The Orderliness and Precision in Conceptual Modelling

Elita Pakalnickiene, Lina Nemuraite, Bronius Paradauskas

Kaunas University of Technology, Studentu 50-315a, LT 51368 Kaunas, Lithuania,
elita.pakalnickiene@gmail.com, lina.nemuraite@ktu.lt, bronius.paradauskas@ktu.lt

Abstract

The quality of conceptual models and processes for their development may be considerably improved bringing the ordering to their structure and analysis process. The method is proposed for development of ordered conceptual model and its enrichment with integrity constraints. The ordering has many advantages: conceptual model conforms to normal forms and ontological foundations of conceptual models; it has the test for the conformance to the observed reality; all three steps – discovery of entities, constraints and testing – are completed by the similar and natural process. The discovery of integrity constraints is supported by the taxonomy created on the base of analysis of the most promising methods for conceptual modelling. Resulting models comprise lattices of formal concepts.

Keywords: conceptual model, integrity constraint, ordering, entity, interface, normalization, UML.

1. Introduction

Conceptual schemas and conceptual models of Information Systems (IS) are under consideration a long before, however their quality remains the challenge for developers as requirements for quality are growing due to the Semantic Web, Model Driven Development and Service orientation. Conceptual models are an urgent part of requirements specifications and make the foundation for the overall IS development process.

The method for development of the Ordered and Precise Conceptual Model (OPCM) represented in this paper pursues two important purposes: a creation of the ordered conceptual model and its enrichment with integrity constraints. Being the consequent extension of the proposal [Nemuraite et. Al. (2005)]), the OPCM comprises the IS state related part (i.e. Data Model) of the Design Independent Modelling (DIM) ([Ceponiene et. Al. (2005)]) that is based on IS fundamentals [ISO/TR 9007 (1987)], according which the conceptual schema of information system must include state, behaviour, constraints and derivation rules.

Use case driven methods for development of conceptual models are widespread, but they lack formal rules and have faced the criticism during the last years (e.g. [Svetinovic et. Al. (2006)]). The suggested ordering of use cases according the descending timescales of life cycles of types of objects obtained as results of transactions performed by these use cases not only defines rules for discovery of interfaces and classes of data entities but also approaches to the matter of time-related object identity. The object identity is one of the core concepts recognized in ontological research; however, object identity is missing in UML conceptual models. We do not keep in mind "default" object identifiers, but the true identity, the uniqueness of domain objects. We argue that the object identity should be explicit in well formed, precise conceptual model.

Our goal is to represent the method for development of ordered conceptual models with integrity constraints and to unfold its possibilities. The rest of the paper is organized as follows. In section 2 methods for creation of conceptual schemas of information systems are investigated. In section 3 the method is presented for development of conceptual model using interface-based representation of use cases. Section 4 proposes process for adding and testing integrity constraints. In section 5 the normalization of resulting model is demonstrated by representing it as a lattice of formal concepts. Finally, section 6 concludes the paper and discusses future work.

2. Related Work

IS conceptual schema can be constructed in many different ways, from different sources; however, the comprehensive methodology is missing till now. Existing methods may be divided into several groups.

Methods, based on the analysis of the natural language currently are ineffective, but have promises for the future. Techniques for discovering classes perform grammatical analysis of requirements specifications searching for nouns, adjectives and verbs, which are candidate classes, attributes and services of the system; in [Vongdoiwang et. Al. (2006)] ontologies for this purpose are used. The comparison of use case driven approaches with grammatical analysis of textual requirements [Anda et. Al. (2005)] results in the conclusion that class diagrams constructed from use cases are structured better, but a clear technique is needed for transition from use cases to class diagrams.

Methods, based on the analysis of use cases. Use cases are important in the identification of classes and associations between them. A use case specifies the sequences of interactions that an actor should be able to perform to achieve his goal − to gain the valuable result − in collaboration with the system under development and, possibly, other systems. In Unified Process and ICONIX class diagrams are constructed from use case diagrams, but a little attention is given to a discovery of entities of problem domain. A Newtonian approach [Roussev et. Al. (2002)] presents

method where use cases are described by state machines and every state is associated with a business object that is gained or lost by the action performed during the transition to that state. Besides many useful suggestions, the method loses its advantages when trying to apply it to information systems where information interchanges take a place instead of interchanges of economic value. Even more rational principles are proposed by the goal-driven methodology where data entities are derived from actor goals [Gustas et. Al. (2003)], but the main drawback of these methodologies is the difficulty to applying them to domains other than business trade.

