Reusability of Conceptual Models: The Problem of Model Variations

Pnina Soffer
Department of Management Information Systems
Faculty of Social Sciences
Haifa University
Haifa 31905, Israel
Phone: +972-4-8288506
Fax: +972-4-8288522
Email: spnina@is.haifa.ac.il

Irit Hadar
Department of Education in Technology and Science
Technion – Israel Institute of Technology
Haifa 32000, Israel
Phone: +972-4-8293101
Fax: +972-4-8213495
Email: hadari@tx.tecnion.ac.il

The total number of words in the paper: 3004

Submission Category: Research-In-Progress

Reusability of Conceptual Models: The Problem of Model Variations

Abstract

Conceptual models are aimed at providing a formal representation of a domain in the real world. However, it is commonly known that different people construct different models of a given domain. While prior research dealt mainly with modeling errors, this paper explores variations among correct models. Model variations, though frequently harmless, may have severe consequences when similarity of models is to be assessed for purposes of reuse.

This paper presents the problem of model variations and its effect on reusability. In particular, it suggests that in order to increase reusability the variations among models should be anticipated in advance. This could be achieved either theoretically or empirically. Corrective actions may include conclusive modeling rules and proper reuse mechanism design.

The paper suggests a theoretical framework for understanding the sources of variations and demonstrates an empirical investigation of variations and their reusability impacts when applied to a reuse application.

Keywords:

Empirical Software Engineering, Requirements Engineering, Conceptual Modeling, Reusability

1. Introduction

Conceptual modeling, defined as "the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication" [11], is applied in the early phases of IS analysis and design. A conceptual model reflects the real world independently of implementation technology and constraints [19]. Nevertheless, it is commonly known that different people usually present different models given the same domain. Two research areas have indirectly addressed these differences: the effect of the differences on database design is an issue addressed by schema integration research [4, 12, 13], and specific modeling decisions that may cause specific variation types are addressed by conceptual modeling research (e. g. [3, 6]).

Variations among "correct" models apparently seem to be harmless. However, in certain situations, such as model reuse, these differences might have severe implications on the outcome. In this paper we explore the phenomenon of model variations and its impact on model reusability.

Related empirical studies have evaluated the quality of models, identified types of modeling errors and discussed their possible sources [1, 2, 7, 14]. In general, various error types were observed and classified to errors related to the mapping of reality into modeling concepts, and syntax errors related to modeling

language grammar. Note, that the judgment of the models in these studies was based on a comparison of the models to one predefined "correct" model produced by an expert. No information was available as to whether all differences were considered as errors.

We believe that different models may still be considered correct, both semantically and syntactically. In what follows we relate to the reusability impacts of variations among correct models, and stress that they must be explicitly understood and considered in this context. We outline a theoretical framework explaining the sources of variations. Our approach is illustrated by a pilot study, demonstrating an empirical investigation of model variations and their impact in a reuse environment, and indicating possible corrective actions. We conclude by outlining directions for future research, based on our findings.

2. Reusability Implications

It may seem that variations among models, which are considered "correct" models, bear no practical consequences. However, their implications might be crucial when the similarity of conceptual models is assessed.

Similarity assessment of conceptual models serves for various purposes, such as identifying common characteristics of different information systems, reengineering and integrating them [4, 12, 13]. Its most common application is for reuse of conceptual models applied for assistance in obtaining requirement specifications or new conceptual models [8, 9, 18], for application development by relating the conceptual models to actual software components [15], and for analysis of the match between a given system and stated organizational needs [17].

In model similarity-based reuse, similarity assessment serves for retrieving existing reusable models from a repository, so they can be assembled, refined or customized to the needs of the current user. If modeling variations exist between a query model, which represents the user requirement, and the reusable model, they might prevent the identification of a suitable model to be retrieved. While reuse applications have been developed, employing various retrieval mechanisms, the variations expected among input models have not been, to the best of our knowledge, considered or investigated, and no effort has explicitly been done to overcome them.

