

A Conceptual Modeling Technique without Redundant Structural Elements

V.V. Ovchinnikov (ovch@lipetsk.ru)

Ovchinnikov V.V., PhD in Engineering Science, is the head of main production systems development department of NISCo (Russia, Lipetsk), and assistant professor of Lipetsk State Technical University. The author is available at e-mail: ovch@lipetsk.ru, phone: +70742411139.

Conceptual modeling performs the key role during realization of information system development projects. At present, there is a quantify of conceptual modeling techniques, for instance, object-role modeling technique which essence is shortly stated in the work.

Existent conceptual modeling techniques have some disadvantages: ample quantity of equivalent representations for a single universe of discourse (UoD); impossibility of conceptual query formulating exclusively by means of UoD terms; necessity of additional knowledge of relation using conditions and, as a result, insufficiency of knowledge of interrelations between UoD terms for query formulating; some difficulty of model familiarization for one who is not a specialist in the field of information system designing. Enumerated disadvantages negatively influence on efficiency and quality of designer's work, on speed and quality of communication among project participants and, as a consequence, on quality of information system being developed.

In the present paper, we introduce a conceptual modeling technique, named semantically complete one, that is devoid of the enumerated features; we produce analysis of its advantages and disadvantages, and an example of its use in practice.

Keywords: conceptual model, object-role model, semantically complete model

1. Introduction

For conceptual modeling techniques, the conceptualization principle [12, 6] should be held true. According to the principle, a conceptual model should solely contain a description of an Universe of Discourse (UoD), and should not affect aspects of effective data storing and access. This principle allows to achieve the following advantages of conceptual model use at the conceptual designing stage beside use of a data form model at this stage. The first advantage is simplicity and controllability of it. Adding of data form aspects to a model leads to complexity increase without any changes of information saturation in the view of the ultimate task. Therefore, during the first designing stage, it is preferable to elaborate a conceptual model. It enables to comprehend the essence of UoD and solving task without deciding with regard to implementation ways of an application system. In this stage, designer's ideas of UoD and solving task undergo appreciable revising, and only after its stabilization, it is reasonable to go to the next stage that is to analyze implementation ways of an application system, since the second stage entirely depends on the decision attained in the first stage.

The second advantage lies in more expandability of a conceptual model. During model expanding, absence of implementation details lets to manage a few of new concepts, which quantity is precisely equal to quantity of added terms of UoD. While a conceptual model is expanding, a corresponding data form model may be in need of a complete revision of its organization principles. Therefore, at this stage of conceptual modeling, it is essential to capture the solving task as completely as possible, to take into consideration maximum heterogeneous aspects. Otherwise, a designer is doomed to repeat the process of making of a data form model on basis of a conceptual model over and over again.

Absence of implementation details leads to that any UoD has less quantity of equivalent representations as conceptual models than as data form models. As a result, when the set problem solution is looked up, a designer has to make far less number of equivalent transformations of

conceptual model than of data form model. It results in acceleration and quality growing of the designing process.

A model structure has a profound effect on formulating of constraints and queries. Absence of necessity to operate with data storing aspects leads to the following advantage of conceptual models – the relative simplicity of constraints and queries formulating. In spite of the fact that conceptual constraints and queries can not frequently be used during an application system implementation directly, they perform the decisive role in the designing process since they have the following property. For conceptual constraints and queries, as for conceptual models, there exists far less number of equivalent representations. It lets to accomplish designing of constraints and queries with better rapidity and quality at the conceptual level than at the implementation level. Switching to analysis of implementation ways is expedient to realize after completion of the conceptual designing phase since moderate changes at a conceptual model may lead to necessity of complete revising of early accepted decisions on constraints and queries implementation.

The further development of conceptual modeling techniques is expedient to accomplish in the direction of evolution of advantages allowed by the conceptualization principle: relative model simplicity, expandability, relative simplicity of constraints and queries formulating.

Expressive power of a modeling technique, conveyed by diversity of constructs used during modeling, is generally accepted as advantage relative to low-powered modeling techniques. But defining of several alternative mechanisms for solving a single task inevitably causes the negative consequence stating that a UoD gets a set of additional equivalent representations. The space of equivalent representations of UoDs enlarges, a designer's degree of freedom increases. This situation is acceptable for data form models, but it is shortage of conceptual models. Alternative mechanisms lead to model complexity growing, while model information saturation remains invariable from the angle of UoD structure. Number of UoD equivalent representations that should be analyzed by a designer is increased. As a result, the designing productivity is decreased, and the complexity of constraints and queries formulating is augmented.

Another criterion of conceptual modeling technique quality is possibility of constraints and queries formulating only by means of UoD terms without using auxiliary concepts that are not reflected to UoD terms directly (in particular, without using relations). To date, there is no a conceptual modeling technique that complies with this criterion in full measure since any technique requires at least to use model relations for constraints and queries making. Attaining of the property in the framework of a conceptual modeling technique lets to simplify formulating of constraints and queries, and lets to increase the model expandability. It is undoubtedly useful on the way of conceptual modeling methodology developing.

Effectiveness of communication among project participants depends on complexity of conceptual model study process. Existing techniques require the following study stages:

- familiarization of UoD terms reflected in a model;
- familiarization of model relations that represent correlations between UoD terms;
- familiarization of using conditions for the relations.

At that, knowledge about relation using conditions is not included into the model and is a part of model environment. Transmitting of the knowledge can not be realized through the model itself. One can transmit it only by means of communication or additional verbal descriptions. Study of a conceptual model may become simpler if its relations will be defined unconditionally, that is all relation using conditions will be set as UoD terms. Below we investigate properties of the unconditionally defined relations and requirements for the conceptual modeling techniques containing exclusively unconditionally defined relations.

Object-role modeling technique (ORM) [9, 2] is one of the most advanced conceptual modeling techniques. It is thoroughly investigated and has several modifications. NIAM [14] is the earliest ORM variation that allows only binary fact types (relation types). FORM [7] and PSM [11] extend NIAM by means of additional constructions based on special object types: set, sequence, polymorphism. PM [1]

is NIAM extension in the direction of n-arity of fact types. NORM [16] is an object-oriented extension of NIAM. In [15, 17], the restriction of ORM facts to binary ones is investigated.