Methods using elementary concepts. A method suggested by Halpin for his Object-Role Modelling [Halpin (2001)] differs from others by taking into account integrity constraints and by making checking of obtained artefacts to ensure their conformance to data of problem domain. The process is quite intuitive. Also, the opinion of ontology researchers is that elementary concepts are insufficient for IS models.

Ontological approaches. For achieving the better quality of conceptual models, the theoretical foundations for the order of concepts and relationships in conceptual modelling may be transposed from the ontological research. The most significant approaches are Bunge-Wand-Weber Ontology [Wyssusek et. Al. (2005)] and General Ontological Language (GOL) [Guizzardi et. Al. (2004)]. The both methodologies are similar though they are using different notions. Currently ontological methodologies yet have not defined all rules necessary for conceptual modelling; moreover, ontological rules are declarative, they do not prescribe how to achieve the needed qualities. It is worth to mention, that the ontological principles of identity, rigidity, unity, dependency, order, and others are in account by the OPCM.

Methods dealing with integrity constraints. Integrity constraints are first class citizens of conceptual models [Debenham (1997)]. However, no comprehensive methodology for discovery of integrity constraints is established till now. In [Miliauskaite et. Al. (2005b)] taxonomy of integrity constraints was proposed based on analysis of types of constraints in the most promising conceptual modelling methods (ER, Extended ER (EER), UML, eXexutable UML (xUML), and ORM). We have revealed the types of constraints that are important for well-formed conceptual models, and applied them for conceptual modelling in UML and OCL. UML is easy extensible with stereotypes that were established in [Miliauskaite et. Al. (2005a)] for visual modelling of all types of taxonomical constraints.

The OPCM development process is based on the sequential analysis of interface-based representation of use cases having semantics of business transactions. It is different from other methods as institutes the order of analysis steps and results in ordered conceptual schema. This is important aspect that makes conceptual schema well-formed, normalized and easy readable; the completeness of use case model and conceptual schema is ensured by cross-checking entities and interface operations. The

resulting conceptual model is semantically enriched by integrity constraints, and subjected for verification of its conformance to valid states of problem domain.

3. Development of conceptual model using interface-based representation of use cases

The OPCM development consists of three steps: creation of ordered conceptual model; adding integrity constraints; testing constraints with instances of domain objects. In the first step, initial use cases represented by use case diagram and specifications are arranged to levels in descending order of timescales of lifecycles of results of business transactions performed by these use cases. The method is demonstrated by an example of Production Information System (Figure 1), in which each product has the sequence of operations of predefined types needed to produce it. Product may be produced according to the plan by executing planned tasks. Persons are divided into managers and employees. Each task is associated with employee and inspector which must be the manager of the assigned employee. Every use case is represented by the interface; activities performed during interactions of actors and use cases are represented by data-specific operations *Create*, *Read*, *Update*, *Delete* (CRUD) of corresponding interfaces.