In order to increase reusability one should (a) anticipate the expected variations in a given environment, (b) minimize them, and (c) minimize their effect.

The first step towards anticipating and minimizing the variations is to understand their sources. Minimizing their effect can be achieved through the design of reuse mechanism that takes into account the potential variations.

3. Sources of Variations Among Models

In this section we propose a framework for understanding the sources of model variations, based on theoretical foundations and on our observations. Variations among models generally appear due to the creative nature of the modeling activity, as well as other factors such as the richness of the spoken language [10], the ambiguities of modeling grammars, and others. Figure 1, which is our modification of the model

presented by [19], presents the factors that influence the model produced by an individual, and their interactions.

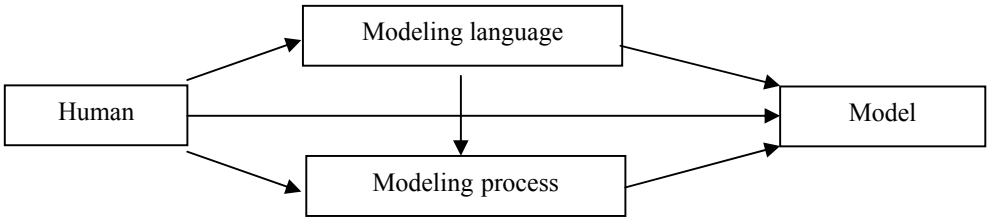

Figure 1: Factors that affect a conceptual model

Human: Human factors that may influence the model include individual perception and interpretation of reality, experience, and perception of model quality. These human factors influence the use of the modeling language through the modeling process, and, consequently, the resulting model.

Perception of reality – A model can only represent the way the modeler perceives the domain. Variations that belong to this category range from names assigned to entities, through differences in inheritance hierarchies, and to different interpretations of the role of certain entities in the domain. While these interpretations may be arguable, they reflect the way the modeler sees and understands the domain.

Experience – The effect of prior experience of the modeler on modeling errors has been studied by [1, 7]. Clearly, previous experience influences the way a modeling task is performed. Specifically, four aspects of the experience are of importance: amount of experience (expert vs. novice), experience in using the specific modeling language, experience in performing the specific task, and programming experience. While the influence of the total modeling experience and the experience gained in specific modeling languages is straightforward, the task and the programming experience need further clarification. Modelers whose experience is mainly in design tasks are likely to perform differently than modelers that mostly deal with conceptual modeling, even if both tasks are carried out using the same modeling language. Designers may find it hard to avoid design and implementation considerations, and their model is likely to be oriented towards implementation. While applying design considerations, programming experience may also affect the model, given certain design possibilities and constraints in the programming language the modeler is accustomed to.

Perception of model quality – Different modelers may have different opinions on the desired properties of a model. For example, some modelers may prefer simplicity in a model, while others perceive a very detailed model as being of a high quality.

Modeling language: Different modeling languages may incur different modeling variations, due to differences in their expressive power and set of constructs involved. According to [20], expressive power means completeness and clarity. Completeness means that a modeling language has all the constructs required for representing the domain. Its absence may drive different modelers to "invent" their own solutions in order to capture more information about the real world in the model. Clarity is the lack of three types of deficiencies: construct redundancy, construct excess, and construct overload. Construct redundancy, i.e., several ways of representing a single real-life phenomenon, definitely causes variations.

Construct excess, when a modeling construct stands for something which does not exist in reality, may also cause variations, by allowing the modeler to determine whether to use a construct or not. Construct overload, when the same construct represents more than one real-world phenomenon, affects model understanding rather than model variations.

Modeling process: Two important aspects of the modeling process, which are related to the modeling language but not dependent on it, are the chronological order of the modeling activities and the rules applied for mapping real life entities and phenomena into modeling constructs.

The importance of the order of modeling activities increases with the number of views in the modeling language. The modeling process involves iterations among the different views. The order in which these iterations are performed may yield differences in the focus of the model.

