

Managing cognitive aspects of conceptual modeling and design in the arrow diagram logic framework: A mathematician's view¹

Conceptual Modeling and Design (CMD) aims at specifying a universe of discourse in abstract terms suitable for computer processing which will be simultaneously transparent for a human. Thus the problem necessarily contains aspects of both cognitive and formal cognitive and formal logic.

A key point in the CMD-problems is that the universe to be specified is not an example from a textbook but a piece of reality like a bank or an airport. In addition, the specification has to be somehow extracted from multiple, very informal and often hardly consistent descriptions provided by different people. Moreover, as the process of information system design is going on, and mutual understanding between the contributors is increasing, these partial descriptions may be changed so that the entire specification is permanently revised and corrected. On a whole, the design process goes along something like a spiral rather than a direct line, and thus the conceptual model is endlessly reengineered until the approaching deadline captures it as it is.

So, a conceptual model has to compress a bulky array of diverse information into a compact comprehensible specification suitable for effective communication between the database designers, experts on the universe and programmers. A natural, and the only practical, choice is to use graphical languages, and indeed, a vast variety of graphical notational systems were developed in the CMD-stream. On the other hand, a conceptual model should be amenable to formalization in order to be readily transformed into precise software specifications in the further stages of design. Thus, a good CM-language should be graphical and comprehensible, formal or ready to be formalized, and sufficiently expressive to capture all the peculiarities of the real world.

Up to the current state of the art in CMD, this synthesis has not been achieved: graphical specification languages in use are either semi-formal (or, worse, quasi-formal) or have a very restricted expressive power, or both. As one of the CMD-experts wrote, "...what exists out in the field today is cookbook approaches ... poorly formed and having too many subjective human considerations" ([2]). Indeed, though a few practically helpful CMD-methods were proposed and heuristically developed, no logically consistent framework underlying the entire field was built. Languages currently employed in CMD are based on very different and particular (if any) logical foundations so that the entire area appears today as somewhat like a Babylon of graphical notational systems. This Babylon has its Tower too - it is the so-called Unified Modeling Language (UML) recently adopted by a consortium of software vendors as an industrial standard ([3]).

Perhaps the Babylon metaphor is not adequate enough to capture the special charm of the CMD-world. During my seven-year wandering in its notational chaos, I encountered terrific creations and studied witty tricks invented to tame them. I met problem-monsters that turned out to be phantoms, disappearing with a single touch of the proper formulation. I observed how subtle the human imagination could be when one stands face-to-face with the severe problems of structuring a reality. And I had the recurrent living nightmare of the UML-Tower collapsing and burying a blooming garden of software design.

¹ I wrote this text in the mid 90s for some project in-between cognitive and computer science.

Of course, this impressionistic picture is merely a mathematician's reminiscence of a highly uncomfortable for him intellectual framework. In reality, the CMD-world is very well governed by engineering and marketing rather than mathematical laws. And despite the logical gaps, the UML-Tower stands unshakable without any sign of collapse. The evident truth is that the cognition structures the human operates are richer than those captured by formal logic; their stability and effectiveness are provided by complex mechanisms beyond logic.

Nevertheless, what I have discovered during my studies of conceptual modeling is that very often, for a given heuristically developed notation, some underlying graph-based logic with precise semantic meaning can be indeed explicated (see [4] for a few examples). Moreover, each time I was successful in this task, the logic in question turned out to be essentially the same logic of viewing the world as a collection of sets and functions varying in time. The latter construct is well known in category theory under the name of topos of variable sets. Thus, the experience of conceptual modeling can be refined as an experience of representing the universe of discourse by a topos-theoretic specification. In other words, the arrow diagram logic appears to be intrinsic for the cognitive aspects of CMD or at any rate very appropriate for explicating them in precise terms.

So an exciting field of interdisciplinary research combining diagram logic, semantics of data and computation, as well as cognitive psychology, is emerging. These disciplines should be integrated in pursuing essentially cognitive problems of structuring reality and presenting it by a comprehensible graphical specification. Applied in this field, the arrow diagram formalism of category theory becomes a special way of thinking rather than just a technical apparatus. It has been proved to be both instrumental in managing concrete problems (see, e.g., [5,6,7]) and appropriate for enabling one to think them out towards far reaching conceptual generalizations (e.g., [8,9]).

Below I will try to present some milestones in the field (which I am hopeful will become cornerstones in the future smart building of CMD). In part, they present what is already done and, in part, they point out problems and directions for future research. Some of the tasks are quite concrete, and I clearly see ways of developing them in detail. These are the problems in sections 1.1, 2.1, 2.2 and, to some extent, in 3. Others are only general pointers and suggestions (sections 1.2, 1.3 and 4). Section PROPAGANDA is somewhat unusual but I consider publicity issues described there an important component of the entire program.