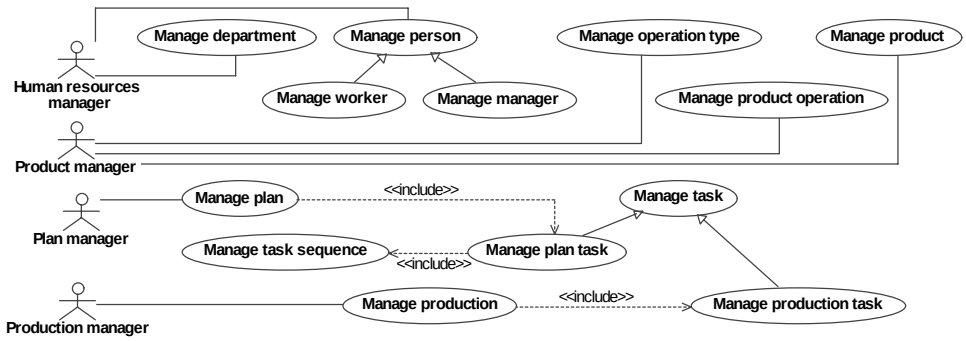
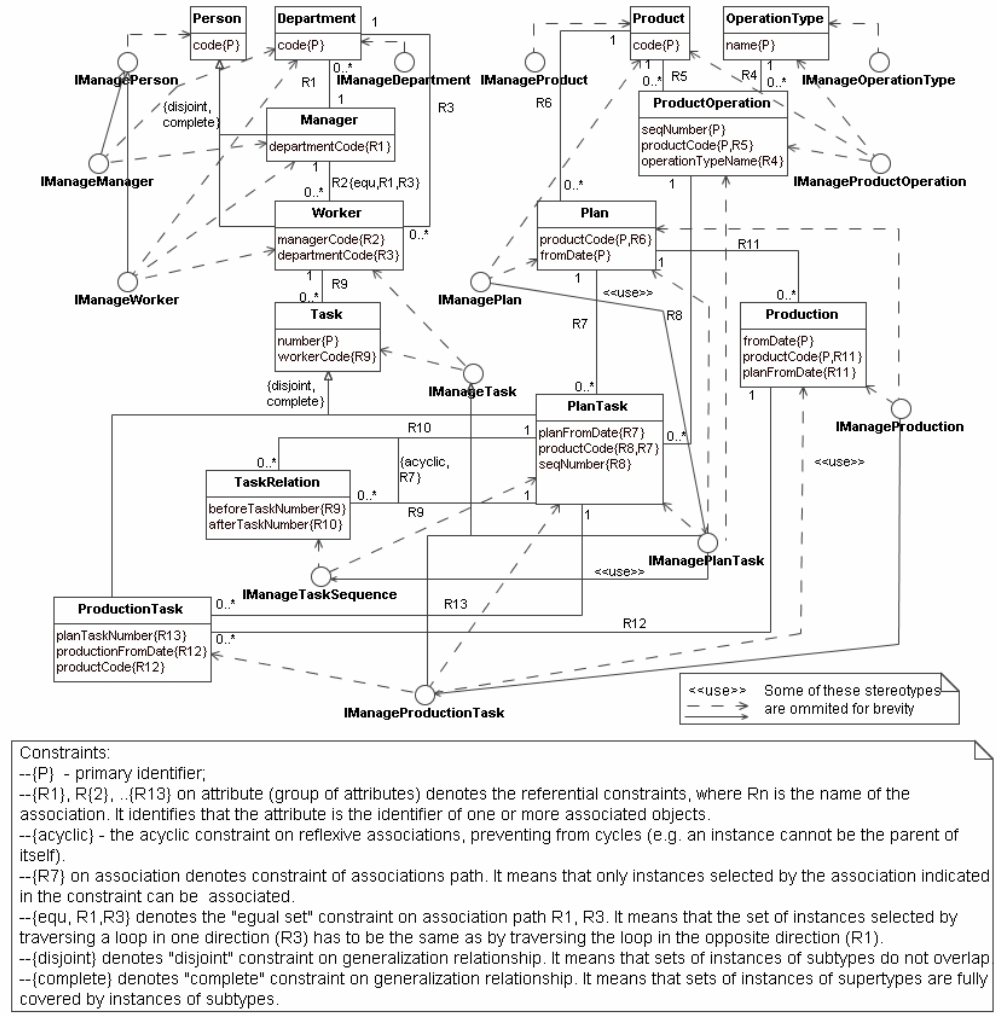


Figure 1. *Use cases of production information system*

Objects collaborating in operations of interfaces are identified from specifications of use case steps, as well as their pre and post conditions. Objects participating in post condition and output of the operation are existence-dependent from objects in precondition and input of this operation. New objects and interfaces are gradually introduced into object diagram starting from use cases of the top level.

The conceptual model is created by mapping objects to entities and their values to attributes (for brevity, only essential attributes – identifiers of entities and their dependencies from other entities – are shown on Figure 2). Entities and interfaces are ordered according dependencies between objects and operations. If objects are

existent-independent but must be related, the new entity must be added to represent this relationship. Existence dependency between entities is represented by the mandatory association of the dependent entity to the base entity.



Constraints:
--{P} - primary identifier;
--{R1}, R{2}, ..{R13} on attribute (group of attributes) denotes the referential constraints, where Rn is the name of the association. It identifies that the attribute is the identifier of one or more associated objects.
--{acyclic} - the acyclic constraint on reflexive associations, preventing from cycles (e.g. an instance cannot be the parent of itself).
--{R7} on association denotes constraint of associations path. It means that only instances selected by the association indicated in the constraint can be associated.
--{equ, R1,R3} denotes the "egual set" constraint on association path R1, R3. It means that the set of instances selected by traversing a loop in one direction (R3) has to be the same as by traversing the loop in the opposite direction (R1).
--{disjoint} denotes "disjoint" constraint on generalization relationship. It means that sets of instances of subtypes do not overlap
--{complete} denotes "complete" constraint on generalization relationship. It means that sets of instances of supertypes are fully covered by instances of subtypes.