When using modeling languages that allow various levels of detail, refinement decisions are usually made by the modeler, who applies individual judgment rather than precise rules.

Most importantly, modeling grammars do not entail precise rules that conclusively define how to map real world phenomena into the modeling constructs. Hence, different mapping approaches may be taken.

4. Increasing Reusability: a Pilot Study

As discussed in Section 2, we propose that model variations should be considered when reuse applications are designed and applied. They should be anticipated and minimized by clear modeling rules and guidance regarding the modeling task and quality criteria. Their effect can be reduced by designing a robust reuse mechanism with respect to the anticipated variations.

To demonstrate our approach, we present a pilot study that was conducted in a reuse environment. The study included two phases: (a) obtain models of a given domain from different modelers and analyze their differences, and (b) apply the models to a reuse application and analyze the resulting differences in the retrieved output. The results indicate potential variation types in the given environment and their effect on reusability. These may lead to the design of the appropriate corrective actions.

4.1 Setting

The modeling language used in the study is Object-Process Methodology (OPM), described in detail in [5]. OPM employs two equally important classes of entities, objects and processes, which are connected by procedural links and structural relations. It uses a single graphic tool, the Object-Process Diagram (OPD) set, to model the major system's aspects, structure and dynamics.

OPM allows refinement of a model by zooming into entities (objects and processes) to enable a top-down analysis. The resulting model is a hierarchical OPD set, which specifies the structure and behavior of the system at a spectrum of detail levels.

The reuse application applied in the study is aimed at developing a model when aligning an ERP system to the needs of an enterprise. The input query includes a set of OPDs, each representing a requirement posed by an enterprise. These are matched against a model that represents all the alternative solutions available by the software package as a large OPD set. The reuse mechanism retrieves single OPDs, which are parts of the ERP model that best match the input requirements, providing matching scores to measure similarity

between the requirements and the retrieved OPDs.

The participants in the study were four PhD students, who are OPM experts. However, only one of them had experience in applying OPM for conceptual modeling in the ERP domain, while the other three were software design oriented, and their domain knowledge was limited. The participants received a textual description of requirements defined by a real organization regarding purchasing and inventory management modules, and were instructed to represent them as OPDs. The requirements addressed business processes as well as involved entities and their required relations and attributes. The resulting requirement models included 25-30 OPDs for each modeler.

4.2 Results

The variations among the models produced in the study were explored and categorized. In this section we present several examples of typical model variations that were found, discuss their sources and the effect they had on reusability, and indicate possible ways of preventing them. We focus on variation types not related to domain knowledge, some of which are specific to OPM models and some are more universal and may be related to other modeling languages as well.

Refinement decisions – OPM, similarly to other modeling languages (e.g., DFD, UML Use Case and Statecharts), allows refinement of an entity in a lower-level diagram, as a decision made by the modeler based on her individual judgment. For example, closing a Purchase Order, which includes closing the order lines and closing the order header, was refined into a lower level OPD by only one of the four modelers in our study, while the others included it in an upper-level OPD. The retrieval mechanism in the applied reuse application is designed to overcome these variations by aggregating lower-level details into upper-level diagrams when no other match is found. Hence, refinement decisions do not cause retrieval failure, but at the cost of significant additional processing time.

Entity naming – There are no defined, standard, globally accepted ontologies that conclusively name domain entities. Hence, the spoken language allows the modeler many choices of possible names. Although most objects have known, agreed names within a given organization, in many cases processes are not that accurately referred to (e.g. Enter Inventory Transaction vs. Register Inventory Transaction). Since this is a very common and well-recognized phenomenon [10], most reuse applications are designed to handle it by employing affinity or semantic similarity identification, and partly overcome the problem.