Ample quantity of different ORM modifications and their formalizations led to necessity of creating of a general formal core of the object-role modeling techniques from which all known modifications are derived as special cases. This work was done in [3].

Subsequent investigations in the field of object-role modeling concerned with conceptual languages on basis of ORM [12, 13], transformation and optimization of conceptual model [8], allocation of abstraction layers on flat conceptual schemas [4, 5], creating of ORM meta-model by means of ORM itself [10], and other topics. All these investigations do not essentially affect the core of object-role modeling.

According to [3], any object-role model is based on the following concepts. The object type is base one and is directly reflected to UoD terms. It is generally divided into two categories: abstract and concrete object types. But in the context of conceptual modeling, this division is premature and should not be done on the conceptual designing stage.

Connections among ORM object types are described with the help of fact types. Each fact type is a relation based on object types. Including of an object type to a fact type is distinguished as an individual concept – role. By using this concept, one formulates constraints of the object-role models.

The described base concepts are enough to formalize any UoD by way of an object-role model. Apart from these concepts, ORM has large quantity of subsidiary concepts that serve for increasing of expressiveness and using convenience. The power type (set type) lets to declare an object type that is a set consisting of objects of another object type. Similarly, the sequence type lets to declare an object type, that is a sequence of objects of another object type. The power type and the sequence type do not differ from the other object types in their use.

The inheritance type is the second type of relations being applied in ORM. It is a directed binary relation that sets a biunique connection between two object types: a supertype and a subtype. The inheritance type has two variants. The specialization type is such an inheritance type for which it is true that each subtype object biuniquely corresponds to a supertype object, and, at the same time, not each supertype object corresponds to a subtype object. The specialization type lets to declare object types that inherit all properties of another object type, the properties being expressed by fact types containing this object type, and probably have additional properties.

The generalization type is an inheritance type that satisfies the following condition. Consider a supertype and all its subtypes that are connected with this supertype by means of generalization types. For each object of the supertype, it is true that this object biuniquely corresponds to an object of some subtype, and does not correspond to more than one objects of the supertype's subtypes, in spite of the fact that the supertype relates to different subtypes. The generalization type lets to declare object types uniting all properties of several other object types and probably having additional properties.

The concept of objectified fact type lets to use a fact type in a model as an object type and represents such an object type that has objects biuniquely determining instances of this fact type. The concept of objectified object-role model lets to use an object-role submodel as an object type. Each object of objectified object-role model biuniquely determines a state of the submodel. Objectified fact type and objectified object-role model may be used just as other object types are used.

Object-role modeling is characterized by the following properties that partially complicate using ORM as a conceptual modeling technique. Firstly, ORM contains redundant structural elements. For each UoD, one may create an object-role model that do not have the following elements: power types, sequence types, objectified fact types, objectified object-role models, specialization types, generalization types. Existing of the redundant structural elements leads to the twofold effect: on the one hand the possibility of more laconic formulating of interrelations between object types appears, and on the other hand a quantity of equivalent representations of one UoD increases and necessity of special way of queries and constraints formulating arises.

Secondly, formulating of queries for an object-role model requires to point explicitly at fact types that are in use. Since fact types do not directly reflect to UoD terms, an elaboration of a query language for object-role model that may let to formulate queries exclusively by means of UoD terms is not possible in the general case.

Thirdly, fact types may set interrelations between object types for some conditions that are not included to a certain object-role model since there may exist several fact types based on the same set of object types. Conditions of using of such fact types are external knowledge with respect to this model.

Thus, the object-role modeling technique is the advanced modeling tool, but it has a quantity of properties that complicate using of it as a pure conceptual modeling tool.

Let us illustrate our assertion that any UoD may be represented as an object-role model without using of its redundant elements. For each redundant element, show how it is reduced to a representation in the form of the base elements: object types and fact types.

A power type can be replaced by a declaration of an object type and a fact type without any information loss in the following way. Initially, we have an object type whose objects need to form into sets. Name this object type as the element type. Declare another object type that we name as the explicit power type. Create a binary fact type based on the explicit power type and the element type. Declare two constraints on this fact type:

- each object of the explicit power type correspond to its own set of objects of the element type, in other worlds, there are no two objects of the explicit power type that correspond to the same set of objects of the element type;
- there exists only one object of the explicit power type that does not correspond to any object of the element type.

The declared explicit power type is characterized by all properties that characterize a power type implicitly declared by means of special construction of object-role modeling technique. Therefore, the implicit declaration of a power type may be replaced by the explicit declaration without using of this redundant construction element of ORM.

A sequence type can be replace by a declaration of two object types and one fact type without any information loss in the following way. Initially, there exists an object type whose objects need to form into sequences. Name the object type as the sequence element type. Declare two object types named as the explicit sequence type and the sequence position. Create a ternary fact type based on the sequence element type, the explicit sequence type, and the sequence position. Declare the following constrains of the fact type:

- the sequence position is the positive integer;
- each combination of an explicit sequence type object and a sequence position value corresponds to a single object of the sequence element type;
- each object of the explicit sequence type corresponds to such a set of sequence position values that change serially from one by one in the framework of this set;
- there are no two objects of the explicit sequence type that correspond to the same set of combinations of a sequence element type object and a sequence position value;
- there exists only one object of the explicit sequence type that does not correspond to any combination of a sequence element type object and a sequence position value.

This properties of the explicitly declared sequence type coincide with the properties of any implicit sequence type declared by means of special construction of object-role modeling. Therefore, each implicit declaration of sequence type may be replaced by an explicit declaration at any moment without using this redundant construction element of object-role model.

Any inheritance type may be replaced by explicit using of a fact type in the following way. Initially, we have an object type being inherited, that we name as supertype. Declare an object type that should inherit from the supertype, and that we name as subtype. Create a binary fact type based on the supertype and the subtype. If the inheritance type is a specialization type, declare the following

constraint: each subtype object biuniquely corresponds to a supertype object. If the inheritance type is a generalization type, declare the other constraint: each supertype object biuniquely corresponds to an object of a single subtype being generalized by this supertype. The explicitly declared subtype has all properties of implicit one. Any implicit declaration of inheritance type may therefore be replaced by an explicit declaration without using this redundant element of ORM.