Figure 2. *Class diagram with integrity constraints representing entities and interfaces of Production IS*

Generalization relationship occurs when object after operation gains new values and links but its identity value remains unchanged. It is a rather subtle task to discover generalizations; often generalization is decided a priori or introduced later for the improvement of model. Relationship <<include>> between use cases is mapped to the

usage dependency (represented by dashed line and stereotype <<use>>), generalization – to generalization.

4. Adding and testing constraints

In conceptual model, integrity constraint is a logic formula, dependent from the problem domain, and it must hold for all meaningful states of the information system [Mellor et. Al. (2002)]. Integrity constraints are able to ensure that domain semantics will be precisely expressed in conceptual model – this quality is not achievable by visual languages. Integrity constraints support criteria of expressivity, clarity, simplicity, orthogonallity, semantic stability and relevance, validation/abstraction mechanisms, and formal foundation [Halpin (2001)]. For adding integrity constraints we will apply the same principles for gradual introduction starting from independent entities.

We will introduce constraints that are required in pre and post conditions for successful accomplishment of operations (the existence of operation parameters is implied by default). All constraints on attributes and their combinations, external dependencies, constraints on inputs, expressed in pre-conditions, remain valid for all lifecycles of objects, created during the operation. Consequently, they are invariants of these entities. Derivation rules specified in post conditions also may be specified in conceptual model. For example, in order to create an instance of the Person we must get not-empty first name, last name and person code; for creation of an instance of the Worker we must have two valid instances of the Person, one of them must be a Manager of the same Department where the second person is intending to become a Worker. The resulting invariant in OCL will be included in the conceptual model of Production IS (stereotype of the equal set constraint on the association R1 {equ, R2, R3} on Figure 2):

context Worker inv :

self.Manager.Department → exists(d : Department | d = self.Department)

Constraints on states are not relevant for this purpose and must not be included as invariants because these constraints must hold at the moment of activation of the operation, but after the accomplishment of that operation they may be irreversibly changed. The process for adding constraints is presented on Figure 3. This process also helps to resolve the problem of assigning invariants to entities: the invariant that is dependent on several entities is assigned to the entity with the shortest lifecycle.

Constraints are derived from requirements, and requirements are the obligations for the system to enforce some business rules. However, the knowledge about business rules is not enough for successful definition of all constraints. In every step of the process of adding constraints one should look for typical situations requiring

constraints. Using OPCM the amount of constraints is significantly decreased as only rigid (i.e. mandatory) properties are relevant for well- formed conceptual model.

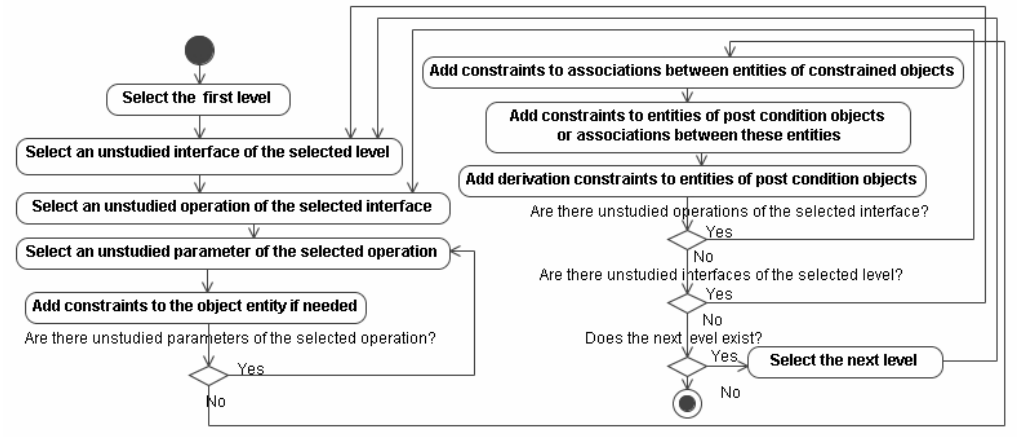


Figure 3. *Process for adding constraints*

Nevertheless, reflexive associations, generalizations, relationship paths, loops, derivations constraints are not avoidable by well-formed structures and require a special attention to revealing them. For example, reflexive relationship for entity plan task requires {acyclic} and path constraints because the concrete task cannot occur repeatedly after itself or several times in the sequence belonging to the same plan. This requirement is expressed with acyclic constraint {acyclic, R7} on Figure 2.

