2007

Constructing Comparable Conceptual Models with Domain Specific Languages

Daniel Pfeiffer
European Research Center for Information Systems, pfeiffer@ercis.de

CONSTRUCTING COMPARABLE CONCEPTUAL MODELS WITH DOMAIN SPECIFIC LANGUAGES

Pfeiffer, Daniel, European Research Center for Information Systems, Leonardo-Campus 3, 48149 Münster, Germany, pfeiffer@ercis.de

Abstract

The scientific discussion on the comparison of conceptual models has mainly focused on existing modelling artefacts so far. Only minimal assumptions are made on the underlying modelling languages. However, we argue that if models are constructed with a particular type of language the model comparison process can be significantly simplified. For this purpose we introduce the class of operational domain specific languages. We formally show that with this language class: (1) type, synonym, homonym, and abstraction conflicts are eliminated as well as (2) the semantic model comparison can be traced back to the syntactic one. Based on the conceptual modelling language PICTURE we demonstrate that advanced semantical operations are facilitated by this language class.

Keywords: Model Comparison, Domain Specific Languages, Conceptual Modelling, PICTURE

1 INTRODUCTION

The broad use of conceptual modelling in software engineering and organisational design calls for models that are comparable. In large modelling projects the acquisition of relevant domain knowledge is performed in a distributed manner. Findings from empirical studies show that whenever various team members are involved the resulting models differ in terms of vocabulary, grade of abstraction and level of detail (Hadar & Soffer, 2006). Through these semantic conflicts automated analyses and transformations of the models are retarded. This hampers or even makes it impossible to efficiently consolidate different models into one integrated overall description. Consequently, a holistic view of the organisation is inhibited and the development of a company wide data model or the identification of process overarching reorganisation potential is constrained (Becker et al., 2006).

To address the problem of deviating descriptions two varying approaches have evolved. The first perspective on model comparison is to consider conflicting models as given and to apply syntactical transformations as well as semantical tests in order to identify similarities. This approach requires only minimal assumptions about the models at hand. Especially research in the area of conceptual data models as well as their integration adopts this position (e. g. Hakimpour & Geppert, 2001, Kashyap & Sheth, 1996, Mitra et al., 1999). Pfeiffer & Gehlert (2005) have presented a general framework for the comparison of conceptual models whose application, however, implies significant efforts. The second approach on the other hand focuses on the construction of conceptual models in order to facilitate comparatively easily matchable artefacts. It has emerged from the insight that comparing models with an arbitrary structure hampers the identification of semantically meaningful results. It is based on the assumption that establishing constraints while a model is constructed can distinctly simplify its later comparison. As a consequence, conventions have been used to restrict the freedom of the modellers to guarantee a certain level of compliance (Rosemann, 2003). In this paper we will take up this second perspective as it can help to reduce the comparison efforts in the first place.

Domain specific conceptual modelling languages are a promising way to simplify the comparison of conceptual models. Contrary to general-purpose languages like the Unified Modelling Language (UML) (Object Management Group, 2004) or the Entity Relationship Model (ERM) (Chen, 1976) domain specific languages are created to solve problems within a defined area of concern (Guizzardi et al., 2002, Rossi et al., 2004). In order to describe a particular domain they apply the specific vocabulary of this part of the world. Thus, not just the resulting models but already the modelling language has a semantic connection with the application domain. Hence, from the domain perspective semantically meaningful operations on the conceptual models, such as a model comparison, can already be defined at the language level. As the constructs of a domain specific language are derived from the domain vocabulary, domain experts acting as modellers are aware of their semantics. Thus, the selection of improper modelling constructs is reduced and the comparability of the models is increased.

The aim of this paper is to show which modelling language characteristics are required, in order to create conceptual models that can be compared in an automated manner. We will explain how domain specific languages can simplify the comparison process and will disclose the underlying causal relations. To reach that goal, we will formalize our arguments and prove the validity of the proposed language characteristics.

The remainder of this paper will proceed as follows: In the next section we will provide the basic vocabulary to discuss the comparison of modelling artefacts. Formal definitions of the terms conceptual model as well as conceptual modelling grammar are given and modelling rules are specified. Subsequently, we will explain the differences between syntactic and semantic model comparison. A formal definition of the different conflicts that can emerge during a model comparison is given. In the following section these conflicts are solved by specific language properties. For this purpose, the class of operational domain specific languages is introduced. The PICTURE-language is presented as an example. The paper closes with a summary of the main results and an outlook to further research.

In the following set-theoretic predicates are applied for formal definitions (Balzer et al., 1987). The notion of intentional and extensional semantics is based on the work of Patig (2004).

2 MODELS, LANGUAGES AND GRAMMARS

Informally, a conceptual model can be defined as a representation of an application domain expressed in a semi-formal, mostly visual language with the purpose of facilitating information systems development and organisational design (Evermann & Wand, 2001, Schütte & Rotthowe, 1998). A conceptual model is the result of an explication of an internal model. The internal model is a product of perception and cognition processes of a modeller who examines an application domain. The content of the internal model is influenced by the intentions of the modeller and the objectives of the modelling project. The internal model $IM = \langle IE, IR \rangle$ consists of a set of elements *IE* and relations $IR \subseteq IE \times IE$ between the elements. Therefore, *IM* can be considered as a system.

A description of the internal model *IM* is denoted as D_{IM}. D_{IM} is a linguistic artefact which provides the intentional semantics of *IM*. The intentional semantics of the English term "morning star" is for example a bright object in the night sky that can be seen only shortly before sunrise. The extensional semantics of *IM* is denoted by $M(D_{IM})$. $M(D_{IM})$ is the set of all interpretations of the description D_{IM}. In the example the extensional semantics consists of the planet Venus. In the case of an adequate and complete description of *IM* it follows that: $M(D_{IM}) = \{IM\}$, because *IM* is precisely characterised by D_{IM}. Each $\varepsilon \in IE$ and $\rho \in IR$ can be described in form of D_{ε} or D_{ρ} accordingly. In the case of an adequate and complete description of ε it holds true that: $M(D_{\varepsilon}) = \{\varepsilon\}$, analogical in the case of ρ: $M(D_{\rho}) = \{\rho\}$. Every description requires a language. A conceptual model *CM* complies with a description $D_{IM}^{CMG,DL}$ of the internal model D_{IM} with the modelling grammar *CMG* and the domain language *DL*. A domain language *DL* contains all meaningful statements which can be formed with the vocabulary of a certain application domain. LC_{DL} constitutes the language community to *DL*. LC_{DL} comprises all individuals who consider the language *DL* as their common property and follow its conventions. A conceptual modelling grammar *CMG* describes the rules governing the use of a conceptual modelling language.

