On Modeling, Mathematics, Category Theory and RM-ODP

In *WOODPECKER'2001: Workshop On Open Distributed Processing: Enterprise, Computation, Knowledge, Engineeiring and Realization*,
Proc. 1st Int. Workshop in conjunction with ICEIS'2001, Setubal, Portugal, July 2001.
Eds. Jose Cordeiro and Haim Kilov, ICEIS Press, Portugal, 2001, pp.38-54.
ISBN 972-98050-5-9

Zinovy Diskin*

Lab for Database Design,*Frame Inform Systems*,Ltd., Riga, Latvia
zdiskin@acm.org

Abstract. RM-ODP is about modeling, and mathematics is one of the oldest, and deserved, modeling disciplines. Experience accumulated in mathematics in general, and some of machineries developed, may well turn out to be relevant for arranging/organizing/ formalizing the concepts managed in RM-ODP. Especially promising here is mathematical category theory that is nothing but a discipline and framework for structure engineering considered in an abstract and precise way.

1 Introduction

This paper is a mathematician's reflection on general formal foundations for RM-ODP [1] and a suitable mathematical framework to set them. Correlations between these subject matters are not accidental: RM-ODP is about modeling, and mathematics is one of the oldest, and deserved, modeling disciplines. Experience accumulated in mathematics in general, and some of machineries developed, may well turn out to be relevant for arranging/ organizing/formalizing the concepts managed in RM-ODP. In section 2 of the paper, a very general schema of modeling is presented, and some suggestions on arranging and interpreting some material considered in RM-ODP are made (sect. 2.3). In particular, its reformulation in terms of mathematical category theory (CT) is suggested as very beneficial.

Indeed, RM-ODP is aimed at specifying structural patterns of extremely polymorphic nature, being as abstract and uniform as possible, thus setting a kind of foundation for structure engineering relevant to ODP. A similar general intention underlies CT which can be considered as a discipline and framework for mathematical structure engineering treated in an abstract and precise way. So, the general CT-methodology of specifying is indeed in good match with some general spirit one can find in RM-ODP. It follows then that the concepts developed in CT form just that mathematical framework RM-ODP needs.

Sections 3, 4 and Appendix A present some elaboration of what was just said. The goal is to state and justify high relevance of CT for arranging and precise formulating specifications managed in software engineering (SE) in a wide sense including business/enterprise modeling and knowledge representation. The main observation is that often a software engineer deals with a stuff in much similar to that managed in mathematics: design (definition), presentation and reasoning about structures modeling a piece of reality (material, informational, computerese) in an abstract way. Hence, to look for a mathematical framework suitable for applications in the ODP-field,

* Supported by Grants 93.315 and 96.0316 from the Latvia Council of Science

one should look at meta-mathematics – a part of mathematics aimed at modeling mathematics by mathematical means.

A few approaches and frameworks were developed in meta-mathematics. One of them is a popular style of formalizing logical systems like in a widely known formalization of first order logic originated from Tarski and now presented in any textbook on mathematical logic. This framework is quite familiar to computer scientists, in particularly, it has strongly influenced some parts of RM-ODP. Another – more recent – meta-mathematical framework is CT where emphasis is made on relations, manipulations and transformations of/between mathematical structures; as it was said, CT is a discipline and framework for mathematical structure engineering. The main thesis here is that Tarskian meta-mathematics (TarMM) is, generally speaking, not relevant for applications in questions as a basic specificational framework while CT offers just those methodology and apparatus which are really needed. A brief outline of this match between CT and SE is presented in section 4.

Acknowledgement. First of all, I'm indebted to Haim Kilov with whom the ideas presented in the paper have been repeatedly discussed in general and in different particular contexts. He encouraged me to shape some draft of these ideas into a text you see below, carefully read and commented it and traces of Kilov's remarks can be found in many places of the paper (but of course the responsibility for possible misinterpreting of RM-ODP's statements is totally mine). Thanks also go to all the participants of the discussion [12], especially stimulating for the present author were postings by Bill Lawvere and Steve Vickers. Special thanks go to Mike Healy with whom we had a few fruitful talks on the theme; after all, a real software project with essential use of CT that Mike has conducted at the Boeing is of special value for any theorist writing on the subject. Finally, as probably everyone writing on CT-applications in SE, I'm indebted to Joseph Goguen for his insights into the field.