In the third step, we will analyze and test integrity constraints that must be satisfied in every state of the system, abstracting from state transition constraints that must be satisfied for application-specific state transitions. The testing of OPCM adhere the same principles as its construction: the sequential procedure starting from the top level. However, the test elements are of the finer granularity as every constraint is tested separately: constraints on attribute, attributes, entity, entities, relationship, relationships, and, finally, on the overall schema. The gradual testing practically may be integrated with creation of models; however, it is not a good practise to elaborate precisely a part of a model without viewing a sketch of its whole.

5. The ordering and normalisation

Subset, existence and usage dependencies of OPCM are reflexive, anti-symmetric, transitive and acyclic. It is possible to define the partial order relation, generalizing the previous dependencies and expressing: the subset dependency, if identifiers of object types are coincident; the usage dependency, if the relation associates interfaces or the interface and entity; the existence dependency, if the relation is hold between entities. If schema elements are not ranked as entities and interfaces, the usage and existence dependencies may overlap.

The ordered conceptual model created by the proposed method meets the conformance to normal forms similar to normal forms of the relational model. The purpose of relational database schema normalization is to preserve the compatibility during its evolution. Normalization allows avoiding data redundancy and anomalies of data renewal.

The principles of normalization of object model include high cohesion and low coupling, dependency inversion, Liskov substitution, open-closed, acyclic and stable dependencies, interface segregation and others. From our point of view it is worth to make normalization in conceptual level, because conceptual schema may have many implementations (relational, XML etc.). Ordered conceptual schema S of relations R has Projection/Join normal form if having constraints Σ for all schema components R_i,R_j the partial order relation $R_i \leq R_j$ holds $i,j \in [1,n]$, generalizing subset, existence and usage dependencies. Furthermore, schema is acyclic. This property is important for behavioural schemas and for implementing model by XML schema.

One of possibilities to verify the conformance of conceptual schema to normal forms is to present it with concept lattice and to apply methods of Formal Concept Analysis (FCA) [Priestley (2002)], [Godin et. Al. (2005)], [Priss (2006)]. The concept lattice for the formal context of Production IS is presented on Figures 4. The formal context has instances (shown in brackets on Figure 4). Each instance has a number and formal attributes related with it. In our example for validation of conceptual model fourteen instances of its components (comprising one complete instance of conceptual schema) are needed. Schema is invalid if it not satisfies relevant instances of problem domain. Practically, for validation of schema the much more instances are needed.

It is important to note that ordered conceptual schemas comprise concept lattices. This issue allows avoiding the growth of analysis scope problems that arise in constructing lattices by means of formal methods. Concept lattices may be used to analyze concept attributes, methods and constraints. Functions defined on concept lattices are idempotent so conformance to lattice confirms the normalization of conceptual schemas constructed using the proposed method.

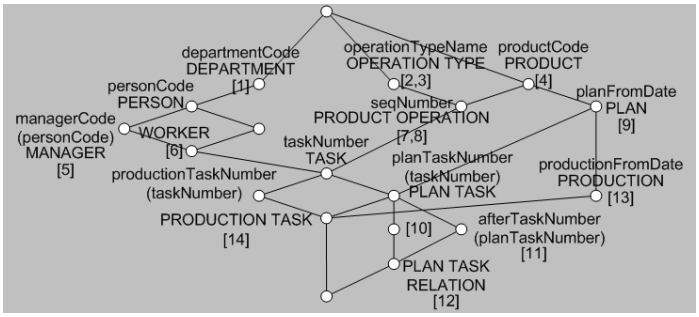


Figure 4. *A concept lattice for conceptual model on Figure 2*

5. Conclusion

The issues of this research confirm that the ordering and integrity constraints are capable to improve conceptual models making them normalized, testable for conformance to the problem domain, easier readable and understandable, and more complete. The existing methods for development of conceptual models and integrity constraints mostly are giving only declarative rules.

The interesting fact of this research is the evidence that empirical methods based on the years of practice are converging to formal methodologies as the resulting models obtained by the proposed method comprise lattices of formal concepts. Other fact is the correspondence between ordered models and ontological recommendations for conceptual modelling of information systems associated with the right identification, rigid classifiers, compound things, dependent roles and phases. However, many questions are not answered yet.