Existing formalisms for conceptual models and modelling grammars (e. g. Desel & Reisig, 1998, Rosemann & van der Aalst, 2007) do not consider intentional or extensional aspects of real world semantics. As these issues are relevant for a meaningful comparison of conceptual models a new formalisation is proposed which separates a modelling grammar from a domain language (Pfeiffer & Niehaves, 2005).

2.1 Conceptual modelling grammars and conceptual models

CMG is a conceptual model grammar iff *C*, *R*, *V*, *G* and *Z* exist such that:

1. $CMG = \langle C, R, V, G, Z \rangle$
2. *C* is a non-empty set of constructs, including object types and relationship types
3. *R* is the set of permitted relations between the constructs with $R \subseteq C \times C$, c represents the incoming construct of the pair $(c, c') \in R$, c' is the outgoing construct
4. *V* is a set of well-formedness rules which restrict the conceptual models of the grammar
5. *G* is a set of graphical symbols
6. *Z* assigns constructs to graphical symbols with $Z \subseteq C \times G$

A *CMG* defines the concrete syntax of a visual conceptual modelling language. A *CMG* without *G* and *Z* represents the abstract syntax of a conceptual modelling language. In the following the terms *CMG* and conceptual modelling language are used synonymous.

The initially mentioned term, domain specific language, can now be formalised. A modelling language is considered domain specific if all constructs have a semantically equal counterpart in the domain language. Formally expressed as: $\forall c \in C, \exists s \in DL : M(D_c) = M(D_s)$. This means a domain specific language does not contain constructs whose semantics is not known in the domain.

CM is a conceptual model iff *E*, *F*, *S*, *A* and *P* exist such that:

1. $CM = \langle E, F, S, A, P \rangle$
2. *E* is a non- empty set of model elements, members of *E* are instantiations of members of *C* with $E \subseteq C \times N$ and *N* as the set of natural numbers
3. *F* is the set of relations between model elements with $F \subseteq E \times E$, e represents the incoming model element of the pair $(e, e') \in F$, e' is the outgoing model element, all undirected edges have the same direction
4. *S* is a set of actual linguistic statements that describe the internal model *IM* with $S \subseteq DL$, the statements consists of technical terms from the application domain
5. *A* assigns technical terms to model elements with $A \subseteq E \times S$
6. *P* defines the position of the graphical elements on the drawing area with $P \subseteq E \times N^2$

Suppose a simplified grammar CMG^{ERM} consisting of entity types (*ET*), relationship types (*RT*) and links (*L*) with $C^{ERM} = \{ET, RT, L\}$. The conceptual model CM^B given in Figure 1 based on CMG^{ERM} can be specified with $CM^B = \langle E^B, F^B, S^B, A^B, P^B \rangle$:

1. $E^B = \{(ET,1),(ET,2),(RT,1),(L,1),(L,2)\}$
2. $F^B = \{((ET,1),(L,1)),((L,1),(RT,1)),((RT,1),(L,2)),((L,2),(ET,2))\}$
3. $S^B = \{Writer, writes, Book\}$
4. $A^B = \{((ET,1),Writer),((ET,2),Book),((RT,1),writes)\}$
5. $P^B = \{((ET,1),(1,1)),((ET,2),(5,1)),((RT,1),(3,1)),((L,1),(2,1)),((L,2),(4,1))\}$

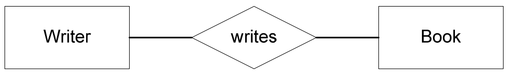


Figure 1. The conceptual model CM^B.

2.2 Modelling rules

For a proper representation of the internal model *IM* with $\{IM\} = M(CM_{IM})$ the conceptual model CM_{IM} must be adequate and complete. This requires that the modelling language and the domain language are comprehensive enough to describe *IM*. Therefore, necessary conditions for $\{IM\} = M(CM_{IM})$ can be formulated:

(R1) All elements of *IM* must be describable with constructs of the modelling language:
$\forall \varepsilon \in IE, \exists c \in C : \varepsilon \in M(D_c)$

(R2) All relations within the internal model *IM* must be describable in terms of permitted relations between constructs of the modelling language:
$\forall (\varepsilon, \varepsilon') \in IR, \exists (c, c') \in R : \varepsilon \in M(D_c) \wedge \varepsilon' \in M(D_{c'})$

(R3) For all elements of *IM* there is an equipollent statement within the domain language:
$\forall \varepsilon \in IE, \exists s \in DL : \{\varepsilon\} = M(D_s)$

If the conditions R1 to R3 are fulfilled then the modelling language and the domain language are called *applicable*. That means *CMG* and *DL* can be used to explicate the internal model *IM*.

For a more convenient presentation some abbreviations are useful. The type of a model element $e \in E$ is its corresponding construct $c \in C$. The function $\tau : E \rightarrow C$, $\tau(e) = \tau((c,k)) = c$ provides the type of a model element. The auxiliary relation $\Psi \subseteq IE \times E$ establishes an one to one mapping between elements of the internal model *IM* and elements of the conceptual model *CM*. $(\varepsilon, e) \in \Psi$ holds iff: $(\forall \varepsilon' \in IE : (\varepsilon', e) \in \Psi \rightarrow \varepsilon' = \varepsilon) \wedge (\forall e' \in E : (\varepsilon, e') \in \Psi \rightarrow e' = e)$.