2 General abstract schema of modeling and RM-ODP

2.1 General schema of modeling (something by something). Mathematical modeling

A quite general schema of modeling relevant to our context is shown on Fig. 1 where

– the node *AppDom*(ain) denotes some application domain and *UoD* is a particular universe of discourse (within *AppDom*) to be modelled.
– *ModDom*(ain) is another domain based on concepts either more abstract or/and more precise or/and more constructive than those of *AppDom*. The endo-arrow $Der\colon\ ModDom \longrightarrow ModDom$ denotes internal manipulations with *ModDom*-concepts resulted in a new derived knowledge. Because of special nature of the *ModDom*, these manipulations often appear as some kind of reasoning, more or less precise, but of course, not necessarily formal.
– The arrow $Mod\colon\ AppDom \longrightarrow ModDom$ denotes ways of encoding of *AppDom*-concepts in terms of *ModDom* and the arrow *App* denotes decoding/reverse interpretations of *ModDom*-statements.
– The arrow *mod* is a kind of specialization/ restriction of *Mod* to *UoD* and *Model* is its range – the model as such. Figuring out this specialization and its actual domain is not automatic, just the opposite: in each particular case the modeler

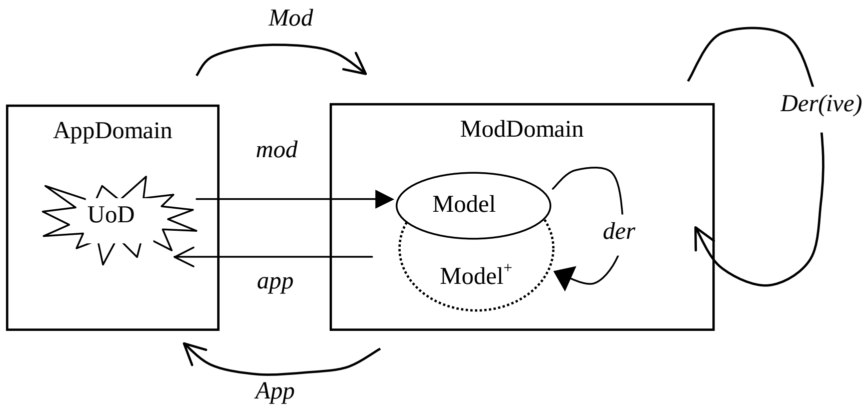


Fig. 1. General Schema of Modeling (something on the left by something on the right)

has to decide which pieces of *UoD* should be presented, and in what way, in the model. Then, because of special properties of *ModDom*, *Model* is something more observable and manageable than *UoD*.

– The arrow *der* is a specialization/restriction of *Der* on *Model* and $Model^+$ is its range, that is, the closure of *Model* under *der*-manipulations. The usefulness of modeling is just in this augmentation of *Model* up to $Model^+$.
– Finally, the arrow $app\colon\ Model^+ \longrightarrow AppDom$ is an inverse interpretation of results of reasoning about the model in the *AppDom*-terms and applying them to *UoD*.

If at the target of the arrow composition

$$mod; der; app\colon\ UoD \longrightarrow UoD,$$

we will get something useful for *UoD*, the entire modeling action may be considered successful. If for a majority of typical *UoD*'s inside *AppDom* such successful triples (mod, der, app) can be found with the same (*ModDom*, *Der*), the latter (together with *Mod* and *App*) can be considered a suitable modeling framework for *AppDom*.

Mathematical modeling in a wide sense is a modeling where *ModDom* includes/heavily uses mathematical structures. Manipulations with mathematical *Model*s are, of course, motivated by connections with *UoD* but should be (the ancient Greeks' prescription) separated from them so that one could analyze correctness of manipulations looking only at their structure. If that is indeed the case and on the right we have some formal structure formulated in entirely abstract terms and our manipulations are based solely on explicitly formulated formal derivation rules, then we have formal mathematical modeling (of course, connections with *UoD* still important but now their role is reduced to motivation/inspiration while the very reasoning as such is free of *AppDom*-arguments). However, the ideal above is not too often achieved in applications and mathematical modeling ranges from formal through formalizable to semi-formal (see appendix B.1). As a rule, in science and engineering only semi-formal mathematical modeling is used (and very successfully! But one should carefully distinguish it from quasi-formal pseudo-mathematical modeling that can be often found in the literature).

