A framework for comparing conceptual models

Daniel Pfeiffer, Andreas Gehlert

European Research Center for Information Systems
Westfälische Wilhelms-Universität Münster
Leonardo-Campus 3
48149 Münster
daniel.pfeiffer@ercis.de

Lehrstuhl für Systementwicklung
Technische Universität Dresden
Münchner Platz
01062 Dresden
andreas.gehlert@wise.wiwi.tu-dresden.de

Abstract: Conceptual models are a widely used mean for documenting software systems as well as describing organisational structures. The trend towards integrated and flexible information systems has encouraged research about the comparison of conceptual models. Current approaches on the identification of similarities between conceptual models often adopt an automation perspective only. In this paper we will unfold severe arguments that a fully automatic model comparison process is not feasible. Furthermore, we will show that only a semi-automatic process can perform the comparison of conceptual models at the semantic level. On this theoretical basis, we will develop a framework which identifies all necessary and sufficient components for comparing conceptual models. We will show that this framework includes all the requirements that a semi-automatic model comparison process must meet.

Keywords: conceptual model, model comparison, automatic model comparison

1 Introduction

Since their first appearance in the mid of the 1970th conceptual models have gained a high economic importance. Together with modelling languages and methods, conceptual models have significantly influenced the way how software systems are developed today. By stressing the customer's perspective on a software system they promoted the transformation of software development from an art to a profession. In the beginning of the 1990s, accompanied by new findings in management science, the generally positive experiences with conceptual models were transferred from software systems engineering to organizational design. This established conceptual models also as a broadly applied mechanism for describing the business processes and corporate structures in an organization. The significance of conceptual models is embodied by the proposal to define them as the core of the Information Systems (IS) discipline [STW03].

In the last 10 years the trend towards integrated and flexible information systems also inspired research about comparison and integration of conceptual models [SHN01; BHP00; CU04; MP01]. The relevance of reference models for the implementation and customization of standard software motivated research about the identification of structural similarities between company specific conceptual models [FL05]. The introduction of data warehouses as well as mergers and acquisitions of companies made the consolidation of databases necessary. These effects also inspired work on the integration of conceptual data models [MZ98; HG01; Jo94; MWJ99]. The scientific inquiries revealed that the ability to compare conceptual models is an essential prerequisite for their integration [RB01].

The first attempts on model comparison were manual approaches. The elements of two or more models were mapped by hand only guided by the experience of the specialists involved. The inherent complexity of model comparison provoked a change towards the search for automatic similarity mechanisms [MZ98; HG01; Jo94]. The objective of the automatic approaches is to construct an algorithm which is able to compare conceptual models with only minimal or no human interaction. However, despite of this intensive scientific discussion, the techniques of model comparison have achieved only little practical impact within the domain of conceptual modelling so far.

This raises the question whether a fully automatic comparison of conceptual models is theoretically feasible. This issue motivates a detailed analysis of the theoretical requirements of model comparison. We will show that this examination leads to the conclusion that conceptual models cannot be compared completely automatically. Rather, this process must include human decisions to address semantic issues within these models. Therefore, we will develop a framework which combines the automatic with the manual perspective on model comparison. We will define the components of this framework and describe their interrelations.

The paper proceeds as follows: In the next section we will explain the problems of model comparison and will give reasons why a limitation on automatable aspects of model comparison is not sufficient. In the subsequent section we will present a framework for a semi-automatic model comparison which separates the manual activities from those which are capable of being automated. The paper concludes with a short summary of our results and an outlook to future research activities.

2 Automatic and semi-automatic model comparison

In this section, firstly, we analyse the requirements of a fully automatic model comparison process. Furthermore, we show that these requirements are theoretically not feasible (subsection 2.1). Therefore, secondly, we conclude, that a semi automatic model comparison is necessary and sufficient for this purpose. To prove our claim we need to separate manual and automable activities and need to show that these activities are in turn feasible. This leads to new and reformulated requirements of a semi automatic model comparison process (subsection 2.2).