In order to preserve the meaning and structure of *IM* during the explication the following conditions are required:

(R4) A model element $e \in E$ refers to exactly one element of the internal model $\varepsilon \in IE$ and its corresponding construct is able to describe ε:
$e \in E \leftrightarrow \exists \varepsilon \in IE : (\varepsilon, e) \in \Psi \wedge \varepsilon \in M(D_{\tau(e)})$

(R5) Each relation between model elements $f \in F$ is assigned to exactly one relation between elements of the internal model $\rho \in IR$ and the modelling grammar permits the relation:
$(e, e') \in F \leftrightarrow \exists (\varepsilon, \varepsilon') \in IR : (\varepsilon, e) \in \Psi \wedge (\varepsilon', e') \in \Psi \wedge (\tau(e), \tau(e')) \in R$

(R6) A domain statement is part of the conceptual model *CM* iff it can be assigned to a model element:
$s \in S \leftrightarrow \exists e \in E : (e, s) \in A$

(R7) A domain statement is assigned to a model element iff the domain statement exactly describes the corresponding element of the internal model, the construct associated with the model element has a more general meaning (larger extension) than the domain statement, and no other domain statement is already connected with the model element:
$(e, s) \in A \leftrightarrow \exists \varepsilon \in IE : (e, \varepsilon) \in \Psi \wedge \{\varepsilon\} = M(D_s) \wedge M(D_{\tau(e)}) \supset M(D_s) \wedge [\forall s' \in S : (e, s') \in A \rightarrow s' = s]$

From a set theoretic perspective the modelling language constructs do not have any impact on the extensional semantics of the conceptual model. $\varepsilon \in M(D_{\tau(e)})$ (R4) and $\{\varepsilon\} = M(D_s)$ (R7) show that *s* is more general than $c = \tau(e)$. $M(D_{\tau(e)}) \supset M(D_s)$ (R7) ensures that the relationship is a strict one. That means that the modelling language construct *c* is redundant from an extensional point of view. The value of *c* within the model is an intentional one. The construct *c* emphasises a certain aspect of *s* and thus helps to structure the domain. For example a modelling construct “entity type” instantiated with the domain statement “colour” tells that colour is considered as an object on its own and not as an attribute. However, this information has no influence on the extension of the domain statement colour. The extension is still blue, green, red, and so on. Without the condition $M(D_{\tau(e)}) \supset M(D_s)$ the construct would lose its role as a structuring element and the domain statement would take over this job. However, this would destroy the original function of a construct.

3 COMPARISON OF CONCEPTUAL MODELS

So far conceptual models have been mainly considered as spin-off products of the software development process or of reengineering projects. However, as they contain valuable domain knowledge that is often not explicated elsewhere, they are more and more regarded as an important artefact on their own and not just seen as an intermediate result to come to a database schema or a workflow implementation. The consequence is that the models are not created for a single purpose anymore but have a lifecycle in which they are modified and extended to keep up with the changes in the environment. The definition of operations on conceptual models like transformation, integration or comparison helps to address this issue (Bernstein et al., 2000).

The operations can be divided into syntactical and semantical ones. From a syntactical perspective conceptual models share many similarities with graphs. A graph $G = \langle N^G, E^G \rangle$ comprises a set of nodes N^G and edges E^G (Diestel, 2000). The nodes N^G show an analogous structure as the model elements E and the edges E^G can be considered to be allied with the relations F. From a semantic point of view conceptual models are different from graphs as they obtain their meaning partially from the modelling language constructs C and the domain statements S. In addition to graphs conceptual models have a concrete form of representation G and its elements have a position on a drawing sheet P. However, representational aspects of conceptual models are disregarded in the following.

3.1 Syntactical and Semantical Equivalence

Many syntactical operations on conceptual models can be led back to operations on graphs. For example there are publications on search, comparison and transformation (Bardohl et al., 1999, Jilani et al., 2001). Existing notions of equivalence (e. g. Milner, 1980, Pomello et al., 1992, Rahm & Bernstein, 2001, van der Aalst et al., 2006, van Glabbeek & Weijland, 1996) lack in the consideration of real world semantics which, however, is needed in order to modify the algorithms for graphs to fit the specific properties of conceptual models. Hence, a new notion of equivalence is given in the following.

Before two entire models can be compared it is necessary to define what it means when two model elements are syntactically equivalent. Considering the models $CM = \langle E, F, S, A, P \rangle$ and $CM' = \langle E', F', S', A', P' \rangle$ two model elements are syntactically equivalent if they share the same type and have a syntactically identical domain statement associated. The syntactical equivalence is represented by the relation: $\Omega \subseteq E \times E'$. $(e, e') \in \Omega$ iff:

(S1) $\tau(e) = \tau(e')$

(S2) $\forall s \in S : (e, s) \in A \rightarrow (e', s) \in A'$

(S3) $\forall s' \in S' : (e', s') \in A' \rightarrow (e, s') \in A$

Two conceptual models are only equivalent if every model element of the first model maps to exactly one element of the second model and vice versa. This is expressed by the relation $\Omega' \subseteq E \times E'$ with $\Omega' \subseteq \Omega$. $(e, e') \in \Omega'$ iff:

(S4) $\forall e'' \in E : (e'', e') \in \Omega \rightarrow e'' = e$

(S5) $\forall e''' \in E' : (e, e''') \in \Omega \rightarrow e''' = e'$

Two conceptual models $CM = \langle E, F, S, A, P \rangle$ and $CM' = \langle E', F', S', A', P' \rangle$ are syntactically equivalent ($CM =_{syn} CM'$) if all elements of the two models have exactly one corresponding element and all elements take part in a relation map. $CM =_{syn} CM'$ iff

(S6) $\forall e \in E, \exists e' \in E' : (e, e') \in \Omega'$

(S7) $\forall e' \in E', \exists e \in E : (e, e') \in \Omega'$

(S8) $\forall (e, e'') \in F, \exists (e', e''') \in F' : (e, e') \in \Omega' \wedge (e'', e''') \in \Omega'$

(S9) $\forall (e', e''') \in F', \exists (e, e'') \in F : (e, e') \in \Omega' \wedge (e'', e''') \in \Omega'$

A semantic comparison requires, in addition to the syntactic one, that the meanings of the constructs as well as the domain statements are considered. So far there is no algorithm that allows for a semantic comparison of conceptual models (Pfeiffer & Gehlert, 2005), as there is not automated way to identify the extension of an arbitrary description. Consequently, semantical model comparison is a manual activity that has to be performed by the members of the corresponding language communities LC_{DL} and LC_{CMG}. These competent and willing persons must come to a consensus that two concepts or two domain statements are the same. In this case, based on the consensus theory of truth, the two statements or concepts can be considered semantically equivalent (Kamlah & Lorenzen, 1984).

