextual information helps avoiding the creation of correspondences between elements of types that do not match (e.g., an "Operation" and a "Property"), it does not guarantee the generation of correct correspondences. Although many different ontology matching approaches have been proposed in the literature [11][12], and most of them exploit the Tbox level of ontologies. Here, we work essentially at the ABox level, adopting syntactic strategies for comparing data properties of OWL individuals. This first choice is due to the fact that we are working in a domain-specific set of ontologies where the terms used to describe the elements at model level through different models is supposed to be similar. Finally, external resources as WordNet do not cover the vocabulary of a particular domain, as the one we consider here.

6 Preliminary evaluation

We have conducted our experiments using the CMS case study. In this first series of experiments, we focused on the "similarity" relationship. The correspondences at meta-model level (HLC) were manually established by a domain expert. Manually created correspondences at model level (LLC) were used as reference alignments to evaluate our approach. As evaluation metrics, we applied well known metrics such as precision (the ratio of correctly found correspondences over the total number of returned correspondences), recall (the ratio of correctly found correspondences over the total number of expected correspondences, i.e., number of correspondences in the reference alignment) and f-measure (the harmonic mean of precision and recall). As depicted in Fig. 5, the alignment at meta-model level is composed of 7 HLCs involving the similarity relation. From these HLCs, through a projection process, for each pair of models (sofwareDesign-persistence, BusinessProcess-SoftwareDesign, and BusinessProcess-Persistence), we automatically generated the set of all possible LLCs (i.e., a LLC is generated between elements whose types are involved in a HLC). We refer to this set as "projection" in the following. In order to filter out incorrect correspondences from this projection set, we syntactically compare the values of the data properties of each ontology individual using equality of strings and Levenshtein edit distance [17]. This metric measures the effort to transform a string into another. We have also run LogMap [18], a publicly available ontology matching system that is able to deal with the ABox level of ontologies. LogMap is based on reasoning and inconsistency repair techniques.

Table 1 presents the results for the 4 sets of correspondences ("projection", "equality", "edit distance", "LogMap"), which have been generated independently of each other. For SofwareDesign-Persistence (SD-PS), the projection set contains 462 correspondences. As expected, the precision for this set is very low (many incorrect correspondences have been generated), while recall is high (many correct correspondences were retrieved). An expected behavior has also been observed for the equality set, where precision is quite high and recall is low. Using the Levenshtein distance, we obtain intermediary values of precision and recall (and the best F-Measure). For LogMap, although having good values of precision, it was not possible to deal with the very specific way individuals are represented in our models. For BusinessProcess-SoftwareDesign (BP-SD), although projection still generates the best recall, equality performed better than edit distance, especially in terms of precision.

	Projection			Equality			Edit distance			LogMap		
	P	F	R	P	F	R	P	F	R	P	F	R
SD-PS	0.01	0.02	0.63	0.50	0.26	0.18	0.26	0.33	0.45	0.28	0.22	0.18
BP-SD	0.06	0.12	1.00	0.40	0.50	0.66	0.18	0.28	0.66	0.00	0.00	0.00
BP-PS	0.01	0.03	1.00	0.00	0.00	0.00	0.11	0.18	0.50	0.00	0.00	0.00

Table 1. Precision (P), Recall (R) and F-measure (F) for the sets of correspondences generated

It is due to the fact that most correspondences in the reference alignment are characterized by an equality between their data property values. Log-Map was not able to found any correspondence for this pair (we do not distinguish empty alignments from

wrong ones – these alignments are indicated with zeros in Tables 1 and 2). Finally, for BusinessProcess-Persistence (BP-PS), an opposite behavior was observed, where edit distance outperforms equality. This is due to the fact that correspondences in the reference alignment do not involve equality and very few ones that can be detected by the metrics we apply in this paper (e.g., "paper to review" and "ArticleDAO").
When considering the intersection between the sets (Table 2), i.e., combining contextual information from projection and a syntactic metric on data property values, for SofwareDesign-Persistence (SD-PS), the best values where obtained when combining projection and edit distance. For BusinessProcess-SoftwareDesign (BP-SD) and BusinessProcess-Persistence (BP-PS), a similar behavior was observed. Although our initial approach is naive, we obtain better results when combining projection and syntactic metrics. Our process could also include a pre-processing step in order to treat compound terms (e.g., "Article DAO" instead of "ArticleDAO"). Furthermore, the use of other syntactic metrics [20] is envisaged.