2.2 Examples.

The schema above is polymorphic and realized in different contexts. In addition, the same domain may appear as *ModDom* in one context/discipline and as *AppDom* in another. Below is a (hopefully instructive for our main theme) list of examples of contexts/disciplines of modeling phrased in terms of the schema. An example (I) in the list is specified by a brief description of what is *AppDom* (the left part of General Schema for (I)) and what are *ModDom* and *Der* pertaining to it (the right part of Schema for (I)).

1. *Physics* (as a typical natural science). On the left we have the real world, on the right is a mixture of mathematical structures (groups, vector spaces, differential/integral calculi etc) formulated and managed with different degrees of abstractness and formality. A typical reasoning here is a mixture of formal derivation steps and steps (explicitly or implicitly) justified by reference to the left, that is, involving intended physical interpretation.
2. *Mechanical/Electrical Engineering.* Similar to 1 above but with *AppDom* set by machines working according to the laws of nature and manufactured and operated by humans. So, *AppDom* lies in some intersection of the natural and social worlds. The presence of the later leads to important aspects of *AppDom* that is hard to capture and understand within mathematical models. Moreover, social aspects influence the very purposes of modeling, for example, more precise and/or more explicit and manageable models may not always be the goal due to some political reasons (Joseph Goguen's observation).
3. *Software Engineering, SE* (in a wide sense including essential fragments of AI, knowledge representation, linguistic engineering...). Similar to 2 with *AppDom* set by computers working according to formal logical instructions and manufactured and operated by humans. So, *AppDom* again lies in some intersection of the logical and social worlds (and the remark above applies here as well, and is even more actual). What should we have on the right for *ModDom* is just the subject of discussion in section 3 and 4. The thesis is that taking CT for *ModDom* and diagram chasing for *Der* would provide a very promising modeling framework for SE. (Diagram chasing is a major machinery in CT, it is nothing but a graph-based analog of term rewriting).
4. *("Classical") Mathematics (a la Bourbaki).* On the left are general intuitions of space and time, continuum, geometric figures and counting, ... On the right are mathematical structures in the sense of Bourbaki. Reasoning about these structures is performed in some strict yet informal classical logic (IntLog) and essentially depends on another intuitive component usually referred to as intuitive (naive) set theory (IntSet).
5. *Metamathematics-I* (Foundations). The domain *AppDom* is IntSet, the naive set theory in the base of mathematics. *ModDom* consists of special mathematical structures studied in so called formal set theories (in classical mathematics) or the topos theory in CT (Appendix A.1). In their early days, foundations required a specially refined – constructive – way of reasoning but it seems that nowadays this requirement is a possible option to be studied but not a must.
6. *Metamathematics-II* (mathematical logic and model theory). The domain *AppDom* consists of the ways of setting mathematical structures and reasoning about them in IntLog. On the right are again special mathematical structures which

have been studied in mathematical logic and model theory (see Appendix A.2). The same remark on constructive reasoning above is also applied here.

7. *Metamathematics-III* (abstract mathematical structures engineering). On the left we have *ModDom* of 4, that is, *AppDom* consists of mathematical structures and operations with them. In *ModDom* we have categorical structures and manipulations with them realized mainly via diagram chasing (see Appendix A.3).

2.3 RM-ODP in the general modeling schema framework.

The general schema on Fig.1 provides some framework to discuss the general structure of RM-ODP, its interpretation and possible formalization. Roughly speaking, what is called 'interpretation concepts' (clause 6 of Part 2) is related to *AppDom* and the mapping *Mod*, and 'modeling concepts' (clause 8) together with specification and structuring concepts (clauses 9 -13) are related to *ModDom* and the mapping *App*. Some remarks can be made even in this quite general arrangement.