Two conceptual models $CM = \langle E, F, S, A, P \rangle$ and $CM' = \langle E', F', S', A', P' \rangle$ are semantically equivalent ($CM =_{sem} CM'$) if they are syntactically equivalent, the types of the corresponding model elements have identical semantics and the associated domain statements share the same meaning. $CM =_{sem} CM'$ iff:

(S10) $CM =_{syn} CM'$

(S11) $\forall (e, e') \in \Omega : M(D_{\tau(e)}) = M(D_{\tau(e')})$

(S12) $\forall (e, e') \in \Omega, \forall s \in S, \forall s' \in S : (e, s) \in A \wedge (e', s') \in A' \rightarrow M(D_s) = M(D_{s'})$

3.2 Model comparison conflicts

There are a couple of conflicts that affect the analysis results when two models are syntactically or semantically compared (taxonomies of these conflicts can be found for example in Davis et al., 2003, Hakimpour & Geppert, 2001, Kashyap & Sheth, 1996, Lawrence & Barker, 2001). These conflicts are exemplarily demonstrated in Figure 2 in the ERM notation.

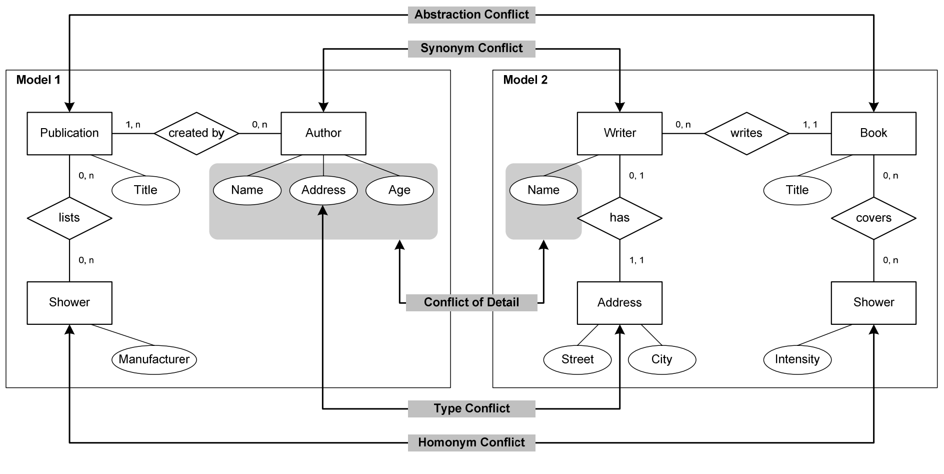


Figure 2. Examples of important model comparison conflicts (Pfeiffer & Gehlert, 2005).

Type conflicts arise whenever the same fact of the application domain is represented by using different constructs of the modelling language. They result if there are choices in the modelling language about what construct is to be used in a certain situation. There is a type conflict between a model $CM_{IM}^{CMG}\langle E,F,S,A,P\rangle$ and a model $CM_{IM}'^{CMG}\langle E',F',S',A',P'\rangle$ iff:

(C1) $\exists e\in E,\exists e'\in E',\exists s\in S: s\in S'\wedge(e,s)\in A\wedge(e',s)\in A'\wedge\tau(e)\neq\tau(e')$

Synonym conflicts occur when two different domain statements have the same meaning. There is a synonym conflict between a model $CM_{IM}\langle E,F,S,A,P\rangle$ and a model $CM_{IM}'\langle E',F',S',A',P'\rangle$ iff:

(C2) $\exists s\in S,\exists s'\in S': s\neq s'\wedge M(D_s)=M(D_{s'})$

Homonym conflicts emerge due to domain statements which have more than one meaning. This is the case if for one domain statement there is a different, adequate and complete description with a varying extension. There is a homonym conflict between a model $CM_{IM}\langle E,F,S,A,P\rangle$ and a model $CM_{IM'}'\langle E',F',S',A',P'\rangle$ iff:

(C3) $\exists s\in S: s\in S'\wedge M(D_s)\neq M(D_s')$

Abstraction conflicts result from the representation of the application domain at deviating levels of abstraction. Different modellers use more general or more precise domain statements for the same fact. There is an abstraction conflict between a model $CM_{IM}\langle E,F,S,A,P\rangle$ and a model $CM_{IM'}'\langle E',F',S',A',P'\rangle$ iff:

(C4) $\exists s\in S,\exists s'\in S': s\neq s'\wedge\left(M(D_s)\supset M(D_{s'})\vee M(D_s)\subset M(D_{s'})\right)$

Type conflicts, synonym conflicts and abstraction conflicts lead to an underestimation of the semantic similarity of two models. Homonym conflicts can cause an overestimation of the similarity.

4 SOLVING THE CONFLICTS

The general approach of this paper is to solve the model comparison conflicts through the adoption of rules for conceptual modelling grammars which simplify the model comparison process. Type conflicts can for example be avoided, if all modelling language constructs are required to be semantically disjoint.

Proposition 1 (Type Conflicts): With $\forall c,c'\in C, c\neq c': M(D_c)\cap M(D_{c'})=\varnothing$ type conflicts between the models CM_{IM}^{CMG} and $CM_{IM}'^{CMG}$ are not feasible.