	Projection+equality			Projection+edit			Projection+LogMap		
	P	F	R	P	F	R	P	F	R
SD-PS	0.50	0.26	0.18	0.65	0.52	0.45	0.50	0.26	0.18
BP-SD	1.00	0.80	0.66	0.66	0.66	0.66	0.00	0.00	0.00
BP-PS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2. Precision, Recall and F-measure when combining projection and syntactic measures

7 Related Works

Model matching. Several research works related to models matching have been discussed in a previous work [13]. Among identified shortcomings of existing approaches, one can note that they manage only binary correspondences and therefore cannot establish n-ary ones that relate a model element to a set of elements belonging to other models. In addition, studied approaches lead to produce a correspondence model between each pair of input models whereas we build a unique model of correspondences among input models. Several approaches consist in transforming the input (meta-)models to other formats in order to apply their matching techniques. For example, MatchBox [7] transforms input meta-models into a tree model called AMC (Auto Mapping Core). The process continues by applying a set of matching strategies to produce the model of correspondences. Another approach, described in [10], consists in transforming the meta-models to directed labeled graphs. The correspondences are then obtained by applying the Similarity Flooding algorithm. We could not use those approaches in our work because they propose to apply matching techniques only at meta-model level and they do not consider the model level. Furthermore they are not appropriate with meta-models which have a big semantic gap between them because in this case, according to [2], this results in a very low precision and in a very poor recall (mostly below 0.10). The result is satisfactory only when it comes to matching techniques between a former meta-model and a new version such as UML 1.4 and UML 2.4 as they share the same properties.

Ontology matching. Many approaches to the matching problem have been proposed in literature since the last decade [14][11][12]. These approaches can be classified along the many features that can be found in ontologies (labels, structures, instances, semantics), or with regard to the kind of disciplines they belong to (e.g., statistics, combinatorial, semantics, linguistics, machine learning, or data analysis). While terminological and syntactic methods lexically compare strings (tokens or n-grams) used in naming entities (or in the labels and comments concerning entities), semantic methods utilize model-theoretic semantics to determine whether or not a correspondence exists between two entities. Approaches may consider the internal ontological structure, such as the range of their properties (attributes and relations), their cardinality, and the transitivity and/or symmetry of their properties, or alternatively the external ontological structure, such as the position of the two entities within the ontological hierarchy. The instances (or extensions) of classes could also be compared using extension-based approaches. In addition, ontology matching systems rely not on a single approach. Finally, using domain ontologies [15] and different knowledge sources [16] as background knowledge within the matching process has been exploited in the literature. One initiative for evaluating ontology matching approaches for model matching was proposed in the context of the Ontology Alignment Evaluation Initiative (OAEI) campaigns. The aim was to compare classical model matching approaches and ontology-based approaches. Each test case consisted of a pair of Ecore meta-models, a pair of OWL ontologies, and a reference alignment. However, no system participating in that campaign was able to deal with this data set, showing that there is room left for exploiting this kind of approach.

8 Final remarks and Future Work

Our general goal is to related heterogeneous models corresponding to different points of view on a complex system. In this paper, we have put the emphasis on the use of ontology matching strategies to automate the production of correspondences at model level. Based on a common set of correspondences at meta-model level, a set of correspondences at model level is generated, and then refined with the help of syntactic matching approaches. Although we focus on the similarity relationship, as most of existing approaches in model matching, our work is a gateway that will enable us to exploit other kinds of relationships. The novelty with respect to other approaches is the use of correspondences at meta-model level, which serve as a context to produce correspondences at model level.