1. An important idea the schema suggests is that though the nature of domains *AppDom* and *ModDom* may be quite different, the very appearance of pieces *UoD* and *Model* in the context of modeling shows a certain similarity of structures involved and explicated in them. In other words, 'objects' *UoD* and *Model* carry similar structures and arrows *mod* and *app* appear as structure preserving mappings. It seems that this idea of structural similarity is just in the RM-ODP's spirit. However, the framework provided by RM-ODP shows some deficiency in this aspect: the set of interpretation concepts (clause 6) is much poorer than that of modeling concepts (clause 8). In particular, it would be useful/reasonable to have the interpretation counterparts of dynamic modeling concepts, at least, those of action, behavior, state.

2. Another idea suggested by the schema is the importance of the arrow *Der* over *ModDom*. In a sense, the essence of modeling is just in manipulations with models providing new knowledge. On a more technical level, this amounts to operations which are allowed to perform over models and with their elements. It seems that this aspect of modeling is not reflected in RM-ODP.

3. More technical remark. RM-ODP is aimed at establishing a really abstract modeling framework independent on specific methodologies, languages and, the more so, tools. The only mathematical framework where such systems of concepts as (a) [syntax, semantics, interpretation] or, say, (b) [specification, instance, implementation] can be defined in a really abstract yet technically substantial way is CT. In particular, for the system (a) above it is done in categorical logic, and an outline how the system (b) could be managed was presented in [3]. For another example, General Schema on Fig.1 could be also formalized in categorical terms. The latter would be hardly more than just an exercise in formalization but one point of the categorical view on the subject might be useful: besides entities, the set of basic interpretation concepts (clause 6) should include some kind of 'inter-entity' concept: interaction, or transformation, or correlation, or morphism between entities.

4. Another technical remark. The part of specification concept set dealing with types, classes and templates (clauses 9.7 - 9.21) would benefit from an abstract categorical reformulation of types and templates as specifications and classes as their extents. A fragment of such a reformulation was presented in [3].

Also, with CT, linguistic concepts (clause 7) can be presented in a formal way suitable for string-based and graph-based (and, in fact, "any-style-based") syntactical frameworks.

3 Mathematical aspects of software engineering

3.1 SE as mathematics.

As it was stated in introduction, an important observation on the stuff a software engineer manages is its great similarity to the stuff managed in mathematics. A general partial evidence for this can be seen in the fact that SE is full of names for concepts, names for systems of concepts and names for systems of names etc., all put in more or less organized syntactical formats (often without precise semantics but this is another story). To get this overall impression it's enough to look through a few industrial standards trying to regulate the use of all that stuff, in particular, to RM-ODP.

Another similarity of general nature is that in both mathematics and SE, understanding/managing some construct/artifact/"phenomenon" often amounts to explicating a structure underlying the construct. The latter is then viewed as something composed/organized rather than holistic. (Well, it's common for scientific understanding in general but the structural component as such is especially important in mathematics and SE).

Specific partial evidences may be found in specific issues/texts. For just one example, in a recent book "Business specifications" [13], the art and technology of business modeling are treated in a way that makes them very similar to (the art and technology of) mathematical modeling.

3.2 Peculiarities of SE's mathematics: length and width vs. depth.

However, the pieces of reality modeled in classical mathematics and in SE are qualitatively different, correspondingly the structures designed are different too. Roughly, theories in classical mathematics are "short and narrow yet deep" while SE structures are multi-level with each level "long and wide yet flat". More accurately, this metaphor means the following.

"Length" : typical first-order mathematical structures are "short" in that they have (i) only few sorts, (ii) only few operations and few relations of (iii) short arities not more than, say, 4 or 5. In contrast, structures in SE have dozens of sorts over which dozens of operations and relations of dozen-cardinality arities are defined.

"Width and depth": typical mathematical structures are "narrow and deep" in that they are defined by a small set of axioms from which highly non-trivial deep consequences (theorems) are derived. In fact, definitions are justified only by theorems that can be derived from them. So, classical mathematical theories are like bore-pits and just this property makes them effective in classical domains' applications (physics etc, see Appendix B.2).

In contrast, structures considered in SE are subjected to dozens of constraints (business rules) but usually we are not interested in deriving deep theorems from them, quite immediate or next to them consequences would be sufficient. Definitions (specifications) are justified by better understanding and presentation of complexly structured subject matters, particularly, an observable, comprehensible and explicit presentation makes communication between experts much easier.