Abstract entities – Modelers may or may not use abstract entities, as illustrated in Figure 2. The Supplier object in Figure 2(a) is abstract, i.e., it has no instances besides the instances of its specializations, as opposed to the Supplier object in Figure 2(b), which has instances of its own. Abstract entities, when used, can take various forms and sizes of inheritance hierarchies, that again increase variations. The reuse application in our study takes the assumption that the query models are generally less detailed than the reusable models to be retrieved. Hence, abstract entities included in the reusable model and not in the query model do not harm retrieval success, but not vice versa. Such variations can be avoided by modeling rules to be employed when creating query models.

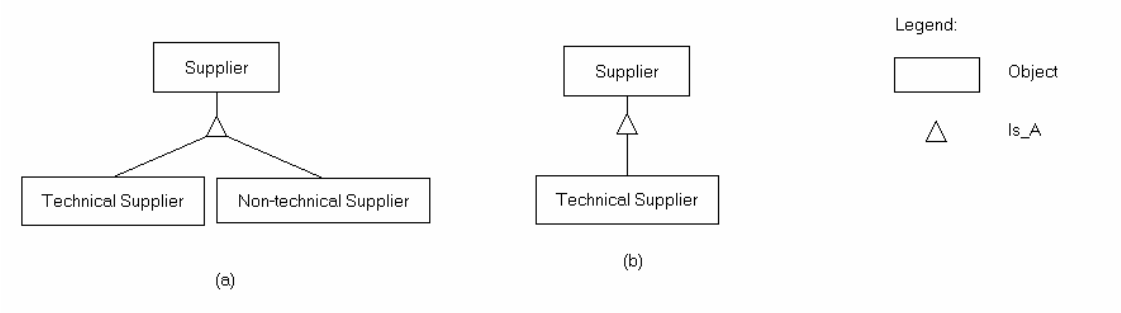

Figure 2: Abstract entities example

Applied design considerations – These may appear in different forms. For example, the effect link in Figure 3(a), although legitimate in design, is clearly an error in terms of conceptual modeling, since, as shown in Figure 3(b), an inventory transaction affects the inventory rather than the warehouse. However, the inventory data can be encapsulated in the warehouse object, in which case, from a design point of view, the model in Figure 3(a) is correct. Variations of this type caused retrieval failure, since the links between the entities are of different types. Clear modeling rules that regard encapsulation as design consideration to be avoided could prevent such variations and resulting retrieval failure.

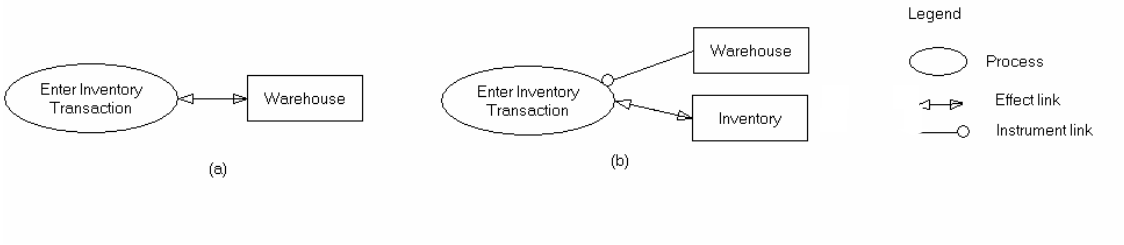

Figure 3: Design considerations example

Another example of design considerations (Figure 4) shows two different ways of representing the relation between an Item and a Supplier. In Figure 4(a) the supplier code is an attribute of the Item, whereas Figure 4(b) represents two structurally related objects, Item and Supplier. Figure 4(a) corresponds to relational data design style, while Figure 4(b) simply represents the relation without implying the implementation directions. The reuse application used in this study was designed to overcome this type of variations, by considering possible confusion between structural relations and characterization. However, not all reuse applications are designed similarly, and may fail in such cases. Again, clear modeling rules could prevent such problems.