Proof: If it is possible to show that from $\forall c,c'\in C, c\neq c': M(D_c)\cap M(D_{c'})=\varnothing$ follows that $(e,s)\in A\wedge(e',s)\in A'\rightarrow\tau(e)=\tau(e')$ type conflict cannot arise. In order that $(e,s)\in A$ holds it is necessary that: $M(D_{\tau(e)})\supset M(D_s)$ (R7). From the condition $\forall c,c'\in C, c\neq c': M(D_c)\cap M(D_{c'})=\varnothing$ follows that: $\forall\dot{e},\ddot{e}\in E,\dot{e}\neq\ddot{e}: M(D_{\tau(\dot{e})})\cap M(D_{\tau(\ddot{e})})=\varnothing$. Thereof one can derive that there is at least one $\hat{c}=\tau(e)$ to meet the condition: $M(D_{\tau(e)})\supset M(D_s)$. Consequently, it must also hold for $(e',s)\in A'$ that: $M(D_{\tau(e')})\supset M(D_s)$. The application of the same modelling grammar leads to the identical: $\hat{c}=\tau(e')$. Hence, it follows that: $\tau(e)=\tau(e')$ if $(e,s)\in A$ and $(e',s)\in A'$. □

Homonym and synonym conflicts can be eliminated if these language defects are removed from the domain language.

Proposition 2 (Synonym Conflicts): With $\forall s,s'\in DL, s\neq s': M(D_s)\neq M(D_{s'})$ synonym conflicts between the models $CM_{IM}^{CMG,DL}$ and $CM_{IM}'^{CMG,DL}$ are not feasible.

Proof: From the definition of a *CM* it follows that $S\subseteq DL$. Consequently, if there are no synonyms in *DL* there are also no synonym conflicts caused by *S*. □

Proposition 3 (Homonym Conflicts): With $\forall s\in DL: M(D_s)=M(D_s')$ homonym conflicts between the models $CM_{IM}^{CMG,DL}$ and $CM_{IM'}'^{CMG,DL}$ are not feasible.

Proof: If there are no homonyms in *DL* there are also no homonym conflicts caused by *S*. □

Subsequently, the implications of proposition 1 and proposition 2 on a syntactical model comparison are evaluated. It is claimed that if the same internal model coupled with an identical modelling language as well as the same domain language are employed, and according to the propositions all synonyms are eliminated as well as all modelling language constructs are disjoint, then two syntactically equivalent models arise. This means that different modellers who share the same internal model will come to a syntactically identical result.

Proposition 4 (Syntactical Equivalence): R1- R7 and

1. $\forall c,c' \in C, c \neq c' : M(D_c) \cap M(D_{c'}) = \varnothing$
2. $\forall s,s' \in DL, s \neq s' : M(D_s) \neq M(D_{s'})$

imply that $CM_{IM}^{CMG,DL} =_{syn} CM'^{CMG,DL}_{IM}$.

Proof: $\forall c,c' \in C, c \neq c' : M(D_c) \cap M(D_{c'}) = \varnothing$ and R1 imply that: $\forall \varepsilon \in IE, \exists c \in C : \varepsilon \in M(D_c)$ and $\forall \varepsilon \in IE, \neg\exists c' \in C, c' \neq c : \varepsilon \in M(D_{c'})$. Thus, c is the only construct in C that can describe ε. R3 and $\forall s,s' \in DL, s \neq s' : M(D_s) \neq M(D_{s'})$ analogical imply that: $\forall \varepsilon \in IE, \exists s \in S : \{\varepsilon\} = M(D_s)$ and $\forall \varepsilon \in IE, \neg\exists s' \in S, s' \neq s : \{\varepsilon\} \in M(D_{s'})$. Thus, s is the only domain statement in DL that can describe the element of the internal model $\varepsilon \in IE$. Consequently, each ε is associated with exactly one pair (c,s) which can be used to represent it. Because of R4 and the properties of Ψ for each ε exactly one e is instantiated with $\tau(e) = c$. Thus, for each ε the pair (c,s) can be extended to a triple (e,c,s).

1. If $M(D_c) \subset M(D_s)$ then because of R7 e is labelled with s. This is expressed by the relation $(e,s) \in A$. Because of R6 it follows that: $s \in S$.
2. If $M(D_c) = M(D_s)$ then because of R7 and R6 $(e,s) \notin A$ and $s \notin S$.
3. $M(D_c) \supset M(D_s)$ contradicts R1 and R3.

For each $\dot{\varepsilon}$ there is also only one corresponding $\dot{c}$ which is instantiated as $\dot{e}$. Due to R5 and R2 for each $(\varepsilon, \dot{\varepsilon}) \in IR$ exactly one $(e, \dot{e}) \in F$ is created.

In the case of two models $CM_{IM}^{CMG,DL}$ and $CM'^{CMG,DL}_{IM}$ for each $\varepsilon \in IM$ there is exactly one triple (e,c,s) with $e \in E$ and exactly one triple (e',c',s') with $e' \in E'$ (R4). Because *CMG* and *DL* are the same for both models, it follows that $c = c'$ and $s = s'$. Because of S1-S3 from $c = c'$ and $s = s'$ follows that: $(e,e') \in \Omega$. S4-S5 are fulfilled because (e,c,s) and (e',c',s') map to the same ε and there is no other $\hat{e}$ which refers to ε. For each $\rho \in IR$ exactly one $f \in F$ and exactly one corresponding $f' \in F'$ exist. As ε and ρ are arbitrary elements or relations of *IM* the conditions S6-S9 are fulfilled. It follows that: $CM_{IM}^{CMG,DL} =_{syn} CM'^{CMG,DL}_{IM}$. □

Based on proposition 4 the consequences of proposition 3 on the semantic model comparison can be analysed. For that purpose the idea of proposition 3 is transferred to the constructs of conceptual modelling grammar. Also, in the set of constructs there must not be homonyms: $\forall c \in C : M(D_c) = M(D'_c)$. Suppose two models which were created with the same modelling language and an identical domain language. It is claimed that if these models are syntactically equivalent and neither the domain language nor the modelling language contain homonyms then the two models are also semantically equivalent.