Any objectified object type may be represented as a simple fact type in the following way. Initially, we have a fact type that needs to be used as an object type. Instead of declaring the fact type as objectified, we declare an explicit object type and add it into the fact type. Then apply the following constraints:

- each object of the explicit object type biuniquely corresponds to a combination of objects of other object types included into the fact type;
- each combination of objects of object types initially constituted the fact type biuniquely determines an object of the explicit object type.

This explicit object type has all properties that characterize an objectified object type. Any implicit declaration of objectified object type may therefore be replaced by an explicit declaration without using of this redundant element of ORM.

Any objectified object-role model may be replaced by a declaration of an explicit object type in the following way. Initially, we have an object-role submodel that should be used as an object type. Declare an explicit object type that we name as model type. Each object of the model type represents one state of the submodel. It is described as follows.

Add the model type to all fact types of the submodel. From this point onwards each fact type of the submodel consists of the model type and initial object types that was a part of the fact type before adding the model type. In the framework of each fact type of the submodel, each object of the model type corresponds to a set of combinations of objects of the initial object types. This set is a state of the initial fact type in the framework of a submodel state designated by this object of the model type. States of all initial fact types, that correspond to one object of the model type, form the submodel state.

Apply the constraint that there are no two objects of the model type that correspond to the same states of all initial fact types. The model type declared so has all properties that characterize an objectified object-role model based on the same submodel. Any implicit declaration of objectified object-role model may therefore be replaced by an explicit declaration without using this redundant element.

Therefore, power types, sequence types, specialization types, generalization types, objectified fact types, and objectified object-role models are redundant construction elements of ORM, that are not obligatory for formalizing any UoD. Appearance of these constructions in ORM is explained by necessity to increase convenience of prevalent concepts formulating.

In this work, we pose a problem of developing basics of an alternative conceptual modeling technique that characterizes by the following peculiar properties:

- A model has the minimal quantity of equivalent representations in the framework of a single UoD. It is attained by minimizing of quantity of alternative structural elements.
- Queries are formulated exclusively by means of UoD terms. It lets to simplify query formulating, to decrease quantity of equivalent ways of query creating, to approach a query formalization to an appropriate natural language sentence.
- Relations are unconditionally defined upon UoD terms. One does not require knowledge not reflected in the models. It lets to hasten familiarization with such conceptual model, to increase manageability and expandability of it.

We name the models that are agreeable to these properties as semantically complete ones because of absoluteness of relations being included in it. It lets us to say that a designer and a user of such model are only in need of knowledge of UoD terms and interrelations of the terms, and knowledge of relation using conditions is not required for them.

The structure of this article is as follows. In the next section, a formal definition of semantically complete model is given in the form of set-theoretic model with constraints set by way of propositional logic expressions. In section 3, properties of semantically complete model, following from its formalization, are listed, advantages and disadvantages of semantically complete model are analyzed. In section 4, application of semantically complete modeling for formalizing semantically complete model itself is showed, the initial set-theoretic formalization is compared with the formalization by way of semantically complete model. In the last section, conclusions about practical applicability of the semantically complete modeling as a conceptual modeling technique are given, directions of further work are provided.

2. Formal Definition of Semantically Complete Model

In the framework of semantically complete modeling technique, a concept of UoD term is basic and therefore has no its own definition. Terms are formed by men during interaction with the surrounding reality. Each term is considered to have a finite or infinite set of possible senses, one sense corresponds to only one term.

Let $Term$ is the set of UoD terms;

$Sense$ is the set of all senses of all terms.

Then, there exists the following map of the term sense set to the UoD term set:

$ST : Sense \rightarrow Term$. For this map, the following constraint is applied: each sense corresponds to exactly one term and each term corresponds to at least one sense:

[SCM1]: $\forall s \in Sense \exists_1 t \in Term \{(s,t) \in ST\} \wedge \forall t \in Term \exists s \in Sense \{(s,t) \in ST\}$, where \exists_1 represents a quantifier “there exists one and only”.

In the framework of an information system, term senses are coded by some values that are represented as numbers, strings or dates. A term sense uniquely determines a value, but a value may code different senses of different terms. At that, aiming at information integrity, we consider that there are no two values coding the same term sense. It follows that we may assert that combination of a term and its value uniquely determines a sense of this term.

Let $Value$ is the set of all values being able to code senses in principle. Then, there exists the following map of senses to values: $SV : Sense \rightarrow Value$. For this map, it is true that each sense corresponds to a value:

[SCM2]: $\forall s \in Sense \exists_1 v \in Value \{(s,v) \in SV\}$

For ST and SV , it is held true that each combination of a term and a value corresponds to a single sense or none of senses:

[SCM3]: $\forall t \in Term \forall v \in Value \{ \exists_1 s \in Sense \{(s,t) \in ST \wedge (s,v) \in SV\} \vee \overline{\exists s \in Sense \{(s,t) \in ST \wedge (s,v) \in SV\}} \}$

It follows from [SCM1] and [SCM2] that each sense biuniquely corresponds to one combination of a term and a value:

[SCM4]: $\forall s \in Sense \exists_1 t \in Term \exists_1 v \in Value \{(s,t) \in ST \wedge (s,v) \in SV\}$

Any semantically complete relation represents a relation based on UoD terms and having the following key properties:

- one term may participate in some relation only once;
- there are no two semantically complete relations based on the same set of terms;
- there are no two semantically complete relations participating in the same semantically complete model that are based on term sets being included one in another.

These constraints is decisive and their effects are mentioned below where properties of semantically complete model are discussed.

For formulating of constraints on semantically complete relations, the concept of relation position is introduced. It represents a combination of a term and a relation and tells about participating

of this term in this relation. A relation position uniquely determines a relation and a term that participates in the relation. A term is considered to be included into a relation if there is a relation position related to the term and the relation.

