Automatic Knowledge Retrieval from Conceptual Models

Jörg Becker, Daniel Pfeiffer

European Research Center for Information Systems, Leonardo-Campus 3,
48149 Münster, Germany
{becker, pfeiffer}@ercis.de

Abstract. Conceptual models are an important repository for knowledge in companies and public institutions. The retrieval of this knowledge can prepare reorganisations projects and support IT investment decisions. However, so far this information source has hardly been utilized in automated analyses. We argue that if modelling languages are endowed with specific characteristics the resulting models can be analysed in an automatic manner. We formally show that with such languages: (1) type, synonym, homonym, and abstraction conflicts are eliminated as well as (2) the identification of semantically equivalent model elements can be traced back to finding syntactic ones.

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

1 Introduction

Conceptual models are an important knowledge source for business decisions. They are used in organisational design to describe the business objects, the processes, and organisational structures of a company or public institution [1-3]. Conceptual models contain information about the flow of activities, the resulting products and services, the required data, as well as the involved organisational units. A detailed analysis of conceptual models can, therefore, help to asses and improve the efficiency of an organisation.

Automated knowledge retrieval from conceptual models significantly increases their value in practice. So far the identification of reorganisation potential and efficiency indicators has mainly been performed manually with high financial efforts. Approaches for an analysis of conceptual models in an automatic manner as in other domains, for example based on data warehouses or websites (e. g. [4, 5]), are missing.

Knowledge retrieval from conceptual models in an automated manner requires specific modelling language characteristics. To perform an analysis with models that exhibit an arbitrary structure hampers the identification of semantically meaningful results [6]. The aim of this paper is to show which modelling language characteristics are required, in order to identify semantically relevant model elements in an automated manner. We will explain how these language properties foster the retrieval of significant knowledge to support decision making.

The remainder of this paper will proceed as follows: In the next section we will provide the basic vocabulary to discuss the retrieval of knowledge from conceptual models. 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 need for an equivalence notion between model elements as basis for an analysis. Formal definitions of the different conflicts that can emerge during a model analysis are given. Subsequently, these conflicts are solved by specific language properties. The paper closes with a summary of the main results and an outlook to further research.

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 [7, 8]. A conceptual model is the result of an explication of an internal model with a modelling language. 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.

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* [9]. 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. 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*. An element *e* of the conceptual model *CM* stands for a description $D_{\varepsilon}^{CMG,DL}$. 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 [10].

Existing formalisms for conceptual models and modelling grammars (e. g. [11] or [12]) do not consider intentional or extensional aspects of real world semantics. As these issues are relevant for a meaningful analysis of conceptual models a new formalisation is proposed which separates modelling and domain language [13]. Set-theoretic predicates are applied for the formal definitions [9, 14].

2.1 Conceptual modelling grammars and conceptual models

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

- $CMG = \langle C, R, V, G, Z \rangle$

- C is a non-empty set of constructs, including object types and relationship types
- 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
- V is a set of well-formedness rules which restrict the conceptual models of the grammar
- G is a set of graphical symbols
- 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.

Domain specific languages are created to solve problems within a particular area of concern [15]. They are different from general-purpose languages like the Unified Modelling Language (UML) [16] or the Entity Relationship Model (ERM) [17] which do not focus a particular domain. In order to describe a particular domain they apply the specific vocabulary of this part of the world. 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. 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 analyses on the conceptual models can already be defined at the language level.

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

- $CM = \langle E, F, S, A \rangle$
- 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
- 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
- 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
- A assigns technical terms to model elements with $A \subseteq E \times S$

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 Fig. 1 based on CMG^{ERM} can be specified with $CM^B = \langle E^B, F^B, S^B, A^B \rangle$:

- $E^B = \{(ET,1),(ET,2),(RT,1),(L,1),(L,2)\}$
- $F^B = \{((ET,1),(L,1)),((L,1),(RT,1)),((RT,1),(L,2)),((L,2),(ET,2))\}$
- $S^B = \{Writer, writes, Book\}$
- $A^B = \{((ET,1), Writer),((ET,2), Book),((RT,1), writes)\}$

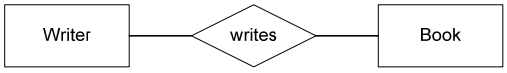

Fig. 1. The conceptual model CM^B.