Proposition 5 (Semantical Equivalence):

1. $CM^{CMG,DL} =_{syn} CM'^{CMG,DL}$
2. $\forall c \in C : M(D_c) = M(D'_c)$
3. $\forall s \in DL : M(D_s) = M(D'_s)$
4. imply that $CM^{CMG,DL} =_{sem} CM'^{CMG,DL}$.

Proof: From $\forall c \in C : M(D_c) = M(D'_c)$ follows that: $c = c'$ implies $M(D_c) = M(D_{c'})$ because both models apply the same modelling language. From $\forall s \in DL : M(D_s) = M(D'_s)$ it follows that: $s = s'$

implies $M(D_s) = M(D_{s'})$, because both models apply an identical domain language. Every $e \in E$ belongs to a triple (e,c,s), each $e' \in E'$ is part of (e',c',s'). The syntactical equivalence of $CM^{CMG,DL} =_{syn} CM'^{CMG,DL}$ implies that: $(e,e') \in \Omega$. Consequently, it follows that $\tau(e) = \tau(e')$ (S1) and where applicable $s = s'$ (S2, S3). Thus, for every pair $(e,e') \in \Omega$ it holds that: $M(D_{\tau(e)}) = M(D_{\tau(e')})$ (S11) and $M(D_s) = M(D_{s'})$ (S12). Hence, it follows: $CM^{CMG,DL} =_{sem} CM'^{CMG,DL}$. □

Proposition 4 and Proposition 5 have important consequences. They assure that under the conditions:

(D1) $\forall c,c' \in C, c \neq c' : M(D_c) \cap M(D_{c'}) = \varnothing$

(D2) $\forall s,s' \in DL, s \neq s' : M(D_s) \neq M(D_{s'})$

(D3) $\forall c \in C : M(D_c) = M(D'_c)$

(D4) $\forall s \in DL : M(D_s) = M(D'_s)$

a semantic model comparison can be traced back to a syntactical model comparison. Starting from two identical internal models two semantically and syntactically equivalent models can be explicated. It is not necessary to apply a semantical operation on the conceptual models as a syntactical operation is sufficient. This in turn implies that if D1-D4 hold then the semantic model comparison can be implemented with operations on graphs and thus can be completely automated.

5 COMPARABLE DOMAIN SPECIFIC LANGUAGES

The implementation of the conditions D1-D4 provides a different challenge. Conditions D1 and D3 refer to the modelling grammar. A modelling grammar is an artificial artefact created by a method engineer. Hence, the method engineer can freely modify the grammar *CMG* such that it complies with D1 and D3. Constraints D2 and D4, however, bear on the domain language. The domain language *DL* is naturally grown and cannot be easily adjusted as it is the shared property of the language community LC_{DL}. The language community decides on how *DL* is used. One possibility to cope with this problem is to reconstruct *DL* in form of the language DL' and to eliminate all homonyms and synonyms during that process. This language DL' could for example be represented by a domain ontology (Guizzardi et al., 2002). Subsequently, it is necessary to oblige the modeller by additional rules or tool support to apply only the vocabulary of DL' in order to explicate an internal model. All occurrences of *DL* within the modelling rules have to be replaced by DL'.

However, there is an alternative approach to meet the conditions D2 and D4. The relevant statements from DL' can be transformed into modelling language constructs and added to *C*. A domain specific language emerges. In parallel the sets *S* and *A* are required to be empty in order to ensure that no ambiguous domain statements from *DL* can be added. A resulting conceptual model *CM* has the following form: $CM = \langle E,F,\varnothing,\varnothing,P \rangle$. Based on the grammar *CMG* a modified grammar $CMG' = \langle C',R',V',G',Z' \rangle$ evolves with $C' \subseteq C \cup DL'$ and $\forall c' \in C', \exists s \in DL : M(D_{c'}) = M(D_s)$. The multi-purpose grammar *CMG* is transformed into a domain specific language CMG'. The drawback of this modification is that CMG' loses the flexibility to be used in an arbitrary domain but is now rather specific to a particular knowledge area.

A domain specific language *CMG* that meets the constraints D1 and D3 provides the following advantages:

1. *Abstraction conflicts are avoided.* Condition D1 is stricter than constraint D2, as D2 just demands for the elimination of synonyms within the domain language but D1 additionally requires the modelling constructs to be semantically disjoint. Thus, it is not possible to have more general and more specific modelling constructs within *C* simultaneously. There cannot be two differently abstract constructs that refer to the same element of the internal model $\varepsilon \in IE$. Therefore, the constraint D1 prevents abstraction conflicts as described in C4. If the modelling grammar *CMG* is declared as mandatory in a certain project because of R1-R3 it is necessary that the

internal models of all modellers share the same level of abstraction. To put it in other words, *CMG* ensures that different modellers represent the reality in an identically abstract manner.

2. *Semantically meaningful operations can be defined at the design time of the CMG.* When domain statements become modelling language constructs, the modelling language does not only provide measures to structure the domain but also it is sufficient to describe it. The use of additional statements from the domain language *DL* is no longer required. With a multi-purpose language the domain specific terms are not part of the modelling language but are added when the conceptual model is constructed. Thus, multi-purpose languages allow for the definition of semantically meaningful operations after the models are constructed. However, with domain specific languages this can already be done at the design time of the language as the domain statements are part of the grammar. Although, domain specific languages are overall less general than multi-purpose languages from the perspective of their semantic operations, they are more widely reusable. For example, an UML diagram can be examined for how many activities it comprises. This is no domain specific analysis though. Alternatively, suppose a domain specific language with the construct "Enter data into IT". With this language it is possible to count how often paper documents are digitised. Hence, consequences for the introduction of a document management system can be derived. With UML such an analysis could be defined based on a set of existing models but not with the language alone. Moreover, contrary to the domain specific language with UML the conflicts C1-C4 had to be addressed.