Let RP is the set of all relation positions;
 SCR is the set of all semantically complete relations.

Then $RP_{Term} : RP \rightarrow Term$ is the map of relation positions to UoD terms; $RP_{SCR} : RP \rightarrow SCR$ is the map of relation positions to semantically complete relations. For these maps, the following constraints are held true.

Each relation position corresponds to one appropriate term and one semantically complete relation:

$$[SCM5]: \forall rp \in RP \{ \exists t \in Term \{ (rp, t) \in RP_{Term} \} \wedge \exists scr \in SCR \{ (rp, scr) \in RP_{SCR} \} \}$$

Each semantically complete relation contains at least one relation position:

$$[SCM6]: \forall scr \in SCR \exists rp \in RP \{ (rp, scr) \in RP_{SCR} \}$$

There are no two different relation positions that are included into the same semantically complete relation and are related to the same term:

$$[SCM7]: \forall rp_1 \in RP \overline{\exists rp_2 \in RP \exists scr \in SCR \exists t \in Term \left\{ \begin{array}{l} rp_1 \neq rp_2 \wedge (rp_1, scr) \in RP_{SCR} \wedge (rp_2, scr) \in RP_{SCR} \wedge \\ (rp_1, t) \in RP_{Term} \wedge (rp_2, t) \in RP_{Term} \end{array} \right\}}$$

To formalize the next constraint, we define a function calculating a set of terms that the given semantically complete relation is based on:

$$SCR_{TermSet}(scr \in SCR) \equiv \{ t \mid \exists rp \{ (rp, t) \in RP_{Term} \wedge (rp, scr) \in RP_{SCR} \}$$

There is no two different semantically complete relations based on the same set of terms:

$$[SCM8]: \forall scr_1 \in SCR \overline{\exists scr_2 \in SCR \{ scr_1 \neq scr_2 \wedge SCR_{TermSet}(scr_1) = SCR_{TermSet}(scr_2) \}}$$

Define a semantically complete model as a set of semantically complete relations. Aim at formulating queries for any semantically complete model exclusively by means of UoD terms (we explain it later in detail). The effect of this aiming is inanity of using of terms that are not included to any semantically complete relation of a certain model in queries since it is impossible to turn from such terms to any other term. Therefore, it is considered that terms are not independently included to semantically complete models, a model contains only those terms that are included to at least one its relation.

Let SCM is the set of semantically complete models, then there is the relation $SCM_{SCR} \subseteq SCM \times SCR$ indicating which semantically complete relations are included to a certain model.

In the framework of any semantically complete model, there are no two different semantically complete relations based on sets that are included one to another:

$$[SCM9]: \forall scm \in SCM \overline{\forall scr_1 \in SCR \exists scr_2 \in SCR \{ (scm, scr_1) \in SCM_{SCR} \wedge (scm, scr_2) \in SCM_{SCR} \wedge scr_1 \neq scr_2 \wedge SCR_{TermSet}(scr_1) \subset SCR_{TermSet}(scr_2) \}}$$

Name an instance (tuple) of semantically complete relation as a semantically complete cohesion. In the general case, a relation instance represents a set of couples of a sense and a relation position since one sense may be met at different positions of the same instance. At that, it follows from [SCM1] and [SCM7] that one sense may be met in semantically complete cohesion only once. Any semantically complete cohesion may therefore be represented as a sense set.

Let SCC is the set of semantically complete cohesions, then there exists the relation $SCC_{Sense} \subseteq SCC \times Sense$ that shows senses forming a certain semantically complete cohesion.

Each semantically complete cohesion consists of at least one sense:

$$[SCM10]: \forall scs \in SCC \exists s \in Sense \{ (scs, s) \in SCC_{Sense} \}$$

There are no two different semantically complete cohesions consisting of the same set of senses:

$$[\text{SCM11}]: \forall scc_1 \in SCC \overline{\exists scc_2 \in SCC} \{scc_1 \neq scc_2 \wedge \{s \mid (scc_1, s) \in SCCSense\} = \{s \mid (scc_2, s) \in SCCSense\}\}$$

Any semantically complete relation may contain different sets of cohesions in different moments. Name the set of cohesions constituting a certain relation in a certain moment as the semantically complete relation state. Each state is biuniquely determined by a cohesion set, that is two different relation states can not contain the same set of cohesions. Since the concept of semantically complete cohesion is an extension of the concept of relation instance (tuple), any cohesion relates to states of only one relation and can not relate to states of different relations.

Let $SCRS$ is the set of all semantically complete relation states, then there is the map $SCRSSCR : SCRS \rightarrow SCR$ showing relations that a certain relation state is related to, and there is the relation $SCRSSCC \subseteq SCRS \times SCC$ showing cohesions that constitute a certain relation state.

Each semantically complete relation state pertains to a certain semantically complete relation, and each semantically complete relation has at least one possible state:

$$[\text{SCM12}]: \forall scrs \in SCRS \exists_1 scr \in SCR \{(scrs, scr) \in SCRSSCR\} \wedge \forall scr \in SCR \exists scrs \in SCRS \{(scrs, scr) \in SCRSSCR\}$$

Each semantically complete cohesion relates to a certain semantically complete relation state:

$$[\text{SCM13}]: \forall scc \in SCC \exists_1 scrs \in SCRS \{(scrs, scc) \in SCRSSCC\}$$

There are no two different semantically complete relation states consisting of the same set of semantically complete cohesions:

$$[\text{SCM14}]: \forall scrs_1 \in SCRS \overline{\exists scrs_2 \in SCRS} \left\{ \overline{\{scrs_1 \neq scrs_2 \wedge \{scc \mid (scrs_1, scc) \in SCRSSCC\} = \{scc \mid (scrs_2, scc) \in SCRSSCC\}} \right\}$$