2.1 Limits of automatic model comparison

Model comparison can be divided into two categories: the syntactical and the semantic model comparison[1]. *Syntactical model comparison* deals with the question, whether certain strings or graphical objects have the same or a similar form. *Semantic model comparison* addresses the issue, whether these strings or graphical objects have the same or a similar meaning. A practically relevant model comparison must operate at least at a semantic level in order to be interpretable from a real world perspective [GPS02].

According to Pfeiffer and Niehaves, a conceptual model consists of a set of modelling language statements on the one hand and multiple domain language expressions on the other hand [PN05]. Esswein et al. argued that the semantics of each conceptual model can be divided into the semantics of the modelling language constructs (abstract semantic) and the semantics of each individual model statement (concrete semantics) [EGS04]. Consequently, the combination of these two approaches leads to three distinct theoretical levels: the level of the conceptual model, the level of the modelling language and the level of the application domain language. From these three levels three prerequisites can be derived which must be fulfilled in order to facilitate a fully automatable model comparison:

[1] In this paper we disregard pragmatic questions.

(R1) *The conceptual models must be represented in a computer readable form.* Model comparison presupposes full access to all elements of the conceptual model. All graphical objects, textual statements and their interconnections must be available. In the case of a computer conducting the model comparison process the conceptual model must comprise an electronic representation of all those formal properties.

(R2) *The conceptual modelling language must be represented in a computer understandable way.* The conceptual modelling language defines the criteria how modelling language constructs can be identified in a real world situation as well as the rules how the constructs can be combined and depicted in the resulting model. A conceptual model instantiates the concepts of a modelling language. Therefore, the meaning of the modelling language constructs significantly influences the proposition of a conceptual model. Thus, the meaning of each modelling language construct must be present during the comparison process. If a computer performs this procedure the meaning of the modelling language constructs has to be explicitly stated in an electronic form.

(R3) *An application domain language must be represented in a computer understandable way.* An application domain language contains all terms that are relevant in a certain scope of reality. Conceptual models include a subset of the terms of an application domain language. In order to be able to compare two conceptual models the meaning of these terms must be known. In the case of a computer conducting the model comparison the machine needs an explicit representation of the meaning of these terms.

Requirement (R1) is straightforward and can easily be fulfilled. Conceptual modelling is a mostly tool based process today. Programs like ARIS Toolset, Rational Rose or Cubetto Toolset dispose of a repository in which all formal properties of a conceptual model can be stored. The repository can accommodate all model elements and their graphical and logical interconnections. Thus, the repository provides the technological basis for a syntactical model comparison. This facilitates an analysis of the structural and formal similarities of the conceptual models. However, it does not permit statements on the semantic relations between the models. Semantic issues are address by the requirements (R2) and (R3).

Requirement (R2) demands for an explicit, computer understandable representation of the semantics of the modelling language. Conceptual modelling languages are semi-formal languages which dispose of the following two kinds of semantics:

1. The *formal semantics*, i. e. a description of how to directly or indirectly map a modelling language construct to a set of computer instructions, is described as a set of matching rules. A conceptual modelling language may but not always does comprise a specification of formal semantics.

2. The *real world semantics*, i. e. how to identify modelling language constructs in their application domain and what meaning they represent in this environment, is described in terms of a natural language statement.

In order to meet requirement (R2) it is crucial that both kinds of semantics can be represented by a computer. Formal semantics is per definition interpretable by a machine. However, real world semantics is distinct from formal semantics, since the world is not a formal system. The representation of real world semantic requires that the meaning of a natural language statement can be grasped by a machine. Thus, whether real world semantics can be processed by a computer is closely connected with requirement (R3). We claim that requirement (R2) holds, iff requirement (R3) can be fulfilled.

Requirement (R3) calls for an explicit, computer understandable representation of the semantics of the application domain language. Thus, for a fully automatic model comparison process it is necessary to represent real world semantics of the application domain language in a computer interpretable form. In other words, real world semantics must be reducible to a finite set of formal rules and therefore, expressible by means of a syntactical structure. However, none of the attempts to describe real world semantics in terms of a finite set of propositions has succeeded so far [Se69]. The following arguments can be brought forward in order to assert that requirements (R3) cannot be fulfilled:

(A1) *A computer cannot grasp a substantial part of real world semantics.* This argument is closely related with the reasoning in John Searle's famous Chinese room thought experiment [Se84]. Every computational operation of a computer is based on a predefined set of rules. These rules permit a transformation of syntactical structures into other syntactical structures. Hence, what in Computer Science is called "formal semantics" is nothing else than a description of what actions a computer has to execute in order to perform a syntactical transformation. Therefore, even if a computer would dispose of a sufficient amount of rules to answers all questions about a certain term, it would still not know what this term means in the real world. It still could only transform syntactical structures. Consequently, real world semantics cannot be represented by a computer system.