1 Formal semantics for object-oriented (OO) and entity-relationship (ER) data modeling

In spite of many statements one may find in the literature that formal semantics for the data modeling part of the OO-specification paradigm is built or, at least, a way of building it is clearly seen (see, e.g., [10]), this is not the case. The issue is more involved than is usually thought, and only in [9] is a formal semantics for the core notion of object identity proposed. This semantics at once makes the space of conceptual modeling multi-dimensional: usual, purely structural, considerations should be completed with logical-time-dimensional considerations in order to capture the diversity of kinds of real world objects properly. Only in this logical-time-enriched set-

up can the distinction between entities and relationships, crucial for conceptual modeling, be explained.

This approach immediately gives rise to the following research tasks.

1.1 GENERALIZED PREDICATE LOGIC. The construct of variable set (varset) is well known in category theory. However, its new OO-interpretation opens the door for the essential generalization and development of the varset semantics and its logical calculus counterpart.

The key novelty is that interconnection between two states of varset is given by a general binary relation rather than by a function between them. It is remarkable that recently a step in the same direction but induced by pure mathematical reasons was made by categorical logicians: the paper [11] states direct links between the relational varset semantics and some generalization of predicate logic (PL) which the authors of [11] called basic PL. However, though varset semantics proposed in [11] is more general than the standard one it is still a special case of that built in [9]. The latter appears to be the most general notion of varset; correspondingly, its logical counterpart appears to be the most general notion of PL, thus,

ordinary PL < basic PL < general PL.

Studying this general PL from proof-theoretical, model-theoretical and algebraic viewpoints (categorically, the algebraic version has to be some very general notion of hyperdoctrine) should be interesting logically and promising for applications. Indeed, while the classical relational database theory can be well developed within the ordinary PL logic framework, consideration in [9] shows that OO database theory needs the framework of general PL.

1.2 CONSTRUCTIVE ONTOLOGY AND MARTIN-LÖF TYPE THEORY (Conjecture). MLTT presents a type-theoretic approach to foundations of constructivism. On the other hand, formal semantics of object identity built in [9] explicates the very basic notion of object via its external identification (observation), that is, constructively in the strongest sense of the term. Moreover, it was shown in [9] that stating conceptual modeling on these constrictive foundations leads to a special constructive ontology of real world objects. It seems that the notion of observable object identity could be incorporated into MLTT making the type-theoretic framework more powerful and flexible.

Is it interesting from the application viewpoint? MLTT has a wide range of applications in computer science, the most close to real practical using are those in the field of functional programming (automated program verification etc). The connections mentioned above can give rise to new areas of MLTT- applications: database theory and, more hypothetically, OO-programming.

1.3 FORMALIZATION OF COGNITIVE PATTERNS (Conjecture). In the varset semantics frameworks of [9], objects of the real world have different ontological statuses in function of the way they are identified, in particular, via other objects. A taxonomic space (something like the Periodic table) for identification relations between objects was built in [9], where well known IsA, IsPartOf, various Association and Qualification relationships take their own place. It looks extremely interesting to relate structural patterns considered in cognitive science with the taxonomy of object identification mentioned above. In particular, as soon as the intuition behind important

cognitive constructs like, e.g., IsA- or IsPartOf- relationships is formally explicated, cognitive problems of structuring reality should become approachable with precise mathematical means.

2 Graph-based logic of entity-relationship-diagrams

Since its invention in the nineteen-seventies, the entity-relationship (ER) approach to conceptual modeling and design has become a kind of de facto standard in the area ([1]). More than 20 years of vitality of software methodology is a remarkable fact which shows that ER was much more than a fashion (to which the software world is so inclined) and suggests that there are certain deep causes of this long success.

To my mind, one of them is that the ER-ontological perspective corresponds well to the cognitive patterns of a human arranging real world data. Moreover, in the varset framework of [9], the ontological diversity of the real world is captured by a certain logical closure of the ER-ontology (to wit: the diversity is packed into a taxonomic parallelepiped where the segment ER forms a big diagonal).

The second principal cause is associated with the graphical notation embodied into the ER-methodology, the famous ER-diagrams. They turned out to be an extremely effective means of conceptual design, and very convenient for communication between the database designer, the user and the programmer. A lot of graphical notational dialects were created on the basis founded by ER-diagrams, in one or another form they appear in any modern CASE-tool. However, any notational system is a more or less direct reflection of the corresponding logic, and I'd assert that the true value of the invention of ER-diagrams is the invention of a special kind of "graphical logic," the ERD-logic.