Beneath the elimination of the conflicts C1-C4 a domain specific language *CMG* that fulfils the conditions D1 and D3 also simplifies the specification of other semantic operations. Such a domain specific language *CMG* is therefore called *operational*.

The applicability of a domain specific language is limited by the dissemination of its particular underlying domain language. Therefore, domain specific languages are especially useful in areas where a common terminology has already been established. Due to legal regulations and a strictly hierarchical organisation structure, the public sector possesses a largely consolidated domain language. In the next section we present an operational domain specific language from this area.

6 PICTURE - AN OPERATIONAL DOMAIN SPECIFIC LANGUAGE

PICTURE is a domain specific language for the efficient representation of the process landscape in public administrations (Becker et al., 2006). With PICTURE processes are modelled as a sequential flow of domain specific process building blocks. A process building block represents a predefined set of activities within an administrational process (Rupprecht et al., 2000). The semantics of a process building block is defined by a corresponding domain statement which is part of the modelling language. The process building blocks can be further described with the aid of attributes. The attributes collect additional information about a process as basis for a subsequent semantical analysis. PICTURE contains altogether 29 process building blocks and more than 50 attributes. An example process with process building bocks and attributes is shown in Figure 3.

Based on the example in Figure 3 $C^{PICTURE}$ results as $C^{PICTURE} \supseteq \{ID, CD, FA, FD, D, S, SM, T, TM, L\}$. The construct L stands for the links between process building blocks and corresponds to the domain statement "is followed by". The complete PICTURE-grammar contains additional building blocks, attributes, and attribute values.

The PICTURE-language has been constructed in consideration of the conditions D1 and D3. It has been strived for a set of modelling constructs whose members are semantically disjoint (cf. D1) and do not comprise homonyms (cf. D3). The constructs have been chosen based on an analysis of existing process models from the public administration domain and an evaluation of electronic government literature. As all PICTURE-constructs correspond with language statements from the public sector, they are domain specific. Attributes have been added with regard to the effects certain basic technologies, such as document management systems or virtual post offices, exert on specific properties of the processes. Although, the PICTURE constructs have been specifically developed for public administrations some of them may also be useful for administrational processes in other domains.

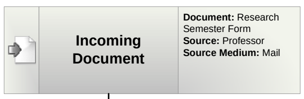
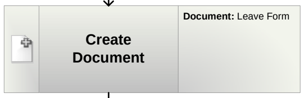
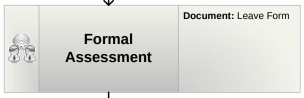
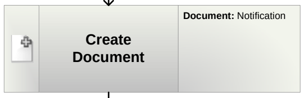
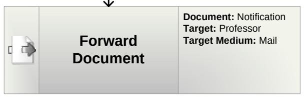
Process	Process Building Blocks	Attributes
 Incoming Document **Document:** Research Semester Form **Source:** Professor **Source Medium:** Mail	Incoming Document (ID)	Document (D) Source (S) Source Medium (SM)
 Create Document **Document:** Leave Form	Create Document (CD)	Document (D)
 Formal Assessment **Document:** Leave Form	Formal Assessment (FA)	Document (D)
 Create Document **Document:** Notification	Create Document (CD)	Document (D)
 Forward Document **Document:** Notification **Target:** Professor **Target Medium:** Mail	Forward Document (FD)	Document (D) Target (T) Target Medium (TM)

Figure 3. Example process with process building blocks and attributes.

With the help of process building blocks and attributes the PICTURE-language supports the distributed modelling of the process landscape of a public administration. The resulting process models have the same level of abstraction and can be analysed for structural similarities and building block patterns. For example it is possible to identify so called Ping-Pong processes in which a document alternates between different organisational units multiple times. Furthermore, PICTURE has proven to be an efficient method to model processes in the public administration domain. So far in two large case studies 21 modellers have collected more than 330 processes with PICTURE. In these two projects, the acquisition of the processes took significantly less time than with traditional modelling approaches (Becker et al., 2006).

The PICTURE-language shows that operational domain specific languages are feasible and can be constructed. It also demonstrates that this language class provides the potential for further advanced semantic operations.

7 CONCLUSIONS AND FUTURE RESEARCH

The perspective of the paper is not to take conceptual models as given when they are compared. Rather, we have argued that if the modelling language complies with certain rules then the comparison process can be noticeably simplified. We have proven that a domain specific language, where all constructs are semantically disjoint and homonyms have been eliminated, significantly reduces the ambiguities during the construction of conceptual models. We have introduced the class of operational domain specific languages that prevent the emergence of type, synonym, homonym, and abstraction conflicts. Furthermore, this language class allows for tracing back the semantic model comparison to the syntactic one. With the PICTURE-language we have presented an example of an operational domain specific language and indicated its potential for advanced semantic operations.

As we have taken internal models for granted so far, our research has focused on their explication. However, this is a simplification since deviations between conceptual models are often also due to varying internal models. Not all of the conflicts emerge because of the freedom a modelling or domain language offers. Conflicts of detail occur when modellers represent a certain fact of reality in differently explicit form in their internal models, for example with more attributes or multiple constructs. Disparities between conceptual models can also arise when the borders of a certain phenomenon are not clear-cut and thus inconsistent internal models are created by different modellers. By reversing the conditions R1-R3 a mandatory modelling language can influence the way the internal model is shaped. It is subject to further research to evaluate appropriate language properties to address the remaining conflicts. In the PICTURE-language the elimination of ramifications has pointed out one possibility.

The proposed construction rules for conceptual modelling languages represent a theoretical result that needs further practical evaluation. Languages that meet these criteria have only a limited scope of application and loose the ability to be used in situations where different abstraction levels are needed or a flexible use of domain language is required. It is due to further research to analyse how some of the criteria can be relaxed without increasing the comparison efforts.

The rules of disjoint constructs and the exclusion of homonyms show similarities with construct-redundancy and construct-overload as derived from the BUNGE-WAND-WEBER ontology (Wand, 1996). It is up to further investigations to evaluate the connections between these research streams.