(A2) *The processing of real world semantics is not Turing computable.* If argument (A1) should be wrong and real world semantics could generally be gasped by a computer system, it would still require an adequate learning mechanism to acquire real world semantics. Kamlah and Lorenzen assume that the semantics of an application domain term can be restrained with the aid of propositions formed by other terms [KL84]. They call this process predication. However, with predication only an approximation to the proper meaning of a term can be reached. Only an infinite number of predications can assign a precise meaning to a term. Based on these assumptions it is possible to show that the process of predication is not Turing computable [Ho04]. Consequently, the acquisition and the processing of real world semantics in a computer system is also not Turing computable.

(A3) *No computer program can define what it means, that two terms of an application domain language are synonym.* If other mechanisms than predication should exist in order to acquire real world semantics or if a finite set of rules is sufficient to grasp it, one severe argument against (R3) would still remain. It is very problematic to state formally what it means that two terms a synonym [Qu63; Se69]. Gödel [Gö92] shows in his well know incompleteness theorem that a rigidly logical system can contain propositions that are undecidable or undemonstrable within the axioms of the system. Since, a computer program is a part of a formal system, one could conclude, that synonymity between terms belongs to those problematic propositions. If this argument is valid, there can be no computer program that is able to define what it means that two statements in a natural language are synonym [Pu88].

Since requirement (R1) holds, a purely syntactical model comparison process is accomplishable for a computer. However, the above arguments indicate that requirement (R2) as well as (R3) cannot be met by a fully automated model comparison process at the semantic level. Manual activities are required in order to address semantic issues within the model comparison process. Consequently, the comparison of conceptual models at the semantic level has to be performed semi-automatically.

2.2 Feasibility of semi-automatic model comparison

In the case of a semi-automatic model comparison process it is not required, that a computer can handle real world semantics, since this aspect can be performed manually. The result of this manual step is provided as input to the automated part of the comparison process. Thus, requirements R2 and R3 can be reformulated as follows:[2]

[2] The advantages of these reformulations will be described in detail in section 3.

(R2') *Equivalence and similarity relations between the conceptual modelling language constructs must be represented in a computer understandable way.* It is not necessary for a computer to understand the semantics of the modelling language in order to compare conceptual models. It is sufficient to dispose of a set of pre-defined rules, which formally describe the equivalence and similarity relations between the modelling language constructs. Based on these mapping rules the computer can compute the comparison result.

(R3') *The relationships between conceptual model elements and application domain terms must be represented in an explicit computer understandable way.* If a manual pre-processing step is assumed than it is no longer necessary that a computer grasps the real world semantics of the application domain terms. However, it is still required that a connection between the model and a specific application domain language exists, in order to dispose of a common reference point during the automated part of the model comparison process. This common reference point can be established by explicitly and formally described links between model elements and application domain terms.

Requirement (R2') holds for a semi-automatic model comparison process because the definition of (generic) mapping rules between the modelling language constructs can be performed manually previous to the comparison operation. The mapping rules again are specified formally and are therefore computer interpretable.

Requirement (R3') holds since ontologies, technical term models and reference models provide the vocabulary which can be linked in order to establish a common reference point. The reconstruction of the relationships between the conceptual model elements and application domain terms can be performed manually previous to the automated comparison operation and can be described by formal means. Thus, a semi-automatic model comparison process does fulfil the requirements (R1), (R2'), (R3') and is therefore feasible.