Each semantically complete relation can have only one state not containing any semantically complete cohesion:

$$[\text{SCM15}]: \forall scr \in SCR \left\{ \overline{\exists_1 scrs \in SCRS \{(scrs, scr) \in SCRSSCR\} \wedge \forall scc \in SCC \{(scrs, scc) \notin SCRSSCC\}} \vee \overline{\exists scrs \in SCRS \{(scrs, scr) \in SCRSSCR\} \wedge \forall scc \in SCC \{(scrs, scc) \notin SCRSSCC\}} \right\}$$

Each semantically complete cohesion relates to states of the same semantically complete relation:

$$[\text{SCM16}]: \forall scc \in SCC \exists_1 scr \in SCR \exists scrs \in SCRS \{(scrs, scc) \in SCRSSCC \wedge (scrs, scr) \in SCRSSCR\}$$

It follows from the relation properties that each semantically complete cohesion consists of senses related to terms that form a relation whose state consists of this cohesion:

$$[\text{SCM17}]: \left\{ \overline{\forall scc \in SCC \forall s \in Sense \forall scrs \in SCRS \forall scr \in SCR \forall rp \in RP \forall t \in Term} \right. \\ \left. \{(scc, s) \in SCCSense \wedge (scrs, scc) \in SCRSSCC \wedge (scrs, scr) \in SCRSSCR \wedge \right. \\ \left. \{(rp, scr) \in RPSCR \wedge (rp, t) \in RPTerm \rightarrow (s, t) \in ST\} \right\}$$

Any semantically complete model may have different states at different moments. A semantically complete model state is interpreted as a set of semantically complete relation states. Each relation of a certain model is represented by one state of this relation in the framework of each state of this model.

Let $SCMS$ is the set of all semantically complete model states, then there is the map $SCMSSCM : SCMS \rightarrow SCM$ presenting the correspondence between semantically complete models and their states, and there is the relation $SCMSSCRS \subseteq SCMS \times SCRS$ presenting the entrance of semantically complete relation states into semantically complete model states.

Each semantically complete model state relates to one semantically complete model, and each semantically complete model has at least one possible state:

$$[\text{SCM18}]: \forall scms \in SCMS \exists_1 scm \in SCM \{(scms, scm) \in SCMSSCM\} \wedge \forall scm \in SCM \exists scms \in SCMS \{(scms, scm) \in SCMSSCM\}$$

Each semantically complete model state contains at least one semantically complete relation state:

[SCM19]: $\forall scms \in SCMS \exists scrs \in SCRS \{ (scms, scrs) \in SCMSSCRS \}$

For each relation participating in a certain model, there is only one state of this relation in the framework of each state of this model:

$\forall scr \in SCR \forall scm \in SCM \forall scms \in SCMS$

[SCM20]: $\left\{ \begin{array}{l} (scm, scr) \in SCMSCR \wedge (scms, scm) \in SCMSSCM \rightarrow \\ \exists_1 scrs \in SCRS \{ (scms, scrs) \in SCMSSCRS \wedge (scrs, scr) \in SCRSSCR \} \end{array} \right\}$

For each state of a certain relation that participates in a state of a certain model, this relation participates in this model:

$\forall scrs \in SCRS \forall scm \in SCM \forall scms \in SCMS$

[SCM21]: $\left\{ \begin{array}{l} (scms, scrs) \in SCMSSCRS \wedge (scms, scm) \in SCMSSCM \rightarrow \\ \exists_1 scr \in SCR \{ (scm, scr) \in SCMSCR \wedge (scrs, scr) \in SCRSSCR \} \end{array} \right\}$

We think it is necessary to note that the given formalization of semantically complete model is invariant to ways of setting relation states, cohesions, model states, and other its elements. This formalization represents the conceptual description devoid of realization details. Effective methods of realizing semantically complete model in the framework of any system require a separate investigation.

3. Properties of Semantically Complete Model

All the maps and relations of the semantically complete model formalization are intentionally declared such that they satisfy all properties of semantically complete relations. Therefore, for illustration of semantically complete model properties, we will use the above formalization.

It directly follows from [SCM7] and [SCM8] that any semantically complete relation is uniquely identified by a term set, that forms the relation. In other words, being acquainted of a term set, one can determine the relation that is based on this term set without using the relation's proper name.

This property of semantically complete relations has the following line of practical consequences. Firstly, semantically complete relations do not need to assign a proper name to them since they can be used through appropriate term sets. To declare, for example, the relation $SCMSSCRS \subseteq SCMS \times SCRS$ as semantically complete one, it is enough to declare its existence as $SCMS \times SCRS$ without necessity to assign a proper name. For the map $SCMSSCM : SCMS \rightarrow SCM$, its declaration may be the following: $SCMS \rightarrow SCM$. Taking into account that semantically complete relations may be used by means of term sets, the expression [SCM21] may be reformulated as follows:

$\forall scrs \in SCRS \forall scm \in SCM \forall scms \in SCMS$

$\left\{ \begin{array}{l} (scms, scrs) \in (SCMS, SCRS) \wedge (scms, scm) \in (SCMS, SCM) \rightarrow \\ \exists_1 scr \in SCR \{ (scm, scr) \in (SCM, SCR) \wedge (scrs, scr) \in (SCRS, SCR) \} \end{array} \right\}$

Secondly, operations on semantically complete relations may be formulated without using of explicit references to relations since, knowing terms that variables are related to, according to the property [SCM8], we can determine the used relations. For example, if we set that the default operation is the membership \in , the expression [SCM21] may be written as the following one:

$\forall scrs \in SCRS \forall scm \in SCM \forall scms \in SCMS \{ (scms, scrs) \wedge (scms, scm) \rightarrow \exists_1 scr \in SCR \{ (scm, scr) \wedge (scrs, scr) \} \}$

Here we replace $(scms, scrs) \in SCMSSCRS$ on $(scms, scrs)$ by the following reasons. Being aware that $scms$ is the variable on senses of $SCMS$, and $scrs$ is the variable on senses of $SCRS$, we can assert that $(scms, scrs)$ is the check of belonging a certain cohesion to the relation $(SCMS, SCMR)$ since this relation is based on $SCMS$ и $SCRS$, and the default operation is the membership \in . As we can see, the declaration and using of semantically complete relations may be simpler than not semantically complete ones. Pay particular attention to the fact that proper names of relations are not used in the given expression, and the expression formulation is realized exclusively by means of UoD terms. Since

any query can be formed as an expression, we conclude that all queries to semantically complete models may be formulated exclusively by means of UoD terms.