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:

- All elements of *IM* must be describable with constructs of the modelling language:

$$\forall \varepsilon \in IE, \exists c \in C : \varepsilon \in M(D_c) \quad \textbf{(R1)}$$

- All relations within the internal model *IM* must be describable in terms of permitted relations between constructs of the modelling language:

$$\forall (\varepsilon, \varepsilon') \in IR, \exists (c, c') \in R : \varepsilon \in M(D_c) \wedge \varepsilon' \in M(D_{c'}) \quad \textbf{(R2)}$$

- For all elements of *IM* there is an equipollent statement within the domain language:

$$\forall \varepsilon \in IE, \exists s \in DL : \{\varepsilon\} = M(D_s) \quad \textbf{(R3)}$$

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 $\tau : E \to C$, $\tau(e) = \tau((c,k)) = c$ 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: $\left(\forall \varepsilon' \in IE : (\varepsilon', e) \in \Psi \to \varepsilon' = \varepsilon\right) \wedge \left(\forall e' \in E : (\varepsilon, e') \in \Psi \to e' = e\right)$.

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

- 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:

$$e \in E \leftrightarrow \exists \in IE : (\varepsilon, e) \in \Psi \wedge \varepsilon \in M(D_{\tau(e)}) \quad \textbf{(R4)}$$

- 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:

$$(e, e') \in F \leftrightarrow \exists (\varepsilon, \varepsilon') \in IR : (\varepsilon, e) \in \Psi \wedge (\varepsilon', e') \in \Psi \wedge (\tau(e), \tau(e')) \in R \quad \textbf{(R5)}$$

- A domain statement is part of the conceptual model *CM* if it can be assigned to a model element:

$$s \in S \leftrightarrow \exists e \in E : (e, s) \in A \quad \textbf{(R6)}$$

- A domain statement is assigned to a model element if 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:

$$(e,s) \in A \leftrightarrow \exists \varepsilon \in IE : (e,\varepsilon) \in \Psi \wedge \{\varepsilon\} = M(D_s) \wedge \quad \textbf{(R7)}$$
$$M(D_{\tau(e)}) \supset M(D_s) \wedge \left[\forall s' \in S : (e,s') \in A \rightarrow s' = s\right]$$

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.

These formalisations are used in the following to propose and prove mechanisms for the elimination of semantic analysis conflicts. Such mechanisms are required in order to reach the objective of this paper to enable the analysis of conceptual models in an automated manner.

3 Semantic Analysis of Conceptual Models

An analysis of conceptual models requires that certain reoccurring element structures can be identified in the models. Based on a match between model elements and predefined semantic patterns, meaningful statements about the domain can be derived. Thus, the identification of semantically equivalent conceptual model elements is a prerequisite for a semantic analysis aiming at knowledge retrieval. As semantic patterns represent sections from models, in the following they will be considered as conceptual models themselves. When a conceptual model *CM* is searched for the pattern *CM'* then the elements of both artefacts must be compared.

3.1 Syntactical and Semantical Equivalence of Model Elements

The comparison of conceptual model elements can be divided into a syntactical and semantical one [18]. Existing notions of equivalence (e. g. [19-23]) lack in the consideration of real world semantics which, however, is important for a meaningful analysis of a conceptual model.