3 A Framework for Semi-Automatic Model Comparison

Every approach which is engaged in the comparison of conceptual models has to account for three sorts of conflicts: type conflicts, naming conflicts and structural conflicts [BLN86; LB01]. These three kinds of conflicts are described in table 1 and exemplarily demonstrated in figure 1 within the notation of the modelling language Entity Relationship Model (ERM). Naming conflicts and structural conflicts are caused by a deviant usage of the application domain language in conceptual models. Type conflicts, however, can be traced back to a divergent representation of reality on the level of the modelling language [BSH99].

We propose the framework depicted in figure 2 for a semi automatic model comparison process. It consists of two main components and two supporting ones. The two main components are used to resolve the conflicts described above. The type conflict resolver eliminates all type conflicts in the models. It belongs to the automatable part of the framework but uses manually prepared inputs for its operation (see subsection 3.3). The domain model builder represents the manual component and resolves naming and structural conflicts (see subsection 3.2).

Type conflicts	*Type conflicts* arise whenever the same fact of the application domain is represented by using different constructs of the modelling language. They result from differences in the utilization of modelling choices with regard to a suitable language concept.
Naming conflicts	*Naming conflicts* emerge due to use of synonym and homonym terms of a certain application domain in conceptual models.
Structural conflicts	*Structural conflicts* result from the description of reality at diverse levels of abstraction [KS96] (abstraction conflict) or whenever domain terms are modelled differently detailed (conflict of detail).