Advancing on the way of expression simplification, let that variables based on senses of a certain term are named the same way as this term with possibility to use indexes if several variables for one term in the framework of one expression are required. In this case, the expression [SCM21] has the following view:

$$\forall scrs \forall scm \forall scms \{ (scms, scrs) \wedge (scms, scm) \rightarrow \exists_1 scr \{ (scm, scr) \wedge (scrs, scr) \} \}$$

Let that if an expression is a predicate, that is non-apparent variables are not explicitly declared for it, then the following rule is applicable. If quantifiers for some expression variables are omitted, consider that a sequence of universal quantifiers for such variables is implied in the very beginning of this expression.

Consider variables being bounded by universal quantifiers in the beginning of an expression. All possible sense combinations of all the bounded variables form Cartesian product of sets of possible senses of these variables. Since the Cartesian product is the same for all possible sequences of bounded variables, we conclude that the chosen sequence of universal quantifiers in the expression beginning may be arbitrary, for example, in the order of variable occurrence in the expression. By applying the rule to the expression [SCM21], we obtain the following one:

$$\{ (scms, scrs) \wedge (scms, scm) \rightarrow \exists_1 scr \{ (scm, scr) \wedge (scrs, scr) \} \}.$$

Introduce the concept of variable chain. Under a variable chain, a sequence of variables separated by the symbol “–” is implied. Any variable chain is interpreted as a concatenation of all variable couples having the symbol “–” between them. For example, $(scms, scrs) \wedge (scms, scm)$ is represented as $(scms - scrs - scm)$. For chains, the parenthesis may be omitted as no ambiguity is begotten. This step of simplification of semantically complete model expression is applicable to binary relations only. Taking into consideration that binary relations are the very often used ones, the given simplification has the profound practical effect. By applying this chain construction rule to the expression [SCM21], we obtain the following one:

$$\{ (scrs - scms - scm) \rightarrow \exists_1 scr \{ (scrs - scr - scm) \} \}$$

At the present time, formalization is generally realized by means of abbreviations designating UoD terms. For example, for designation of the term “semantically complete model state”, we used the abbreviation *SCMS*. At this case, designer has to work in two planes: the plane of UoD terms and the plane of abbreviations of these terms. Necessity of continual switching between these planes makes additional difficulties for designers, hinders concentrating on UoD essence during conceptual modeling. Therefore, for designating UoD terms, we suggest to use best understood phrases rather than ciphered abbreviations. During formalization of expressions of high complexity, this approach certainly begets the other problem lying in that some expression formalization becomes bulky. In this situation, there is no an unambiguous solution what is the best for designer – formulating an expression in the intuitively obvious form or in the more compact form. From our point of view, for designer and model users, using of sensible term names greatly simplifies returning to model analysis after some interruption. By using of the suggested approach, the expression [SCM21] may be reformulated as follows:

$$(SC \text{ Relation State} - SC \text{ Model State} - SC \text{ Model}) \rightarrow \exists_1 SC \text{ Relation} \{ SC \text{ Relation State} - SC \text{ Relation} - SC \text{ Model} \}.$$

The direct analogy is observed between this approach to formulating expression and its verbal disclosure: “for each semantically complete relation states included in a state of a semantically complete model it is true that there is only one semantically complete relation to which the given semantically complete relation state is related and which is included into the given semantically complete model”. So considerable closeness of the text to the formalization, first of all, is the consequence of possibility to interact with the semantically complete models exclusively by means of UoD terms.

Clarity of a certain conceptual model for man who is not a specialist in the field of information system designing is the model advantage. Such property of the model tells about its closeness to UoD essence, and facilitates communication among project participants. The given property may be achieved by representing the conceptual model in the form that is close to natural language sentences.

Since all the semantically complete relations are based on unique term sets ([SCM7], [SCM8]), and any semantically complete model does not require to assign a proper name to the relations, all the semantically complete relations may be declared in the form of the UoD facts formulated on a natural language and represented as descriptions of interrelation between selected UoD terms. For example, instead of declaring the following relation as “SC Model State × SC Relation State”, the declaration of this relation may be presented in the more intuitively clear form as “SC Model State consists of SC Relation State”.

Terms are intentionally marked out by capital letters to be able to discern their from the text. The copula “consists of ” may be replaced by any word-combination that, by opinion of a specialist in the given UoD, reflects the essence of interrelation between the terms. Frequently, a designer has possibility to choose from a set of such word-combinations. By this way of relation verbalization, we achieve intuitive clarity and partially self-documentation of the semantically complete models.

A verbal declaration of semantically complete relation simultaneously represents declarations of terms participating in this relation. There is no necessity to separately declare existence of the terms since existence of the relation itself tells about presence of these terms, and, at the same time, word-combinations designating the terms represent their short descriptions.

Since any map is a relation, each map declaration may be realized by the same way as for relations. The fact that the given relation is the map we reflect by the designation “[→]” at the end of the relation verbalization. For example, as follows: “SC Model State is included in SC Model [→]”.

Thus, properties of the semantically complete models let us to significantly simplify each propositional logic expression applied to such a model, and to advance the semantically complete model representation to intuitively clear form. It is undoubtedly important step on the way of developing of conceptual modeling methods.

4. An Example of Semantically Complete Model Application

As an example of semantically complete model use for conceptual modeling, consider the formalization of semantically complete model itself. During creating of the semantically complete model formalization presented above, we took into account all the requirements of the semantically complete modeling. Therefore, presenting of that formalization by the semantically complete way does not require structural changes of it, numbering and essence of expressions are kept. Verbal interpretations of these expressions should be taken from the formalization presented above.

Further, a semantically complete model of semantically complete model itself, formed according to early described constraints and rules, is presented. “SC” is used for shortening of the word-combination “semantically complete”. Introducing of this abbreviation is motivated by striving to make the expressions less bulky, and, at the same time, to keep the sensible sound of term verbalizations.