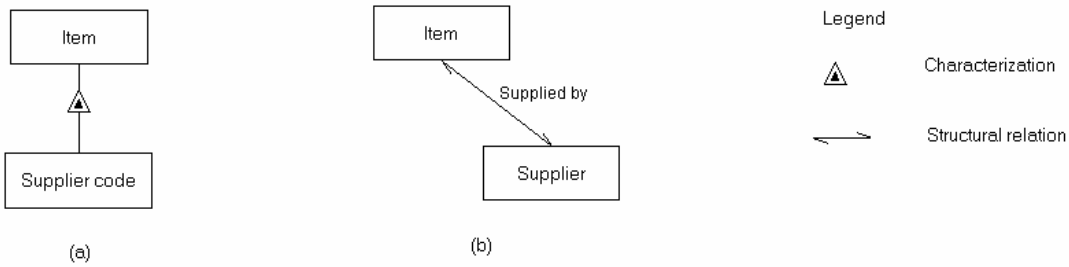

Figure 4: Attributes / relations example

Independence of processes – Processes in OPM can appear independently of any object, or as features of an object. For example, Insert Purchase Order may be regarded as a feature of Purchase Order or as an independent process, involving other activities such as supplier selection. The choice depends on the modeler's preference and interpretation of the domain, and can also reflect design considerations. This type of variation is clearly unique to OPM models. The effect on reuse was negligible, since perceiving a process as a feature of an object does not change its nature.

Invocation link – this is a modeling construct in OPM, denoting an immediate triggering of a process by the completion of another process. What actually triggers the process is an event, which is not modeled explicitly. An ontological analysis has identified the invocation link as a case of construct excess [16], and indeed, there were cases where some modelers used an invocation link while others represented a process triggered by an event. As a result, the matching score computed was low, yet retrieval was successful. Here again, clear modeling rules can be drawn in order to minimize these variations.

5. Discussion and Directions for Future Research

This paper discusses the phenomenon of model variations, and its effect on reusability of models. We stress that in order to increase reusability model variations should be anticipated, minimized, and taken into account in the design of reuse mechanisms.

The anticipation of model variations in a given environment can be achieved empirically, as demonstrated in the pilot study reported here, or through understanding the sources of variations that may occur in that environment, on the basis of the framework suggested in Section 3. More research is needed in order to establish a full understanding of the sources of variations. For example, in order to examine the quality perceptions of different modelers, a designated research can be conducted, including extracting such perceptions, finding their sources, inquiring their implications in different modeling languages and suggesting a strategy to achieve their uniformity.

The anticipation of variation types may affect the selection of a modeling language. Once a language is selected, the human factors that lead to the anticipated variations are beyond our control. The controllable factor that can be addressed as a corrective action is the modeling process. Variations can be reduced by providing clear guidance to the modeling process and the employment of the various modeling grammar constructs. A good example of efforts in this direction are the modeling rules [6], which define precisely the mapping of real world phenomena into UML constructs for the purpose of conceptual modeling. Applying such rules should reduce the model variations significantly, while different rational is still valid when UML is applied for design purposes.

Other rules that address the modeling process can contribute to the consistency among different views, uniformity in the use of modeling constructs and grammar, and well-defined refinement decisions. Naming standards and rules, which emerge in recent research efforts [10, 21], may reduce naming variations. Should these or futures suggestion of this kind be adopted, modeling languages will become more than a set of syntax rules, and will provide modeling methodology as well.

Regarding reuse and retrieval mechanisms, it is possible to design these to be robust to expected variations.

However, it is important to note that a robust retrieval mechanism is likely to be very complicated and its accuracy might be reduced. Therefore, reducing the variations in the models is preferable if possible.
Further research is currently being conducted by the authors, using other modeling languages, mainly UML, in order to gain additional insights regarding possible types, characteristics and impacts of modeling variations and to validate and generalize a theoretical framework that addresses them.

References:

[1] Agarwal, R. and Sinha, A. P., 1996, The Role of Prior Experience and Task Charasteristics in Object-Oriented Modeling: An Empirical Study, *Human-Computer Studies* 45, pp. 639-667.