Table 1: Important model comparison conflicts

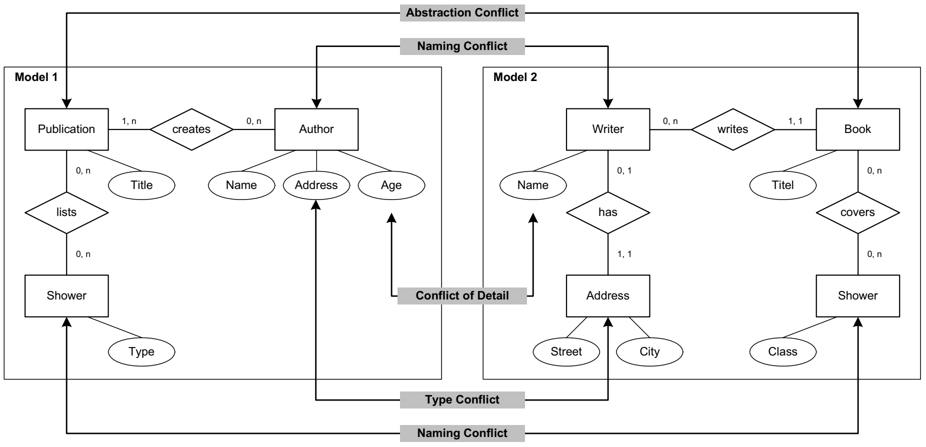

Figure 1: Examples of important conflicts

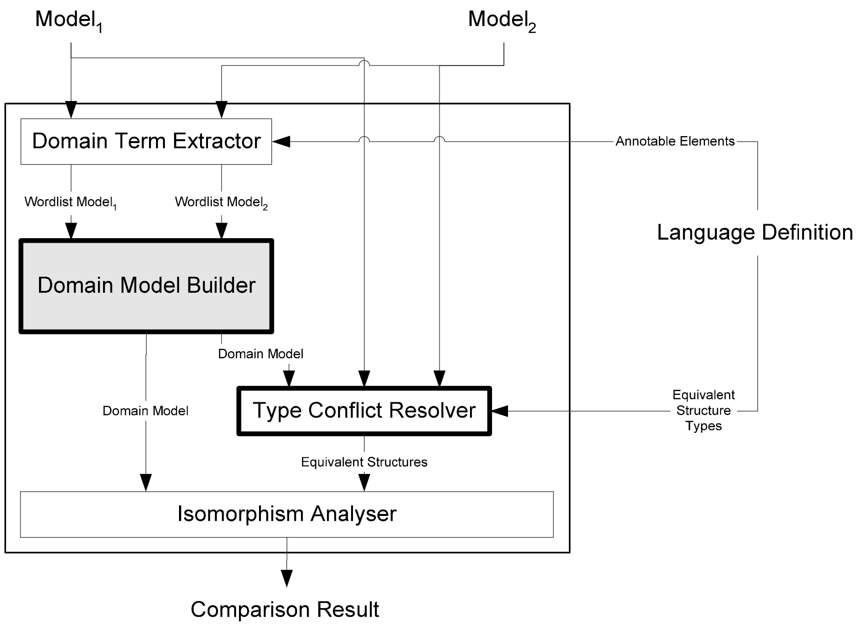

Figure 2: Model Comparison Framework

	Input	Output
Domain Term Extractor	Model_1, Model_2 as well as a list of modelling language constructs which can contain domain language expressions	domain language expressions of Model_1 and Model_2
Domain Model	lists of domain language expressions	relations between the domain language expressions in a computer readable format (Domain Model)
Type Conflict Resolver	Model_1, Model_2, the Domain Model and a generic description of equivalent and similar language expressions	concrete description of similar and equivalent model elements
Isomorphism analyser	the Domain Model and a concrete description of similar and equivalent model elements	Comparison Result

Table 2: Inputs and Outputs of the components of the framework

This framework uses the explicitly represented Model$_1$ and Model$_2$ as inputs and, therefore, fulfils requirement (R1). Similarities and equivalences relations between the modelling language constructs are taken as inputs to the framework according to requirement (R2'). Finally, the framework provides a component - the domain model builder - to account for the requirement (R3'). The domain model builder produces a net of domain language expression as well as a mapping to their corresponding model elements.

Comparing conceptual models using the proposed framework proceeds as follows (see table 2). All domain expressions relevant for the comparison are extracted from both models using the Domain Term Extractor component. On this basis the domain model builder identifies relations between these terms and stores them in a computer readable format. In parallel, a type conflict resolver examines the models and identifies similar or equivalent model structures based on equivalent and similar structure types described abstractly on the level of the modelling language. Since all information is now expressed formally, an isomorphism analyser uses the domain model and the equivalent structures to compose the comparison result.

3.1 Domain Term Extractor

A prerequisite for the domain model builder are the domain language expressions, which must be extracted from both conceptual models. The domain term extractor processes each model element with an associated domain language expression. In order to be able to identify these model elements there must be rules that define which corresponding modelling language constructs can actually contain a domain language expression. Consequently, beneath the two conceptual models the second input of the domain term extractor is the set of modelling language constructs which are capable of carrying a domain language expression. The output of this component is the list of all relevant domain language expressions.

3.2 Domain Model Builder

The application domain terms used in the model do not follow a formally pre-defined meta language but are rather referenced implicitly in the model. Hence, the decision which model element maps to what application domain term is specific to a concrete conceptual model only and cannot be described generically. As a consequence, it is a manual activity to find semantically equivalent or similar application domain terms.

The domain language builder addresses naming and structural conflicts. The handling of naming conflicts is straightforward as they can be simply resolved by a renaming operations leading to a controlled domain language [OS96]. Structural conflicts include conflicts of abstraction and conflicts of detail. Both conflict types can only be resolved by a systematic analysis and reconstruction of the domain language expressions of both models. In other words, the researcher who performs this activity needs to find similarities and differences between the diverse domain language expressions.

Within the domain model builder a researcher tries to find correspondences between the domain language expressions of both models. The resulting domain model specifies these relations including their mapping type (similarity or equivalence) in a computer readable format.

This domain model can be achieved either directly or indirectly (see figure 3). In the direct approach domain language expressions from both models are analysed and pair-wise similarity or equivalence relationships are made explicit (figure 3 left). The indirect approach uses a reference ontology (or reference model). Each domain language expression is compared with the constructs of that reference ontology (figure 3, right). The similarity or equivalence relationships can be derived transitively from these pairwise mappings by their composition [DL04]. Since this indirect approach uses the transitivity feature, it must be applied very carefully if similarity is involved, because the similarity relation is not generally transitive [DR02; GE05].