Two model elements $e \in E$ and $e' \in E'$ are syntactically equivalent ($e =_{syn} e'$) if they share the same type and have a syntactically identical domain statement associated. $e =_{syn} e'$ iff:

- $\tau(e) = \tau(e')$ **(S1)**
- $\forall s \in S : (e,s) \in A \rightarrow (e',s) \in A'$ **(S2)**
- $\forall s' \in S' : (e',s') \in A' \rightarrow (e,s') \in A$ **(S3)**

Semantically equivalent model elements require, in addition to the syntactic ones, 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 model elements [18]. There is no automatic way to identify the extension of an arbitrary description. Consequently, semantic 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 [10] the two statements or concepts can be considered semantically equivalent.

Two model elements $e_{\varepsilon}^{CMG,DL}$ and $e_{\varepsilon}'^{CMG,DL}$ are semantically equivalent ($e =_{sem} e'$) if they are syntactically equivalent, the types of the corresponding model elements have identical semantics, and the associated domain statements share the same meaning. $e =_{sem} e'$ iff:

- $e =_{syn} e'$ (**S4**)
- $M(D_{\tau(e)}) = M(D_{\tau(e')})$ (**S5**)
- $\forall s \in S, \forall s' \in S : (e,s) \in A \wedge (e',s') \in A' \rightarrow M(D_s) = M(D_{s'})$ (**S6**)

3.2 Semantic Analysis Conflicts

There are a couple of conflicts that can arise when a semantic analysis of a conceptual model is performed (taxonomies of these conflicts can be found for example in [24-27]).

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 $CM_{IM}^{CMG}\langle E,F,S,A\rangle$ and a semantic pattern $CM_{IM}'^{CMG}\langle E',F',S',A'\rangle$ iff:

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

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

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

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 $CM_{IM}\langle E,F,S,A\rangle$ and a semantic pattern $CM'_{IM'}\langle E',F',S',A'\rangle$ iff:

$$\exists s \in S : s \in S' \wedge M(D_s) \neq M(D'_s) \quad \textbf{(C3)}$$

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 $CM_{IM}\langle E,F,S,A\rangle$ and a semantic pattern $CM'_{IM'}\langle E',F',S',A'\rangle$ iff:

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

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

3.3 Solving the Conflicts

The general approach of this paper is to eliminate the semantic analysis conflicts through the adoption of rules for modelling grammars which simplify the examination of conceptual models.

Type conflicts can 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 a model CM_{IM}^{CMG} and a semantic pattern CM'^{CMG}_{IM} 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 $\forall c, c' \in C, c \neq c' : M(D_c) \cap M(D_{c'}) = \varnothing$ the following conclusion can be derived: $\forall \dot{e}, \ddot{e} \in E, \dot{e} \neq \ddot{e} : M(D_{\tau(\dot{e})}) \cap M(D_{\tau(\ddot{e})}) = \varnothing$. Thereof one can follow 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 a model CM_{IM}^{CMG} and a semantic pattern CM'^{CMG}_{IM} 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 a model CM_{IM}^{CMG} and a semantic pattern CM'^{CMG}_{IM} 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 an identification of syntactical equivalence 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 model elements arise. This means that different modellers who share the same internal model will come to a syntactically identical result.

Proposition 4 (Syntactical Equivalence of Model Elements): R1- R7 and

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

imply that $e_{\varepsilon}^{CMG,DL} =_{syn} e'^{CMG,DL}_{\varepsilon}$.

Proof: $\forall c, c' \in C, c \neq c' : M(D_c) \cap M(D_{c'}) = \varnothing$ and R1 imply that: $\forall \varepsilon \in IE, \exists c \in C : \varepsilon \in D(M_c)$ and $\forall \varepsilon \in IE, \neg \exists c' \in C, c' \neq c : \varepsilon \in D(M_{c'})$. 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: $\forall \varepsilon \in IE, \exists s \in S : \{\varepsilon\} = D(M_s)$ and $\forall \varepsilon \in IE, \neg\exists s' \in S, s' \neq s : \{\varepsilon\} \in D(M_{s'})$. 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).

