On Behavioral Model Quality and Transformation

— Position Statement —

Martin Gogolla
University of Bremen, Informatics Department
Postfach 330440, D-28334 Bremen, Germany
E-Mail: gogolla@informatik.uni-bremen.de

Abstract

This statement reflects experience of our group in conceptual modelling with Entity-Relationship and object-oriented approaches. It describes a general view on structural and behavioral aspects of conceptual models, their interrelationships, and their transformations, and sketches a proposal to build integrated models of them.

1 Conceptual Modelling of Information Systems

Frame for Conceptual Modelling: We assume design of structural and behavioral aspects of a complex information system is done in several successive steps where the degree of detail and closeness to implementation increases from step to step.

$$\langle S_0, B_0 \rangle \underset{T_1}{\rightarrow} \langle S_1, B_1 \rangle \underset{T_2}{\rightarrow} \ldots \underset{T_n}{\rightarrow} \langle S_n, B_n \rangle$$

Each $\langle S_i, B_i \rangle$ is a conceptual model with structural (S_i) and behavioral (B_i) aspects, and each T_i is a (semi-automatic) transformation of conceptual model $\langle S_{i-1}, B_{i-1} \rangle$ to conceptual model $\langle S_i, B_i \rangle$ guaranteeing that $\langle S_{i-1}, B_{i-1} \rangle$ is equivalent to $\langle S_i, B_i \rangle$ in the sense that the relevant properties (RP) of $\langle S_{i-1}, B_{i-1} \rangle$ also hold in $\langle S_i, B_i \rangle$.

$$\underbrace{\{\ p_i \mid \langle S_i, B_i \rangle \models p_i\ \}}_{\alpha} \quad \underbrace{\models}_{\beta} \quad \underbrace{RP(\{\ p_{i-1} \mid \langle S_{i-1}, B_{i-1} \rangle \models p_{i-1}\ \})}_{\gamma}$$

Each p in $\{\ p \mid \langle S, B \rangle \models p\ \}$ is a property which is valid in $\langle S, B \rangle$. The complete formula requires that a relevant property (γ) of conceptual model $i-1$ must be valid (β) with respect to the properties (α) of conceptual model i.

Languages for Conceptual Modelling: The languages for describing the conceptual models on the different levels may be quite different. The spectrum ranges from (A) languages with loose semantics over (B) languages allowing to state rather abstract but precise requirements to (C) languages which are efficiently implemented. A typical order and choice could be (1) informal requirements, (2) classical (merely structural) Entity-Relationship schemas [Che76, GH91, Gog94], (3) UML- or OMT-like descriptions (with behavioral aspects) [RBP$^+$91, BJR97], (4) formal object specifications (e.g. TROLL light [GCH93, GHC$^+$94, RG97] or other object specifications [SSE87, KMSJ88, BBE$^+$90, JSHS91, Wie91, DDP93, LEW94, GMB94]) with varying grade of detail and abstraction, (5) ODMG descriptions [Cat96], (6) programming language code for a concrete DB system [Deu91, LLOW91]. Especially formal specification languages could be used more than one time in one of the above steps.

Transformation Correctness: The transformations, the respective correctness assertions (checking that the next conceptual model has the relevant properties the previous conceptual model required) and their formality depend on the level under consideration ("Which are the relevant properties"). The correctness assertions may include for example (1) checking that an ER schema graph is represented by portions of an UML schema graph, (2) formal correctness proofs between formal specifications of different levels, (3) assuring that an ODMG description is capable of implementing some formal specification, and (4) checking that a concrete implementation satisfies an ODMG specification.

Formal Specifications: Of central concern in the complete picture are formal specifications. On the one hand, they are capable of describing structure and behavior aspects in a general and yet precise manner. On the other hand, they are able to abstract from irrelevant implementation details in order to guarantee a high degree of clearness. In the following we concentrate on the qualities a behavioral model presented in a formal object specification language has and on the question how to transform such specifications in order to bridge the gap between generality and abstraction on the one side and efficiency and implementability on the other side.