It was shown in [4,9] that as soon as one wishes to explicate the ERD-logic in some direct way, one will necessarily come to a certain generalization of the sketch logic developed in category theory; mathematical presentation of the machinery of generalized sketches is displayed in [12]. It is remarkable that essentially the same approach was developed by Makkai [13] out of purely mathematical reasons. Here we again can observe a nice correspondence between categorical logic and logical machinery needed for software engineering.

The success of explaining and developing the ER-diagrams technique via generalized sketches gives rise to the following research tasks.

2.1 GENERAL GRAPH-BASED LOGIC AND ALGEBRA. In [12], the algebraic framework of generalized sketches is displayed for the simplest case when diagram operations are independent. In applications, however, operations are ordered so that the shape of one operation may well involve markers of some preceding operations. In the case of string-based logic and algebra, this phenomenon is well studied under the name of essentially algebraic theories (invented by Peter Freyd). For the graph-based logic situation, the issue needs further elaboration though I don't anticipate any principal problems. In a wider context, the standard universal algebraic material well known for the string-based situation needs to be reproduced in the diagram algebra framework.

REMARK. Of course, there are well known categorical approaches to the problem (see survey [14]). However, from the application viewpoint, they are overly internalized. In particular, signatures are assumed to be abstract objects of some abstract category. In contrast, applications need a simpler and concrete framework where signatures are sets consisting of diagram markers.

2.2 SPECIFICATION VERSUS VISUALIZATION. Comprehensibility is a crucial aspect of a logical specification from the viewpoint of conceptual modeling. To manage it properly, one should carefully distinguish between logical specification as such and its visualization, that is, the form in which it appears to the reader. The issue in question can be also formulated in cognitive psychology terms as the distinction between looking and seeing.

The sketch framework for conceptual modeling does allow one to separate visualization from specification. Moreover, with sketches the informal notion of visualization appears in a quite precise form as visualization of three items from which sketches are built: visualizations of nodes, arrows and diagram markers. In this way a formal model of visualization mechanism can be built, which allows studying and developing comprehensibility aspects of the problem within a precise mathematical framework. For example, it allows formulating the commutativity principle: visualization of conjunction of specification constructs should be conjunction of their visualizations. More generally: given a graphical specification language, both pools, that of specification constructs and that of visualization constructs, are to be organized in similar mathematical structures. Then, a good visualization mechanism should provide that a correct visual specification appears as a homomorphism from the underlying logical specification into the visualization pool structure.

Building a mathematical model along the lines above, and applying it to never trivial problems of designing a good notation, would be an interesting and extremely useful interdisciplinary research task.

3 Semantics of computation

It seems that computation modeling and specification methods live in different worlds. One of the most successful attempts to bridge the gap is the evolving algebra (EA) project (more recently also known as the abstract state machine (ASM) project [15]).

A basic assumption underlying the project is that any state of computing system can be well specified - on any level of abstraction -- as an algebra. This view is indeed fruitful in the ADT-framework for functional programming but OO-programming needs another approach. Namely, a natural way of specifying object classes goes via coalgebraic rather than algebraic specifications (see, e.g. [9]). Moreover, a peculiarity of OO-languages in practical use is that normally ADT-domains and OO-classes coexist in data specifications. So, actually one should deal with mixed algebraic-coalgebraic description. In addition, positions of algebraic and coalgebraic parts are not symmetric: objects are described by values, not the reverse. Thus, managing data modeling part of computation modeling in an abstract way as it is intended in the EA-project should be more involved and more universal than it is taken in EA.

What I'd like to suggest and to try in this respect is to adopt the generalized sketch machinery. Indeed, with sketches one has a surprising possibility to say what the data are in general, and on any level of abstraction. It seems that the framework of modeling computation via EA could be readily reformulated in the sketch framework for data modeling.

Moreover, one can consider even more general setting when data specifications are abstract entities to be considered as objects of some abstract category. Of course, this category should be sufficiently rich to provide necessary operations with specifications. Considered as motivating example, generalized sketches suggest taking this category to be a finite presheaf topos. Modeling computation in this framework along the EA-project lines should necessarily lead to the construct of a really *abstract* state machine and a really *abstract* model of computation. Research in this direction looks both extremely interesting mathematically and extremely useful for software applications (compare with EA-project's success in building formal semantics for industrial PROLOG: with abstract state machine as above one hopefully will be able to build formal semantics for industrial C++).

4 Topos-theoretic patterns in natural language structures (conjecture)

As it was said, conceptual modeling can be seen as an art of representing a universe of discourse by a topos-theoretic specification; the fruitfulness of such a view was demonstrated in [4,9]. If we take a natural language description of the universe as its substitute, the statement above can be reformulated in the linguistics terms: conceptual modeling is translation of a natural language description into a topos-theoretic specification. This leads to a conjecture that structuring natural language constructions along topos lines should be helpful, at least in the software engineering issues. So, it might be a promising task to explicate the topos-theoretic patterns in the structures of natural language, and apply them for conceptual modeling, natural language processing and similar problems.