3.3 SE's mechanisms to manage complex structures.

So, in SE the vertical structure of theories/specifications is quite simple but the very definition of structures (horizontal section) in a manageable and comprehensible way is not trivial and is really a problem. Use of abstraction, modularization and comprehensible presentation mechanisms becomes a must, and a few were invented. For example, among the most important and effective in SE are:

– *object orientation*, a major concept of modern software engineering;
– *set orientation*, underlying the modern OO analysis and design and, independently of OO, implemented in relational databases via SQL;
– *graphic notation* for many specification languages, eg, the ER-diagrams and the UML, and for a million of this-tool-vendor languages;
– *multi-level specification architectures* in many versions.

Another mechanism, of different nature yet also very important, is elegance. It's difficult to formalize what it is but what is for sure that a SE-expert has a well-shaped working notion of elegant structure/specification (basically coinciding with those of other experts!) and heavily relies on the elegance test: a proper structure/specification is necessary elegant. E.W.Dijkstra stated that in computing (software) elegance is not a disposable luxury but a matter of life and death[1].

4 CT in a SE-view: Object-oriented, setwise and graph-based mathematics with multi-level architecture

So, it follows from the above that mathematics for SE is meta-mathematics but because of peculiarities of SE's structures we should look for a special metamathematics, or elaborate it from scratch, or develop something that already exists to adjust it to SE. Particularly, we should look for those metamathematical frameworks/methodologies/techniques which make a good match with mechanisms listed in 3.3. Surprisingly (or just the opposite, quite naturally) but peculiarities of the "ridiculously abstract" CT's methodology and apparatus make a great match with SE peculiarities and mechanisms above. We will consider them in parallel with the list 3.3.

4.1 CT as OO-language: Mathematical structures via arrows.

Maybe, the key idea underlying CT is that the very definition of a mathematical structure can be done by "meta-means", that is, via morphisms (arrows) relating structures between themselves. In other words, an internal structure of a mathematical object (a mathematical structure a la Bourbaki) is set externally by specifying some structure over connections/relations of this object to other objects. Because these connections/relations are described in CT by arrows, the CT-thesis is that *a mathematical structure is a structure over arrows.* Correspondingly, that special way of thinking out and reasoning about mathematical structures that is induced by the CT-framework is often called the *arrow thinking.*

[1] I'm indebted to Haim Kilov for calling my attention to this really important mechanism, and for the reference to Dijkstra too.

Specifying a mathematical construct by arrows gives a really abstract specification applicable in almost any context: just organize the objects of interest into a category (that is, define morphisms between them). Particularly, when the context is a universe (category) of sets (and mappings), the arrow specification can be reformulated in terms of elements. It may be said that such an elementwise description is an implementation of the arrow specification.

In the general description above one can recognize a core feature of OO – the so called encapsulation of object's structure and its accessibility only via the (arrow) interface. So, arrow thinking is directly related to OO.

4.2 CT as a setwise language. Toposes.

After object encapsulation, another major property of OO is a way to specify object communities by organizing objects into classes – sets of similar objects: it is nothing but OO-realization of set orientation. But we have the same in CT where a major intended informal interpretation of abstract objects is to see them as set-like collections.

Moreover, object identity – a key property of objects about which tons of inks were dropped out in the OO-literature – is quite naturally modeled as a mapping between sets presenting different states of an object class (see [6] and also [4] for a quite brief exposition). A similar idea was developed in CT under the name of 'variable set' and thus, object classes are nothing but variable sets, varsets. Correspondingly, a universe of discourse appears as a collection of varsets over which certain relations between them and operations with them are defined.

The concept of such a universe of set-like objects carrying a certain structure of diagram predicates and operations was deeply elaborated in CT under the name of topos (see Appendix A.1 for some details). Of course, for applications we need a special version of toposes where all the operations are constructive, including a special constructive version of the powerset operation. It'd not be a too big exaggeration to say that what the community of conceptual data modeling is trying to do in theory for the last thirty years is nothing but an attempt to manage the notion of universe of sets closed under certain operations, that is, in fact, the notion of topos, built from scratch.