2 Behavioral Model Qualities

Closeness to Real World: One possible choice for describing behavioral (and structural) conceptual models $\langle S, B \rangle$ are formal object specification languages (e.g. TROLL light). Formality seems to us a necessary and therefore also a desirable quality because other features in our picture (like formal equivalence proofs and semi-automatic transformations) heavily rely on solid formal foundations. TROLL light allows to be close to real world in being able to directly express properties (attributes) and transitions (events, methods) of respective real world objects. In TROLL light, the objects are arranged in a whole-part hierarchy where complex objects can have simpler ones as subobjects. Object states are restricted by constraints, and derived attributes are described by derivation rules. For the behavior model, valuation axioms determine the effect of events on attributes, interaction rules specify object communication, and behavior patterns describe allowed life cycles of event sequences. In the language, we have a certain degree of intra-object concurrency for complex objects whereas atomic objects are not able to execute transitions concurrently.

Comparison of Behavioral Model Power: In TROLL light, different behavioral models can be compared with respect to their event interfaces (signatures) and the corresponding specified allowed life cycles. Given a certain set of events, it can be checked whether a life cycle of one behavioral model can also be represented in another one. This gives a formal means to compare different behavior models.

Computational Completeness: The question of computational completeness can be handled in our approach on the level of the underlying data types. In our object specification language we have a distinction between data instances (values) and object type instances (objects). Properties of objects can be described by data-valued attributes belonging to certain application dependent data types. The data types are assumed to be specified with a general data type specification language. Even languages with modest forms of axioms like conditional equations are capable of expressing every computable function. Computational completeness also concerns the derivation rules for derived attributes. Recursively derived attributes are present in the language, and if we use them, then any computable function can be described in this way.

3 Behavioral Model Transformation

Normal Forms: One well-established general transformation technique is the definition of normal forms. With respect to the behavior patterns of our behavioral models, classical automata theory and the area of process theory provides well-founded mechanisms to transform these patterns into equivalent but minimal ones establishing thereby the possibility to automatically compute meaningful conversions.

Quality Measure and Quality Assertion: In our approach, quality measure and quality assertion can be treated by validation and verification. For us, validation refers to checking whether properties formulated in an informal or implicit way hold in a conceptual model. Validation is typically done by an animation tool allowing to observe structural and behavioral properties of specifications. In contrast, verification is directed to checking whether properties given in a formal notation can be derived in a formal way from the specification, for instance, by theorem proving techniques. These same techniques for guaranteeing quality of single conceptual models can be applied for checking information preservation through transformation steps. For different levels, interface (signature) comparisons can be tested, and validation of informal properties and verification of formal properties can be carried out.

Specification Transformation: Design transformation are based in our scenario on notions of equivalence. For example, in TROLL light, specification transformation can be done by resolving static constraints into preconditions of behavior patterns. By firstly stating constraints in a declarative way with respect to single states one achieves a well-founded overview on the desired properties. Later, one looks for ways to efficiently guarantee that the properties hold by moving the respective conditions into preconditions of transitions (events). Thereby, one can operationalize the declarative requirements.

Independence from Implementation Platform: By choosing a formal specification language as the central link between requirements and implementation, our approach allows for keeping independence from implementation platform in early steps as long as possible. Thereby we guarantee the possibility of reusing design steps for different implementation platforms or reusing specification parts for different applications.

Interaction between Structural and Behavioral Constructs: In our formal specification setting there is close interaction between structural and behavioral constructs. One of the central design decisions is the question which information to model as a data-valued attribute and which to model as an object type. Usually, one will choose the data type values to be rich in structure allowing complex value-returning but state- and time-independent operations ("A date is a value") whereas the objects can be 0 by data-valued attributes which change in time ("A person is an object whose properties change"). This will influence the signatures with respect to the question whether an operation will belong to the (structural) data type level or to the (behavioral) event level. A second point of such interaction is the use of derived attributes (of the structural level), for instance, for improving the clearness of the definition of preconditions in the behavior patterns (of the behavioral level).

References

[BBE+90] G. v. Bochmann, M. Barbeau, M. Erradi, L. Lecomte, P. Mondain-Monval, and N. Williams. Mondel: An Object-Oriented Specification Language. Département d'Informatique et de Recherche Opérationnelle, Publication 748, Université de Montréal, 1990.

[BJR97] G. Booch, I. Jacobson, and J. Rumbaugh. *UML Summary (Version 1.0).* Rational Corporation, Santa Clara, 1997. http://www.rational.com.