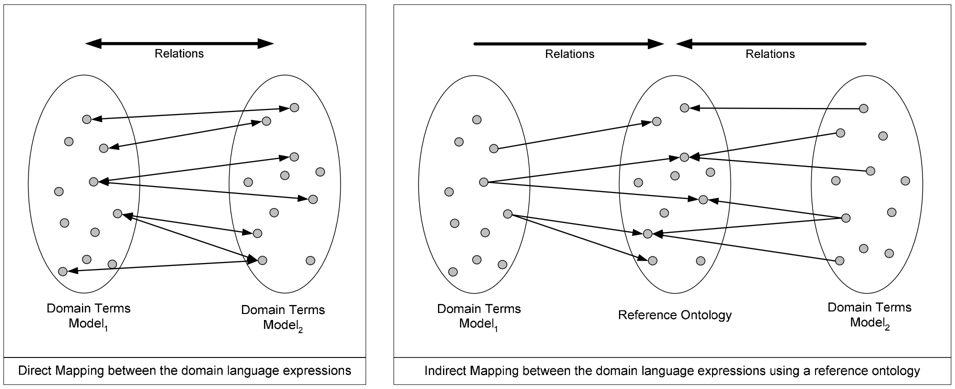
Relations
Domain Terms Model$_1$
Domain Terms Model$_2$
Direct Mapping between the domain language expressions
Relations
Relations
Domain Terms Model$_1$
Reference Ontology
Domain Terms Model$_2$
Indirect Mapping between the domain language expressions using a reference ontology

Figure 3: Direct and indirect relations between domain language expressions

3.3 Type Conflict Resolver

As described above, it is not feasible to establish equivalence and similarity mappings between modelling language constructs within a fully automated model comparison process. However, the clear cut connections between the model statements and the modelling languages can be exploited to separate this issue into an automatic and a manual part. Since, conceptual modelling languages are artificial languages which are explicitly defined it is generally feasible to specify mapping rules within these languages. Such mapping rules specify similar or equivalent modelling construct type structures.

This explicit set of mapping rules can be used as input to the model comparison process. Hence, irrespective of whether it is possible to represent the real world semantics of a modelling language in a computer understandable form, it is viable to define the connections within one language in terms of a set of manually defined rules. As the connection between a model element and its modelling language construct is not contextual, this set of rules can be created as generic mapping structure types for the involved languages and is therefore reusable in multiple comparison processes.

Each defined mapping rule consists of a pre-condition and a body. The pre-condition specifies the context in which the rule is applicable. In figure 1, for instance, the entity type address is equivalent with the attribute address (body) only because an identity between the entity types writer and author has been found during the semantic analysis (context). While the definition of the mapping rules is independent from the model and syntactic in nature only, resolving type conflicts requires semantic information encoded in the domain model.

Consequently, the type conflict resolver itself operates on the model level. This component instantiates the generic mapping rules, relates them to semantic information from the domain model and applies them to the models and, thereby, resolves their type conflicts.

As a consequence, (R2') holds for a semi-automatic model comparison process since the generic mapping rules can be defined prior to the actual model comparison and, hence, the automatic and manual activities can be clearly separated.

3.4 Isomorphism Analyser

At the end of the comparison process the isomorphism analyser summarises the information from the domain model as well as the equivalent and similar structures from both models and composes the comparison result. Since there might be more than one possible correspondence between the constructs of both models, the main task of the isomorphism analyser is to find the best possible matches based on all predefined information. Although, the isomorphism problem is algorithmically challenging it is feasible in terms of methods and techniques of Computer Science [FSV01].

The isomorphism analyser produces the result of the comparison. The result consists of relations between structures of the models 1 and 2, which are either similar or equivalent. In the case of similarity, the degree of similarity a degree of correspondence measure can be used [FL05].

4 Conclusions

In this paper we argued that an automatic model comparison is theoretically not feasible since it cannot handle the real world semantics of the conceptual models. We justified our reasoning based on three requirements each model comparison process must comply with. We could show that a fully automated model comparison algorithm is not able to cope with all of these three requirements. Consequently, we concluded that there must be a human involvement in the comparison process in order to account for the real world semantics. We modified two of the requirements based on the assumption, that the model comparison process is performed with human participation. We found evidence that a semi-automatic model comparison process can fulfil the modified requirements.

On this basis, we proposed a framework, which meets all the above mentioned criteria and provides the theoretical fundament in order to resolve type, structure and naming conflicts semi-automatically. The framework separates the automatable from the manual activities of the model comparison process. Therefore, the framework is well embedded into current research results of the Computer Science and the Information Systems discipline.