- 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: $s \in S$.
- If $M(D_c) = M(D_s)$ then because of R7 and R6 $(e,s) \notin A$ and $s \notin S$.
- $M(D_c) \supset M(D_s)$ 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 model elements $e_{\varepsilon}^{CMG,DL}$ and $e_{\varepsilon}'^{CMG,DL}$ 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 model elements, it follows that $c = c'$ and $s = s'$. Because of S1-S3 from $c = c'$ and $s = s'$ follows that: $e_{\varepsilon}^{CMG,DL} =_{syn} e_{\varepsilon}'^{CMG,DL}$. □

The idea of proposition 3 is now transferred to the constructs of a 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 model elements which were created with the same modelling language and an identical domain language. It is claimed that if these model elements are syntactically equivalent and neither the domain language nor the modelling language contain homonyms then the two model elements are also semantically equivalent.

Proposition 5 (Semantical Equivalence of Model Elements):

- $e^{CMG,DL} =_{syn} e'^{CMG,DL}$
- $\forall c \in C : M(D_c) = M(D_c')$
- $\forall s \in DL : M(D_s) = M(D_s')$

imply that $e^{CMG,DL} =_{sem} e'^{CMG,DL}$.

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 $e^{CMG,DL} =_{syn} e'^{CMG,DL}$ implies that $\tau(e) = \tau(e')$ (S1) and where applicable $s = s'$ (S2, S3). Consequently it holds that: $M(D_{\tau(e)}) = M(D_{\tau(e')})$ (S5) and $M(D_s) = M(D_{s'})$ (S6). Hence, it follows: $e^{CMG,DL} =_{sem} e'^{CMG,DL}$. □

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

- $\forall c,c' \in C, c \neq c' : M(D_c) \cap M(D_{c'}) = \varnothing$ (**D1**)
- $\forall s,s' \in DL, s \neq s' : M(D_s) \neq M(D_{s'})$ (**D2**)
- $\forall c \in C : M(D_c) = M(D_c')$ (**D3**)
- $\forall s \in DL : M(D_s) = M(D_s')$ (**D4**)

the identification of two syntactically equivalent model elements can be traced back to a syntactical pattern. Starting from two identical internal models semantically and syntactically equivalent model elements are explicated. It is not necessary to apply a semantic analysis operation on conceptual models as a syntactic analysis operation is sufficient to identify a specific pattern. This in turn implies that if D1-D4 hold then a semantic analysis of conceptual models can be completely automated. The process

modelling and analysis method PICTURE contains a language that has been built based upon these criteria and that can be used for automated knowledge retrieval [6].

4. Conclusions and Future Research

The perspective of the paper is not to take conceptual models as given when they are analysed. Rather, we have argued that if the modelling language complies with certain rules then a semantic analysis process can be noticeably simplified. We have also shown that these language characteristics prevent the emergence of type, synonym, homonym, and abstraction conflicts as well as enable an automated semantic analysis.

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 investigate on how some of the criteria can be relaxed without increasing the analysis efforts.

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 they are an important artifact on their own. 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 search helps to address this issue [19, 28]. It is due to further research to evaluate how the proposed language characteristics influence these semantic operations.

Acknowledgments. 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

[1] Green, P. F., Rosemann, M.: Integrated Process Modeling: An Ontological Evaluation. Information Systems 25 (2000) 73-87
[2] Shanks, G., Tansley, E., Weber, R.: Using ontology to validate conceptual models. Communications of the ACM 46 (2003) 85-89
[3] Becker, J., Kugeler, M., Rosemann, M.: Process Management - A Guide for the Design of Business Processes. Springer, Berlin et al. (2003)
[4] Schwartz, D. G., Divitini, M., Brasethvik, T.: Internet-Based Organizational Memory and Knowledge Management. Idea Group, Hershey (2000)
[5] Witten, I. H., Frank, E.: Data Mining - Practical Machine Learning Tools and Techniques. 2nd edn. Elsevier, San Francisco (2005)
[6] Pfeiffer, D.: Constructing comparable conceptual models with domain specific languages. In: Proc. 15th European Conference on Information Systems (ECIS2007) (2007)
[7] Evermann, J., Wand, Y.: Towards Ontologically Based Semantics for UML Constructs. In: Proc. 20th International Conference on Conceptual Modeling (ER 2001) (2001) 354-367