[Cat96] R.G.G. Cattell, editor. *The Object Database Standard: ODMG-93.* Morgan-Kaufmann, 1996.

[Che76] P.P. Chen. The Entity-Relationship Model – Towards a Unified View of Data. *ACM Trans. on Database Systems*, 1(1):9–36, 1976.

[DDP93] E. Dubois, P. Du Bois, and M. Petit. O-O Requirements Analysis: An Agent Perspective. In O.M. Nierstrasz, editor, *Proc. European Conf. on Object-Oriented Programming (ECOOP'93)*, pages 458–481. Springer, Berlin, LNCS 707, 1993.

[Deu91] O. Deux et. al. The O2 System. *Communications of the ACM*, 34(10):34–48, 1991.

[GCH93] M. Gogolla, S. Conrad, and R. Herzig. Sketching Concepts and Computational Model of TROLL *light*. In A. Miola, editor, *Proc. 3rd Int. Conf. Design and Implementation of Symbolic Computation Systems (DISCO'93)*, pages 17–32. Springer, Berlin, LNCS 722, 1993.

[GH91] M. Gogolla and U. Hohenstein. Towards a Semantic View of an Extended Entity-Relationship Model. *ACM Trans. on Database Systems*, 16(3):369–416, 1991.

[GHC+94] M. Gogolla, R. Herzig, S. Conrad, G. Denker, and N. Vlachantonis. Integrating the ER Approach in an OO Environment. In R. Elmasri, V. Kouramajian, and B. Thalheim, editors, *Proc. 12th Int. Conf. on the Entity-Relationship Approach (ER'93)*, pages 376–389. Springer, Berlin, LNCS 823, 1994.

[GMB94] S. Greenspan, J. Mylopoulos, and A. Borgida. On Formal Requirements Modeling Languages: RML Revisited. In Bruno Fadini, editor, *Proc. 16th Int. Conf. on Software Engineering (ICSE'94)*, pages 135–148. IEEE Computer Society Press, 1994.

[Gog94] M. Gogolla. *An Extended Entity Relationship Model. Fundamentals and Pragmatics.* Springer, Berlin, LNCS 767, 1994.

[JSHS91] R. Jungclaus, G. Saake, T. Hartmann, and C. Sernadas. Object-Oriented Specification of Information Systems: The TROLL Language. Informatik-Bericht 91–04, Technische Universität Braunschweig, 1991.

[KMSJ88] M. Koubarakis, J. Mylopoulos, M. Stanley, and M. Jarke. TELOS: A Knowledge Representation Language for Requirements Modelling. Technical Report CSRI-222, University of Toronto, 1988.

[LEW94] S. W. Liddle, D. W. Embley, and S. N. Woodfield. A Seamless Model for Object-Oriented Systems Development. In E. Bertino and S. Urban, editors, *Proc. Int. Symposium Object-Oriented Methodologies and Systems (ISOOMS'94)*, pages 123–141. Springer, Berlin, LNCS 858, 1994.

[LLOW91] C. Lamb, G. Landis, J. Orenstein, and D. Weinreib. The ObjectStore Database System. *Communications of the ACM*, 34(10):50–63, 1991.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall, Englewood Cliffs (NJ), 1991.

[RG97] M. Richters and M. Gogolla. A Web-based Animator for Object Specifications in a Persistent Environment. In M. Bidoit and M. Dauchet, editors, *Proc. 7th Int. Conf. Theory and Practice of Software Development (TAPSOFT'97)*, pages 867–870. Springer, Berlin, LNCS 1214, 1997.

[SSE87] A. Sernadas, C. Sernadas, and H.-D. Ehrich. Object-Oriented Specification of Databases: An Algebraic Approach. In P.M. Stocker and W. Kent, editors, *Proc. 13th Int. Conf. on Very Large Data Bases (VLDB'87)*, pages 107–116. Morgan-Kaufmann, Palo Alto, 1987.

[Wie91] R.J. Wieringa. Equational Specification of Dynamic Objects. In R.A. Meersman, W. Kent, and S. Khosla, editors, *Object-Oriented Databases: Analysis, Design & Construction (DS-4), Proc. IFIP WG2.6 Working Conference, Windermere (UK) 1990*, pages 415–438. North-Holland, Amsterdam, 1991.