From the requirements and based on the framework the following practical implications can be drawn:

- Firstly, the constructors of conceptual modelling languages could make the model comparison process significantly less complex, if they would explicitly and formally state, what semantic relations between their modelling language constructs exist. By this means they would help to comply with requirement (R2').
- Secondly, the model comparison process could be simplified if the connection between the conceptual model and a technical term model, an ontology or a reference model were already established during the model construction. In this case, requirement (R3') would be fulfilled from the outset.
- Thirdly, a producer of a conceptual modelling tool can enhance the product by supporting technical term models, ontologies or reference models. With the aid of these artefacts a common semantic reference point for the model comparison can be established. Thus, with an additional type conflict resolver and a graph isomorphism analyser a conceptual modelling tool could be augmented by the functionality to compare conceptual models.

Since the purpose of the paper is to structure current and to guide future research (see [Pa03] for the guidance of frameworks), the proposed framework has neither been implemented nor empirically tested yet. Both aspects are subject to further research activities.

5 References

[BLN86] Batini, C.; Lenzerini, M.; Navathe, S. B.: A Comparative Analysis of Methodologies for Database Schema Integration. In: ACM Computing Surveys. Vol. 18 (1986), Issue 4; pp. 323-364.

[BHP00] Bernstein, P. A.; Halevy, A. Y.; Pottinger, R. A.: A vision for management of complex models. In: SIGMOD Record (ACM Special Interest Group on Management of Data). Vol. 29 (2000), Issue 4; pp. 55-63.

[BSH99] Brinkkemper, S.; Saeki, M.; Harmsen, F.: Meta-modelling based assembly techniques for Situational Method Engineering. In: Information Systems. Vol. 24 (1999), Issue 3; pp. p. 209-228.

[CU04] Caruso, F.; Umar, A.: Architectures to Survive Technological and Business Turbulences. In: Information Systems Frontiers. Vol. 6 (2004), Issue 1; pp. 9-21.

[DR02] Do, H. H.; Rahm, E.: COMA - A System for Flexible Combination of Schema Matching Approaches. In: Proceedings of the Proceedings of VLDB, 2002. pp. 610-621.

[DL04] Dragut, E.; Lawrence, R.: Composing Mappings Between Schemas Using a Reference Ontology. In (Tari, Z.; Ed.): Proceedings of the On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2004, Agia Napa, Cyprus, October 25-29, 2004. Proceedings, Part I, Berlin, Heidelberg, 2004. Springer Verlag; pp. 783-800.

[EGS04] Esswein, W.; Gehlert, A.; Seiffert, G.: Towards a Framework for Model Migration. In (Persson, A.; Stirna, J.; Eds.): Proceedings of the 16th International Conference on Advanced Information Systems Engineering, CAiSE 2004, Riga, Latvia, 2004. Springer.

[FL05] Fettke, P.; Loos, P.: Zur Identifikation von Strukturanalogien in Datenmodellen - Ein Verfahren und seine Anwendung am Beispiel des Y-CIM-Referenzmodells von Scheer. In: Wirtschaftinformatik. Vol. 47 (2005), Issue 2; pp. 89-100.

[FSV01] Foggia, P.; Sansone, C.; Vento, M.: A Performance Comparison of Five Algorithms for Graph Isomorphism. In: Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition. Ischia, 2001; pp. 188-199.

[GE05] Gehlert, A.; Esswein, W.: Ontological Analysis Reconsidered - A Formal Approach. In (Lytras, M.; Ed.): Proceedings of the First Workshop on Ontology, Conceptualizations and Epistemology for Software and Systems Engineering, 2005.

[Gö92] Gödel, K.: On Formally Undecidable Propositions of Principia Mathematica and Related Systems. Edition, Dover Publications, New York, 1992.

[GPS02] Guizzardi, G.; Pires, L. F.; Sinderen, M. J. v.: On the role of Domain Ontologies in the design of Domain-Specific Visual Modeling Languages. In: Proceedings of the 2nd Workshop on Domain-Specific Visual Languages, 17th ACM Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2002), Seattle, 2002.