4.3 CT as a graphic language. Sketches.

SE has really suffered badly from the lack of precise semantics for major notations in use. Computer scientists and software engineers approach the problem in the only framework they are familiar with, that is, Tarskian metamathematics (TarMM, Appendix A.2) and specify, say, semantics of ER-diagrams in the ordinary first-order logic (FOL), or semantics of the UML diagrams in the OCL, a FOL-like language also based on $\exists, \forall$-quantifiers and logical connectives.

However, the two essential characteristics of TarMM is that it's string-based and elementwise. The first property is not a very serious problem though of course formulating semantics of a graph-based notation in a string-based language does not contribute to having a clear and comprehensible semantic picture. Much more serious is that TarMM is *elementwise* while a majority of high-level languages (eg, SQL, the ER-diagrams, the UML) are *setwise*, that is, the objects they specify are systems of sets (and implicitly mappings between them) and manipulations with sets (and implicitly mappings between them) rather than individual items/elements these sets consist of.

In contrast to TarMM, categorical logic is a quite natural modeling tool for the fragment of SE in question. Namely, major graphic specification languages developed in SE, eg, ER-diagrams and UML-diagrams and many others, can be well treated as *different visualizations of the same basic specification pattern* ([9]), namely, the pattern of sketch developed in CT for specifying mathematical structures (some very brief description of sketches is in Appendix A.2). Together with formal semantics for object identity as described above, this sketch treatment of ER-, UML-, <your favorite OO analysis and design>-diagrams provides them with precise formal semantics.[2] So, the sketch format appears to be something like a core language structure common to many graphic notations in use[3].

And even semantics for SQL, though the latter is string-based, can be easily recovered in CT-terms because SQL-expressions for relational operations are nothing but string-based interfaces for specifying essentially diagram operations (eg, joins are pull-backs and familiar conditions of schema matching are nothing but specifying graphical arities for diagram operations).

4.4 CT as a language for multi-level specification architectures.

CT provides a rich arsenal of means for structuring subject matters. A category is a collection of (abstract entities called) *objects* and (another kind of abstract entities called) *morphisms* between objects. You may consider also morphisms between categories – *functors* – and so form categories whose objects are themselves categories. There are also morphisms between functors – so called *natural transformations*, and thus you may form categories whose objects are functors (which are morphisms between categories which are morphisms between objects); and so on. Inside of this hierarchy there are also richer structural constructs. For example, given some algebraic theory (query language) over a category (of data schemas), special morphisms called *Kleisly morphisms* are important: they are nothing but *views*, a well known and important in applications notion [7]. For another example, there are special functors called fibrations that provide extremely rich structurizing facilities. An example of how this stuff can work in SE can be found in [2].

5 ODP: The challenge of integration and CT

ODP-systems are necessary heterogeneous including architectural (hardware and software) heterogeneity and semantic (that is, in types of data and behavior) heterogeneity . Handling architectural heterogeneity amounts to resolving communication problems (file transfers, remote logins etc.) and it seems that by now the basic conditions for it either have been achieved or will be achieved in the nearest future.

[2] For example, building semantics for the structural part of the UML class diagrams within the sketch framework turned out a pleasant exercise in categorical logic [5] but of course, a lot of non-trivial work has to be done in explicating formal meaning of semantic relations between classes like generalization, aggregation, composition ([6], see also [4] for a brief overview).

[3] and similar idea may be traced in general linguistics

In contrast, managing semantic heterogeneity (so called *interoperation*) is far from being solved and, moreover, as is noted in [10], *the problem itself is still at the stage of being understood*.

Clearly, the interoperation problem is a specification problem: to manage it one should be able to specify heterogeneous structures in precise and unifying general abstract terms. This is just what RM-ODP is aimed at, and this is just what CT can provide. As we have seen in section 4, there is a good match, and even parallelism, between some of SE- and CT-ideas to manage complexity of structures. In the SE-terminology, the CT-framework might be called object-oriented, setwise (set-oriented) and graph-based.[4] Moreover, in CT these features are inherent and smoothly integrated while their integration (as functionalities) in a ODP-system is still a challenge in SE.