A Sense is related to a Term [→]

[SCM1]: $\exists_1 Term\{(Sense - Term)\}, \exists Sense\{(Sense - Term)\}$

A Sense is coded by a Value [→]

[SCM2]: $\exists_1 Value\{(Sense - Value)\}$

[SCM3]: $\exists_1 Sense\{(Term - Sense - Value)\} \vee \overline{\exists Sense\{(Term - Sense - Value)\}}$

[SCM4]: $\exists_1 Term \exists_1 Value\{(Term - Sense - Value)\}$

A Relation Position contains a Term [→]

A Relation Position is related to a SC Relation $[-\rightarrow]$

[SCM5]: $\exists_1 \text{Term} \{ \{ \text{Relation Position} - \text{Term} \} \}, \exists_1 \text{SC Relation} \{ \{ \text{Relation Position} - \text{SC Relation} \} \}$

[SCM6]: $\exists \text{Relation Position} \{ \{ \text{Relation Position} - \text{SC Relation} \} \}$

$\exists \text{Relation Position}_2 \exists \text{SC Relation} \exists \text{Term}$

[SCM7]: $\left\{ \begin{array}{l} \text{Relation Position}_1 \neq \text{Relation Position}_2 \rightarrow (\text{SC Relation} - \text{Relation Position}_1 - \text{Term}) \wedge \\ (\text{SC Relation} - \text{Relation Position}_2 - \text{Term}) \end{array} \right\}$

Let $\text{SCRTermSet}(\text{SC Relation}) \equiv \{ \text{Term} \mid \exists \text{Relation Position} \{ \{ \text{Term} - \text{Relation Position} - \text{SC Relation} \} \} \}$.

[SCM8]: $\overline{\exists \text{SC Relation}_2} \{ \text{SC Relation}_1 \neq \text{SC Relation}_2 \wedge \text{SCRTermSet}(\text{SC Relation}_1) = \text{SCRTermSet}(\text{SC Relation}_2) \}$

[SCM9]: $\overline{\exists \text{SC Relation}_2} \left\{ \begin{array}{l} (\text{SC Relation}_1 - \text{SC Model} - \text{SC Relation}_2) \wedge \text{SC Relation}_1 \neq \text{SC Relation}_2 \wedge \\ \text{SCRTermSet}(\text{SC Relation}_1) \subset \text{SCRTermSet}(\text{SC Relation}_2) \end{array} \right\}$

A SC Model consists of SC Relations

A SC Cohesion consists of Senses

[SCM10]: $\exists \text{Sense} \{ \{ \text{SC Cohesion} - \text{Sense} \} \}$

[SCM11]: $\overline{\exists \text{SC Cohesion}_2} \left\{ \begin{array}{l} \text{SC Cohesion}_1 \neq \text{SC Cohesion}_2 \wedge \\ \{ \text{Sense} \mid (\text{SC Cohesion}_1 - \text{Sense}) \} = \{ \text{Sense} \mid (\text{SC Cohesion}_2 - \text{Sense}) \} \end{array} \right\}$

A SC Relation State belongs to a SC Relation $[-\rightarrow]$

[SCM12]: $\exists_1 \text{SC Relation} \{ \{ \text{SC Relation} - \text{SC Relation State} \} \}, \exists \text{SC Relation State} \{ \{ \text{SC Relation} - \text{SC Relation State} \} \}$

A SC Relation State consists of SC Cohesions

[SCM13]: $\exists_1 \text{SC Relation State} \{ \{ \text{SC Cohesion} - \text{SC Relation State} \} \}$

[SCM14]: $\overline{\exists \text{SC Relation State}_2} \left\{ \begin{array}{l} \text{SC Relation State}_1 \neq \text{SC Relation State}_2 \wedge \\ \{ \text{SC Cohesion} \mid (\text{SC Relation State}_1 - \text{SC Cohesion}) \} = \\ \{ \text{SC Cohesion} \mid (\text{SC Relation State}_2 - \text{SC Cohesion}) \} \end{array} \right\}$

[SCM15]: $\exists_1 \text{SC Relation State} \{ \{ \text{SC Relation State} - \text{SC Relation} \} \} \wedge \forall \text{SC Cohesion} \{ \overline{\{ \text{SC Relation State} - \text{SC Cohesion} \}} \} \vee \overline{\exists \text{SC Relation State} \{ \{ \text{SC Relation State} - \text{SC Relation} \} \} \wedge \forall \text{SC Cohesion} \{ \overline{\{ \text{SC Relation State} - \text{SC Cohesion} \}} \} \}$

[SCM16]: $\exists_1 \text{SC Relation} \exists \text{SC Relation State} \{ \{ \text{SC Cohesion} - \text{SC Relation State} - \text{SC Relation} \} \}$

[SCM17]: $\{ \{ (\text{Sense} - \text{SC Cohesion} - \text{SC Relation State} - \text{SC Relation} - \text{Relation Position} - \text{Term}) \} \rightarrow (\text{Sense} - \text{Term}) \}$

A SC Model State relates to a SC Model $[-\rightarrow]$

[SCM18]: $\exists_1 \text{SC Model} \{ \{ \text{SC Model State} - \text{SC Model} \} \}, \exists \text{SC Model State} \{ \{ \text{SC Model State} - \text{SC Model} \} \}$

A SC Model State consists of SC Relation States

[SCM19]: $\exists \text{SC Relation State} \{ \{ \text{SC Relation State} - \text{SC Model State} \} \}$

[SCM20]: $\left\{ \begin{array}{l} (\text{SC Relation} - \text{SC Model} - \text{SC Model State}) \rightarrow \\ \exists_1 \text{SC Relation State} \{ \{ \text{SC Relation} - \text{SC Relation State} - \text{SC Model State} \} \} \end{array} \right\}$

[SCM21]: $\left\{ \begin{array}{l} (\text{SC Relation State} - \text{SC Model State} - \text{SC Model}) \rightarrow \\ \exists_1 \text{SC Relation} \{ \{ \text{SC Relation State} - \text{SC Relation} - \text{SC Model} \} \} \end{array} \right\}$

Compare the attained formalization of semantically complete model with the previous one. As we can see, the declarations of terms and relations have more intuitively clear from than in the case of using of abbreviations. The expressions have the simpler structure, but they are partially bulky because of verbose names of used terms. The absence of necessity to patently refer to relations leads to the expression structure being close to its verbal formulation.