[HG01] Hakimpour, F.; Geppert, A.: Resolving Semantic Heterogeneity in Schema Integration: an Ontology Based Approach. In (Welty, C.; Smith, B.; Eds.): Proceedings of the International conference on Formal Ontologies in Information Systems FOIS'01, Ogunquit, Maine, 2001. ACM Press; pp. 297-308.

[Ho04] Holten, R.: Integration von Informationssystemen - Theorie und Praxis im Supply Chain Management, in German. Habilitation, Habilitation, Westfälische-Wilhelms-Universität Münster, 2004.

[Jo94] Johannesson, P.: Linguistic instruments and qualitative reasoning for schema integration. In: Proceedings of the third international conference on information and knowledge management. ACM Press, Gaithersburg, Maryland, 1994; pp. 252-262.

[KL84] Kamlah, W.; Lorenzen, P.: Logical Propaedeutic. Pre-School of Reasonable Discourse. Edition, University Press of America, Lanham, MD, 1984.

[KS96] Kashyap, V.; Sheth, A.: Semantic and schematic similarities between database objects: a context-based approach. In: The International Journal on Very Large Data Bases (VLDB). Vol. 5 (1996), Issue 4; pp. 276-304.

[LB01] Lawrence, R.; Barker, K.: Integrating relational database schemas using a standardized dictionary. In: Proceedings of the 16th ACM Symposium on Applied Computing, Las Vegas, USA, 2001. ACM Press.

[MP01] Mikkonen, T.; Pruuden, P.: Flexibility as a design driver. In: IEEE Computer. Vol. 34 (2001), Issue 11; pp. 52-56.

[MZ98] Milo, T.; Zohar, S.: Using Schema Matching to Simplify Heterogeneous Data Translation. In (Gupta, A., et al.; Eds.): Proceedings of the 24rd International Conference on Very Large Data Bases, New York, 1998. Morgan Kaufmann Publishers Inc.; pp. 122 - 133.

[MWJ99] Mitra, P.; Wiederhold, G.; Jannink, J.: Semi-automatic Integration of Knowledge Sources. In (Blasch, E.; Ed.): Proceedings of the Second International Conference on Information Fusion (Fusion '99), Sunnyvale, California, 1999. International Society of Information Fusion.

[OS96] Ortner, E.; Schienmann, B.: Normative Language Approach - A Framework for Understanding. In (Thalheim, B.; Ed.): Proceedings of the Conceptual Modeling - ER '96, 15th International Conference on Conceptual Modeling, Cottbus, 1996. Springer; pp. 261-276.

[Pa03] Palvia, P., et al.: Mangagement Information Systems Research: Waht's There in the Methodology? In: Communications of the Association for Information Systems. Vol. 11 (2003); pp. 289-309.

[PN05] Pfeiffer, D.; Niehaves, B.: Evaluation of conceptual models - a structuralist approach. In: Proceedings of the European Conference on Information Systems (ECIS 2005), Regensburg, 2005.

[Pu88] Putnam, H.: Representation and reality. Edition, MIT Press, Cambridge, Massachusetts, 1988.

[Qu63] Quine, W. V. O.: Two Dogmas of Empiricism. In (Quine, W. V. O.; Ed.): From a Logical Point of View. Harper & Row, New York, 1963.

[RB01] Rahm, E.; Bernstein, P. A.: A survey of approaches to automatic schema matching. In: The VLDB Journal - The International Journal on Very Large Data Bases. Vol. 10 (2001), Issue 4; pp. 334-350.

[SHN01] Saleh, J. H.; Hastings, D. E.; Newman, D. J.: Extracting the essence of flexibility in system design. In: Proceedings of the The Third NASA/DoD Workshop on Evolvable Hardware, Long Beach, California, 2001. pp. 59-72.

[Se84] Searle, J. R.: Minds, brains and science. Edition, Harvard University Press, Cambridge, Massachusetts, 1984.

[Se69] Searle, J. R.: Speech acts : an essay in the philosophy of language. Edition, University Press, Cambridge, 1969.

[STW03] Shanks, G.; Tansley, E.; Weber, R.: Using ontology to validate conceptual models. In: Communications of the ACM. Vol. 46 (2003), Issue 10; pp. 85-89.