Acknowledgements

The work published in this paper is partly funded by the European Commission through the STREP PICTURE. It does not represent the view of European Commission or the PICTURE consortium and the authors are solely responsible for the paper's content.

References

Balzer, W., Moulines, C. U. and Sneed, J. D. (1987). An Architectonic for Science - The Structuralist Program. D. Reidel Publishing Company, Dordrecht et al.

Bardohl, R., Minas, M., Schürr, A. and Taentzer, G. (1999). Application of Graph Transformation to Visual Languages. In Handbook of Graph Grammars and Computing by Graph Transformation: Applications, Languages and Tools (Ehrig, H., Engels, G., Kreowski, H.-J. and Rozenberg, G. Eds.), World Scientific Publishing, Singapore.

Becker, J., Algermissen, L., Falk, T., Pfeiffer, D. and Fuchs, P. (2006). Model Based Identification and Measurement of Reorganization Potential in Public Administrations – the PICTURE-Approach. In Proceedings of the Tenth Pacific Asia Conference on Information Systems (PACIS 2006), pp. 860-875, Kuala Lumpur, Malaysia.

Bernstein, P. A., Halevy, A. Y. and Pottinger, R. A. (2000). A vision for management of complex models. SIGMOD Record (ACM Special Interest Group on Management of Data), 29 (4), pp. 55-63.

Chen, P. P.-S. (1976). The Entity Relationship Model - Toward a Unified View of Data. ACM Transaction on Database Systems, 1 (1), pp. 9-36.

Davis, I., Green, P., Milton, S. and Rosemann, M. (2003). Using Meta Models for the Comparison of Ontologies. In Proceedings of the Eighth CAiSE/IF IP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, Austria.

Desel, J. and Reisig, W. (1998). Place/Transition Petri Nets. In Lectures on Petri Nets I: Basic Models (Reisig, W. and Rozenberg, G. Eds.), pp. 122-173, Springer, Berlin.

Diestel, R. (2000). Graph Theory. Springer, New York.

Evermann, J. and Wand, Y. (2001). Towards Ontologically Based Semantics for UML Constructs. In Proceedings of the 20th International Conference on Conceptual Modeling (ER 2001) (Kunii, H. S., Jajodia, S. and Sølvberg, A. Eds.), pp. 354-367, Yokohama, Japan.

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

Hadar, I. and Soffer, P. (2006). Variations in Conceptual Modeling: Classification and Ontological Analysis. Journal of the AIS, 7 (8), pp. 568-592.

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

Jilani, L. L., Desharnais, J. and Mili, A. (2001). Defining and Applying Measures of Distance Between Specifications. IEEE Transactions on Software Engineering, 27 (8), pp. 673-703.

Kamlah, W. and Lorenzen, P. (1984). Logical Propaedeutic. Pre-School of Reasonable Discourse. University Press of America, Lanham, MD.

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

Lawrence, R. and Barker, K. (2001). Integrating relational database schemas using a standardized dictionary. In Proceedings of the 16th ACM Symposium on Applied Computing, Las Vegas, USA.

Milner, R. (1980). A Calculus of Communicating Systems. Springer, Berlin.

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

Object Management Group (2004). UML 2.0 Superstructure Specification. Downloaded from http://www.omg.org/cgi-bin/doc?formal/05-07-04 on 2006-Apr-30.

Patig, S. (2004). Measuring Expressiveness in Conceptual Modeling. In Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE 2004), Riga, Latvia.

Pfeiffer, D. and Gehlert, A. (2005). A framework for comparing conceptual models. In Proceedings of the Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2005), pp. 108-122, Klagenfurt, Austria.

Pfeiffer, D. and Niehaves, B. (2005). Evaluation of Conceptual Models - A Structuralist Approach. In Proceedings of the 13th European Conference on Information Systems (ECIS 2005), Regensburg, Germany.

Pomello, L., Rozenberg, G. and Simone, C. (1992). A Survey of Equivalence Notions for Net Based Systems. In Lecture Notes in Computer Science, Vol. 609; Advances in Petri Nets 1992 (Rozenberg, G. Ed.), pp. 410-472, Springer, Berlin.

Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching.

Rosemann, M. (2003). Preparation of Process Modeling. In Process Management (Becker, J., Kugeler, M. and Rosemann, M. Eds.), pp. 41-78, Berlin et al.

Rosemann, M. and van der Aalst, W. M. P. (2007). A configurable reference modelling language. Information Systems, 32 (1), pp. 1-23.

Rossi, M., Ramesh, B., Lyytinen, K. and Tolvanen, J.-P. (2004). Managing Evolutionary Method Engineering by Method Rationale. Journal of the Association for Information Systems, 5 (9), pp. 356-391.

Rupprecht, C., Funffinger, M., Knublauch, H. and Rose, T. (2000). Capture and Dissemination of Experience about the Construction of Engineering Processes. In Proceedings of the 12th International Conference on Advanced Information Systems Engineering (CAiSE2000) (Wangler, B. and Bergman, L. Eds.), pp. 294-308, Stockholm, Sweden.

Schütte, R. and Rotthowe, T. (1998). The Guidelines of Modeling - An Approach to Enhance the Quality in Information Models. In Proceedings of the 17th International Conference on Conceptual Modeling (ER 1998) (Ling, T. W., Ram, S. and Lee, M. L. Eds.), pp. 240-254 Singapore.

van der Aalst, W. M. P., Alves de Medeiros, A. K. and Weijters, A. J. M. M. (2006). Process Equivalence: Comparing Two Process Models Based on Observed Behavior. In Proceedings of the 4th International Conference on Business Process Management (BPM2006) (Dustdar, S., Fiadeiro, J. L. and Sheth, A. P. Eds.), pp. 129-144, Vienna, Austria.

van Glabbeek, R. J. and Weijland, W. P. (1996). Branching Time and Abstraction in Bisimulation Semantics. Journal of the ACM 43 (3), pp. 555-600.

Wand, Y. (1996). Ontology as a foundation for meta-modelling and method engineering. Information and Software Technology, 38 (1996), pp. 281-287.