To summarize, the topos-oriented viewing the world fits in with some intrinsic cognitive patterns of the human mind. So, it is worthwhile to examine the possibilities and benefits of incorporating the topos specification framework into the cognitive studies.

5 Propaganda

Problems of conceptual modeling and design are certainly engineering problems and, as with any other engineering stuff, it would be a rough mistake to see them in purely mathematical perspective, or overestimate the effect of mathematical treatment. Nevertheless, the logical component of conceptual modeling is so essential, and the relevance of the corresponding arrow machinery is so high, that stating the area on proper arrow logic foundations promises serious benefits. Unfortunately, normally an engineer is rather skeptic about usefulness of any advanced mathematical treatment of his/her problems, and software engineers are not an exception. Moreover, after the great success of the relational databases and logic programming, which basically relied on solid theoretic foundations, the software world has gone far ahead while no corresponding theoretical framework was developed. This has resulted in the almost common

disappointment of practitioners in theoretical ideas, including mathematical logic and its language. (As one of the referees of our manifesto [16] with Boris Kadish wrote, "none of the theorems (beyond the really basic stuff) is at all helpful").

Of course, not mathematics at all is unsuitable, but unsuitable mathematics is not suitable. My seven year studies of the software world and practical experience of working at a software company have convinced me of the extremely high relevance of the diagram logic framework for a host of problems in software, theoretical as well as practical. So, the problem is to explain and convince the software engineering community on the benefits of the approach. On the other hand, I believe that incorporating the arrow diagram logic into the practice of software engineering is extremely fruitful for development of the diagram machinery as such.

Thus, writing declarative texts like "The arrow manifesto" [16] or postings [7] seems to be an important issue. Of course, it is far beyond mathematics as such but is necessary for keeping mathematics lively.

References

(My papers can be found in the ftp-directory <ftp://ftp.cs.chalmers.se/pub/users/diskin/>. Their relative addresses are pointed in angle brackets).

- [1] C. Batini, S. Ceri and S. Navathe, Conceptual database design: an Entity Relationship Approach, Benjamin Cummings, 1992
- [2] S. Navathe, The next ten years of modeling, methodologies and tools. In: Entity Relationship modeling, Springer LNCS'645, 1992
- [3] The Unified Modeling Language (UML), <http://www.rational.com/uml/>
- [4] Z. Diskin, The graph-based logic of ER-diagrams and taming heterogeneity of semantic data models. Technical Report 97-04, Frame Inform Systems/LDBD, 1997 <REPORTS/tr9704.ps>
- [5] M. Johnson and C.N.G. Dampney, On the value of commutative diagrams in information modeling. In: Algebraic Methodology and Software Technology, AMAST'93, Springer, 1993
- [6] B. Cadish and Z. Diskin, Heterogeneous view integration via sketches and equations, In: Foundations of Intelligent Systems, Springer LNAI'1079, 1996
- [7] Applications for category theory. Discussion in the "Categories" e-mailing list, summer 1997 <MANIFEST/discuss2.*>
- [8] B. Jacobs, Objects and classes, coalgebraically. Report CS-R9536, CWI, Amsterdam, June 1995
- [9] Z. Diskin and B. Kadish, Variable set semantics for generalized sketches: Why ER is more object-oriented than OO. To appear in Data and Knowledge Engineering (for manuscript see <ER/ERvsOO.ps>)
- [10] G. Vossen, On formal models for object-oriented databases. OOPS Messenger,6(3):1-19,1995.
- [11] S. Ghilardi and G. Meloni, Relational and partial variable sets and basic predicate logic. J.Symbolic Logic, 61:843-872, 1996
- [12] Z. Diskin, Generalized sketches as an algebraic graph-based framework for

- semantic modeling and database design. Technical Report 97-03, Frame Inform Systems/LDBD, 1997 <REPORTS/tr9703.ps>
- [13] M. Makkai, Generalized sketches as a framework for completeness theorems. Technical report, McGill University, 1994.
- [14] E. Robinson, Variations on algebra: monadicity and generalizations of equational theories. Technical report, University of Sussex, 1994
- [15] Yu. Gurevich, Evolving algebras: Lipari Guide. In Specification and Validation Methods, Ed. E. Börger, Oxford University Press, 1994.
- [16] Z. Diskin and B. Kadish, The Arrow Manifesto: Towards software engineering based on rigorous yet comprehensible specifications, 1997 <MANIFEST/arrfest.ps>