As future work, we have identified several directions to improve the approach proposed in this paper. First, we intend to exploit other relationships than similarity (i.e., dependency, aggregation), what is marginally treated in both models and ontology matching areas. Secondly, we plan to use matching approaches, in particular those based on background knowledge (e.g., using domain ontologies). This approach can be interesting in order to find the non-trivial relations between models. Third, we plan to apply other syntactic metrics and combine them with semantic ones.

References

1. Jean-Rémy Falleri, Marianne Huchard, Mathieu Lafourcade, and Clémentine Nebut. Metamodel matching for automatic model transformation generation. In *Model Driven Engineering Languages and Systems*. Springer, pages 326–340, 2008.
2. Gerti Kappel, Horst Kargl, Gerhard Kramler, Andrea Schauerhuber, Martina Seidl, Michael Strommer, and Manuel Wimmer. Matching meta-models with semantic systems-an experience report. In *BTW Workshops*, 38–52, 2007.
3. Mahmoud El Hamlaoui, Sophie Ebersold, Adil Anwar, Bernard Coulette, Mahmoud Nassar. Heterogeneous models matching for consistency management. Proc. IEEE RCIS, 1-12, Marrakech, Morocco, 2014.
4. Nenad Medvidovic and Richard Taylor. A classification and comparison framework for software architecture description languages. Software Engineering, IEEE Transactions on, 26(1):70–93, 2000.
5. Fernando Silva Parreiras, Steffen Staab, and Andreas Winter. TwoUse: Integrating UML models and OWL ontologies. Technical Report 16/2007. Tech. Inst. für Informatik, 2007.
6. OMG UML. Uml 2.0: Superstructure specification. http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF/, November 2007.
7. Konrad Voigt, Petko Ivanov, and Andreas Rummler. Matchbox: combined meta-model matching for semi-automatic mapping generation. SAC '10, ACM, pages 2281–2288, New York, NY, USA, 2010.
8. Dennis Wagelaar. Context-driven model refinement. In Proceedings of the 2003 European Conference MDAFA'03, pages 189–203, Berlin, Heidelberg, 2005. Springer-Verlag.
9. Wayne Wolf. Hardware-software co-design of embedded systems [and prolog]. Proceedings of the IEEE, 82(7):967–989, 1994.
10. Michael Wooldridge. Agent-based software engineering. Software Engineering. IEEE Proceedings- [see also Software, IEEE Proceedings], 144(1):26–37, 1997.
11. Rahm Erhard, Bernstein Philip. A survey of approaches to automatic schema matching. The VLDB Journal. Edited by P. Scheuermann. Springer-Verlag, 334-350, 2001.
12. Euzenat Jerôme, Shvaiko Pavel. Ontology matching. Springer, 2nd edition, 2013.
13. Mahmoud El Hamlaoui, Sophie Ebersold, Adil Anwar, Bernard Coulette, Mahmoud Nassar. Towards a framework for heterogeneous models matching. Journal of Software Engineering, Science Alert, 132-151, New York, USA, 2014.
14. Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. The Knowledge Engineering Review Journal (KER). Volume 18, 1-31, 2003.
15. Zharko Aleksovski, Warner ten Kate and Frank van Harmelen. Using multiple ontologies as background knowledge in ontology matching. CISWeb Workshop, 35-49. 2008.
16. Marta Sabou, Mathieu d'Aquin, Enrico Motta. Exploring the Semantic Web as Background Knowledge for Ontology Matching. Journal on Data Semantics XI. Volume 5383, 156-190, 2008.
17. Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. English Translation in Soviet Physics Doklady, 10(8) p. 707-710, 1966.
18. Ernesto Jimenez Ruiz, Bernardo Grau, Yujiao Zhou and Ian Horrocks. Large-scale Interactive Ontology Matching: Algorithms and Implementation. ECAI, 444-449, 2012.
19. Thomas Gruber. Toward principles for the design of ontologies used for knowledge sharing, International Journal of Human-Computer Studies, 43(5-6), 907–928, 1995.
20. Michelle Cheatham and Pascal Hitzler. String Similarity Metrics for Ontology Alignment. In Proceedings of the International Semantic Web Conference. Springer, 294-309, 2013.
21. Martin Fowler. Domain-Specific Languages. Addison-Wesley, 2010.