Our future work is addressed to looking for more ontological foundations from the one side, and for elaboration of support to the proposed method in CASE tools from the other side. While information technologies are encouraging, the CASE tools have not unfolded their potential and require tremendous efforts.

6. References

Anda B., Sjoberg D.I.K. (2005), *Investigating the Role of Use Case in the Construction of Class Diagrams*, Empirical Software Engineering, vol. 10, pp. 285-309.

Ceponiene, L., Nemuraitė, L. (2005), *Design independent modeling of information systems using UML and OCL*, in Databases and Information Systems: selected papers from the 6th International Baltic Conference on Databases and Information Systems Riga, Latvia, June 06-09, 2004. Amsterdam: IOS Press, 2005. ISBN 1-58603-485-5. pp. 224-237.

Debenham, J. (1997), *An analysis of Database Rules*, in Proc. IDEAS '97: International Database Engineering and Applications Symposium, Montreal, CANADA, ISBN: 0-8186-8114-4, pp.113–120.

ISO/TR 9007, (1987), *Concepts and Terminology for the Conceptual Schema and Information Base*, ANSI, New York, pp. 120.

Godin R., Valtchev P. (2005), *Formal concept analysis-based class hierarchy design in object-oriented software development*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg Publishers, ISBN 978-3-540-27891-7, pp. 304-323.

Guizzardi, G., Wagner, G., Guarino, N., Sinderen, M. (2004), *An Ontologically Well-Founded Profile for UML Conceptual Model,* in A.Person and J.Stirna (Eds.): CAISE 2004, LNCS 3084, pp. 112-126.

Gustas, R., Gustiene, P. (2003) *Towards the Enterprise engineering approach for Information system modelling across organisational and technical boundaries*, in Proceedings of the fifth International Conference on Enterprise Information Systems, vol. 3, Angers, France, pp. 77-88.

Halpin T. (2001), *Information modeling and relational databases: From conceptual analysis to logical design*, Morgan Kaufmann Publisher, ISBN 1-0155-8606-726.

Mellor S.J., Balcer M.J. (2002) *Executable UML. A foundation for model-driven architecture*, Addison-Wesley Publisher, Boston, ISBN 0-2017-4804-5.

Miliauskaitė, E., Nemuraitė, L. (2005a) *Representation of integrity constraints in conceptual models,* Information technology and control, Kauno technologijos universitetas, ISSN 1392-124X. Vol. 34-4, pp. 355-365.

Miliauskaite, E., Nemuraite, L. (2005b) *Taxonomy of integrity constraints in conceptual models,* P.Isaias et all. (Eds.): Proceedings of the IADIS Virtual Multi Conference On Computer Science and Information Systems, IADIS Press, ISBN: 972-8939-00-0, p. 247-254.

Nemuraite L., Paradauskas B. (2005), *From use cases to well structured conceptual schemas*, in Proc. of 13th international conference Information Systems Development: Advances in Theory, Practice, and Education, Vilnius, Lithuania, September 9-11, 2004. Vol. 28. New York : Springer Science+Business Media, pp. 303-314.

Priestley H.A. (2002), *Ordered Sets and Complete Lattices,* in International Summer School and Workshop: Algebraic and Coalgebraic Methods in the Mathematics of Program Construction, Lecture Notes in Computer Science, vol. 2297/2002, Springer Berlin / Heidelberg Publishers, pp.21-78

Roussev B. (2002), *Generating OCL Specifications and Class diagrams from Use Cases: A Newtonian Approach*, in Proc. HICSS'03: Proceedings of the 36th Hawaii International Conference on System Sciences.

Svetinovic D., Daniel M. BerryMichael W. Godfrey. (2006), *Increasing quality of conceptual models: is object-oriented analysis that simple*, in Proceedings of the 2006 international workshop on Role of abstraction in software engineering, Shanghai, China, pp. 19-22.

Vongdoiwang W., Batanov N. B. (2006), *An ontology-based procedure for generating object model from text description*, Journal Knowledge and Information Systems, Springer London Publisher, Issue vol. 10, Number 1, pp. 93-108.

Wyssusek, B., Klaus, H. (2005), *On the foundation of the ontological foundation of conceptual modeling grammars: the construction of the Bunge–Wand–Weber ontology*, in Proc. of the CAiSE '05 Workshops PHISE: 1st International Workshop on Philosophical Foundations of Information Systems Engineering, Porto, Portugal, pp. 583–593.