[8] Schütte, R., Rotthowe, T.: The Guidelines of Modeling - An Approach to Enhance the Quality in Information Models. In: Proc. 17th International Conference on Conceptual Modeling (ER 1998) (1998) 240-254
[9] Patig, S.: Measuring Expressiveness in Conceptual Modeling. In: Proc. 16th International Conference on Advanced Information Systems Engineering (CAiSE 2004) (2004)
[10] Kamlah, W., Lorenzen, P.: Logical Propaedeutic. Pre-School of Reasonable Discourse. University Press of America, Lanham, MD (1984)
[11] Rosemann, M., van der Aalst, W. M. P.: A configurable reference modelling language. Information Systems 32 (2007) 1-23
[12] Desel, J., Reisig, W.: Place/Transition Petri Nets. In: W. Reisig and G. Rozenberg, (eds.): Lectures on Petri Nets I: Basic Models. Springer (1998) 122-173
[13] Pfeiffer, D., Niehaves, B.: Evaluation of Conceptual Models - A Structuralist Approach. In: Proc. 13th European Conference on Information Systems (ECIS 2005) (2005)
[14] Balzer, W., Moulines, C. U., Sneed, J. D.: An Architectonic for Science - The Structuralist Program. D. Reidel Publishing Company, Dordrecht et al. (1987)
[15] 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: Proc. 17th ACM Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2002) (2002)
[16] Object Management Group. UML 2.0 Superstructure Specification.[Online]. Available: http://www.omg.org/cgi-bin/doc?formal/05-07-04
[17] Chen, P. P.-S.: The Entity Relationship Model - Toward a Unified View of Data. ACM Transaction on Database Systems 1 (1976) 9-36
[18] Pfeiffer, D., Gehlert, A.: A framework for comparing conceptual models. In: Proc. Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2005) (2005) 108-122
[19] Rahm, E., Bernstein, P. A.: A survey of approaches to automatic schema matching. The VLDB Journal 10 (2001) 334-350
[20] van Glabbeek, R. J., Weijland, W. P.: Branching Time and Abstraction in Bisimulation Semantics. Journal of the ACM 43 (1996) 555-600
[21] van der Aalst, W. M. P., Alves de Medeiros, A. K., Weijters, A. J. M. M.: Process Equivalence: Comparing Two Process Models Based on Observed Behavior. In: Proc. 4th International Conference on Business Process Management (BPM2006) (2006) 129-144
[22] Pomello, L., Rozenberg, G., Simone, C.: A Survey of Equivalence Notions for Net Based Systems. In: G. Rozenberg, (ed.) Lecture Notes in Computer Science, Vol. 609; Advances in Petri Nets 1992. Springer (1992) 410-472
[23] Milner, R.: A Calculus of Communicating Systems. Springer, Berlin (1980)
[24] Hakimpour, F., Geppert, A.: Resolving Semantic Heterogeneity in Schema Integration: an Ontology Based Approach. In: Proc. International Conference on Formal Ontologies in Information Systems FOIS'01 (2001) 297-308
[25] Kashyap, V., Sheth, A.: Semantic and schematic similarities between database objects: a context-based approach. The International Journal on Very Large Data Bases 5 (1996) 276-304
[26] Lawrence, R., Barker, K.: Integrating relational database schemas using a standardized dictionary. In: Proc. 16th ACM Symposium on Applied Computing (2001)
[27] Davis, I., Green, P., Milton, S., Rosemann, M.: Using Meta Models for the Comparison of Ontologies. In: Proc. Eighth CAiSE/IF IP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (2003)
[28] Batini, C., Lenzerini, M., Navathe, S. B.: A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys 18 (1986) 323-364