[2] Batra D., 1993, A Framework for Studying Human Error Behavior in Conceptual Database Modeling, *Information and Management*, 25, pp. 121-131.

[3] Bodart F., Patel A., Sim M., and Weber R., 2001, Should optional Properties be Used in Conceptual Modeling? A Theory and Three Empirical Tests, *Information Systems Research* 12 (4), pp. 384-405.

[4] Castano S., De Antonellis V., Fogini M. G. and Pernici B., 1998, Conceptual Schema Analysis: Techniques and Applications, *ACM Transactions on Database System* 23(3), pp. 286-333.

[5] Dori, D., 2002, *Object Process Methodology – A Holistic Systems Paradigm*, Springer Verlag, Heidelberg, New York.

[6] Evermann J. and Wand Y., 2001, Towards Ontologically Based Semantics for UML Constructs, In: *Proceedings of ER 2001* (LNCS 2224), Springer-Verlag Berlin, pp. 354-367.

[7] Kim Y. and March S. T., 1995, Comparing Data Modeling Formalisms, *Communications of the ACM*, 38 (6), pp. 103-115.

[8] Lai L. F., Lee J. and Yang S. J., 1999, Fuzzy Logic as a Basis for Reusing Task-Based Specifications, *International Journal of Intelligent Systems* 14, pp. 331-357.

[9] Massonet P. and Lamsweerde A.V., 1997, Analogical Reuse of Requirements Frameworks, In *RE'97, Proc. of the Third IEEE Symposium on Requirements Engineering*, IEEE Press Los Alamitos CA. pp. 26-37.

[10] Moriarty T., 2000, The Importance of Names, *The Data Administration Newsletter* 15.

[11] Mylopoulos J., 1992, Conceptual Modeling and Telos. Chapter 2 in: Loucopoulos and Zicari (eds). *Conceptual Modeling, Databases, and CASE*. Wiley, pp. 49-68.

[12] Rahm E. and Bernstein P. A., 2001, A Survey of Approaches to Automatic Schema Matching, *The VLDB Journal* 10, pp. 334-350.

[13] Palopoli L., Sacca D., Terracina G., Ursino D., Uniform Techniques for Deriving Similarities of Objects and Subschemes in Heterogeneous Databases. IEEE Transactions on Knowledge and Data Engineering (to appear)

[14] Peleg M. and Dori D., 2000, The Model Multiplicity Problem: Experimenting with Real-Time Specification Methods, *IEEE Transactions on Software Engineering* 26 (8) pp. 742-759

[15] Pernici B., Mecella M. and Batini C., Conceptual Modeling and Software Components Reuse: Towards

the Unification, In: *IS Engineering: State of the Art and Research Themes*, Springer London, pp. 209-220.

[16] Soffer P., Golany B., Dori D., and Wand Y., 2001, Modelling Off-the-Shelf Information Systems Requirements: An Ontological Approach, *Requirements Engineering*, 6, pp. 183-198

[17] Soffer P., 2003, Aligning an Enterprise System with Enterprise Requirements: an Iterative Process, In: *Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS '03)*, (to appear).

[18] Sutcliffe A. and Maiden N. A., 1998, The Domain Theory for Requirements Engineering. *IEEE Transactions on Software Engineering* 24(3), pp. 174-196.

[19] Topi H. and Ramesh V., 2001, Human Factors Research on Data Modeling: Review of Prior Research and Suggestions for Future Directions, *EMMSAD'01, Proc. of the Sixth CaiSE/IFIP8.1 Int. Workshop on Evaluation of Modeling Methods in Systems Analysis and design*, pp. XV-1 – XV-14.

[20] Wand Y. and Weber R., 1993, On the Ontological Expressiveness of Information Systems Analysis and Design Grammars, *Journal of Information Systems* 3 pp. 217-237.

[21] http//:www.semanticweb.org