From our point of view, the last formalization is more preferable since it is based exclusively on UoD term names without artificial abbreviations. It lets to lead an analysis and formalization by

means of the same term designations. The maximal closeness of relation declarations and expression formalizations to the structure of natural language sentences, and the unconditional character of relations let to increase the effectiveness of designer's work and to improve the quality of communication among project participants.

5. Conclusion and Further Work

Any semantically complete model is only based on two structural elements: terms and semantically complete relations. It is fully devoid of any redundant elements. What is more, according to the expressions [SCM1] – [SCM21], not every relation set based on some terms may be treated as a semantically complete model. Therefore, it should be asserted that any semantically complete model has less quantity of equivalent representations than an appropriate object-role model or set-theoretic model for the same UoD. It lets to hasten the study of this conceptual modeling technique, to increase a designer's working efficiency during processes of modeling, queries and constraints formulating, checking of correspondence between a semantically complete model and its realization by way of a data form model.

Relations of semantically complete model are based on unique sets of terms, and can not be included one in another in the framework of the same model. This property leads to unconditional character of semantically complete relation: if one knows that a semantically complete relation exists, one can assert that this relation completely describes the interrelation between the underlying terms, and there is no another relation in the same model that describes this interrelation in a different way. To use the semantically complete models, it is enough to know UoD terms and its interrelations, it is not required to be familiarized with relation applicability for query formulating in some external for model conditions. As a result, we achieve simplicity of model familiarization, its controllability and scalability.

The corollary of semantically complete relation absoluteness is the possibility of expression formulating exclusively by means of UoD terms. To extract a connection between some terms included into one relation, it is enough to enumerate these terms. Therefore, one can assert that queries to any semantically complete model may be formulated exclusively by means of terms without necessity to explicitly refer to used relations. This property lets to simplify queries and constraints formulation, to advance their formalizations to natural language sentences. It leads to increasing a designer's working efficiency and increasing efficiency of communication among project participants.

At present, the author is investigating fundamentals of a conceptual query and manipulation language permitting to formulate queries and modifications for any semantically complete model exclusively by means of UoD terms. Also it is fulfilling the research of using of the semantically complete models as the logical interface for access to data and knowledge that are physically stored in the framework of the relational schemes. On this way, the author analyses and formalizes transformations of queries and modifications formulated using developing conceptual language to queries and modifications formulated using SQL.

6. References

1. van Bommel P., ter Hofstede A.H.M., van der Weide. Semantics and verification of object-role models // *Information Systems*, 16(5), 1991, pp.471-495.
2. Bronts G.H.W.M., Brouwer S.J., Martens C.L.J., Proper H.A. A Unifying Object Role Modelling Approach. *Information Systems*, 20(3), 1995, pp. 213-235.
3. Brouwer S.J., Martens C.L.J., Bronts G.H.W.M., Proper H.A. Towards a Unifying Object Role Modelling Approach // *Proceedings of the First International Conference on Object-Role Modelling (ORM-1)*, Magnetic Island, Australia, 1994, pp. 259-273.
4. Campbell L.J., Halpin T.A., Proper H.A. Conceptual Schemas with Abstractions – Making flat conceptual schemas more comprehensible // *Data & Knowledge Engineering*, 20(1), 1996, pp. 39-85.

5. Creasy P.N., Proper H.A. A Generic Model for 3-Dimensional Conceptual Modelling // Data & Knowledge Engineering, 20(2), 1996, pp. 119-162.
6. J.J. van Griethuysen, editor. Concepts and Terminology for the Conceptual Scheme and the Information Base. Publ. nr. ISO/TC97/SC5-N695, 1982.
7. Halpin T.A., Orłowska M.E. Fact-Oriented Modelling for Data Analysis // Journal of Information Systems, 2(2), 1992, pp. 1-23.
8. Halpin T.A., Proper H.A. Database scheme transformation and optimization. // Proceedings of the OOER'95, 14th International Object-Oriented and Entity-Relationship Modelling Conference, vol. 1021 of Lecture Notes in Computer Science, pp. 191-203, Gold Coast, Australia, 1995.
9. Halpin T.A. Conceptual Schema and Relational Database Design. Prentice-Hall, Sydney, Australia, 2nd edition, 1995.
10. Halpin T.A. An ORM Metamodel // Journal of Conceptual Modeling, 16, 2000.
11. ter Hofstede A.H.M., van der Weide. Expressiveness in conceptual data modeling // Data & Knowledge Engineering, 10(1), 1993, pp. 65-100.
12. ter Hofstede A.H.M., Proper H.A., van der Weide. Formal definition of a conceptual language for the description and manipulation of information models // Information Systems, 18(7), 1993, pp. 489-523.
13. ter Hofstede A.H.M., Proper H.A., van der Weide. A Conceptual Language for the Description and Manipulation of Complex Information Models. In Seventeenth Annual Computer Science Conference, vol. 16 of Australian Computer Science Communications, 1994, pp. 157-167.
14. Nijssen G.M., Halpin T.A. Conceptual Schema and Relational Database Design: a fact oriented approach. Prentice-Hall, Sydney, Australia, 1989.
15. Shoval P., Zohn S. Binary-relationship integration methodology // Data & Knowledge Engineering, 6(3), 1991, pp. 225-250.
16. de Troyer O.M.F. The OO-Binary Relationship Model: A Truly Object Oriented Conceptual Model. // Proceedings of the Third International Conference CaiSE'91 on Advanced Information Systems Engineering, vol. 498 of Lecture Notes in Computer Science, Trondheim, Norway, 1991, pp. 561-578.
17. de Troyer O.M.F. A formalization of the Binary Object-Role Model based on logic // Data & Knowledge Engineering, 19(1), 1996, pp. 1-37.