For example, SQL is set-oriented but is neither OO nor graph-based while many OO-systems operate on individual objects and lack set-orientation. For another example, the language of ER-diagrams is set-oriented and graph-based but as is common to think is not OO.[5] For one more example, the UML is explicitly OO, set-oriented and graphical language yet its semantics is still far from being complete (and many of its fragments are still not well defined).

So, precise arrangement of the concepts above in the categorical terms would at once allow to get specification framework integrating OO and set-orientation, having a well defined graphical syntax and a precise formal semantics as well. Such an integral framework ("a dream of computer science" [8]) would really has a capacity for constructing a flexible (and mathematically justified) *system* of concepts and so provide a basis for (i) precise specifying, hence understanding, SE-artifacts and systems, (ii) effective communication of different communities of specialists between themselves and with subject matter experts and, finally, (iii) for effective implementation.

Of course, implementation has its own specific problems independent of specification but precise and conceptually consistent specifying would allow to separate concerns and solve specification and implementation problems with the corresponding means. What one may often observe now in SE is nothing but resolution of specification problems by implementation means (eg, OO-implementations of element-oriented specifications or graphic interfaces to essentially string-based specifications). Well, the construction is working and the principal goal of engineering efforts is achieved, but it works in a so hard to observe and check, and costly to design, maintain and update way, that declarations of "software crisis" and similar claims are abundant in the literature. [6]

Once one starts to specify a complex structure in CT-terms, patterns developed in CT begin to guide the process of structuring "towards a proper arrangement" (cf.

[4] A word of caution about terminological mess should be said. Amalgamation of object- and set-orientations in SE, in CT-terms would be phrased as arrow orientation (while objects play in CT just an auxiliary role of place-holders for arrow's sources and targets). On the other hand, in discussions like "cats vs. sets" , what is attributed to 'sets' is often a reference to element-orientation as opposed to set-orientation.

[5] Actually ER-diagrams do have OO-semantics but it can be revealed only in precise semantic framework for them [6], which is essentially categorical and not widely known to the community.

[6] Well, software crisis have been mentioned since the second half of 60s yet we all observe an incredible progress of this engineering discipline.

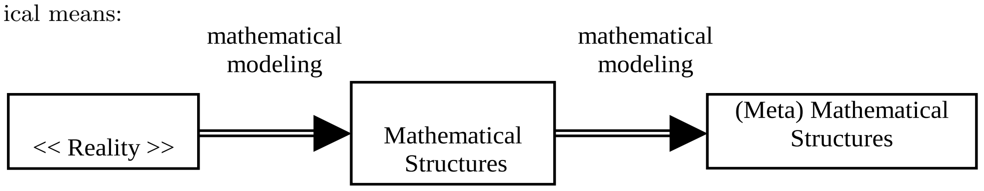
Goguen's Manifesto [11]). A great advantage of the CT as an integral mathematical discipline is that it allows you to see "what a proper arrangement is". After all, it looks like that by some mystic way the structures one can find in the computerese reality follow categorical patterns (compare with Appendix B.2). So, the relevance of CT for specifying ODP-systems is evident and exciting, and may seem surprising. But actually it is quite natural: the same goal of building a framework where complex structures (computational and specificational) could be managed in a consistent and effective way has led software engineers and mathematicians to close results.

References

1. RM-ODP Standards. http://www.dstc.com/Research/Projects/ODP/ref_model.html.
2. Z. Diskin. The arrow logic of meta-specifications: a formalized graph-based framework for structuring schema repositories. In B. Rumpe H. Kilov and I. Simmonds, editors, *Seventh OOPSLA Workshop on Behavioral Semantics of OO Business and System Specifications, OOPSLA98*, TUM-I9820, Technische Universitaet Muenchen, 1998. (ftp://cs.chalmers.se/pub/users/diskin/PAPERS-DB/oopsla98.ps).
3. Z. Diskin. Abstract metamodeling, I: How to reason about meta- and metamodeling in a formal way. In K. Baclawski, H. Kilov, A. Thalassinidis, and K. Tyson, editors, *8th OOPSLA Workshop on Behavioral Specifications, OOPSLA99.* Northeastern University, College of Computer Science, 1999.
4. Z. Diskin. On mathematical foundations for business modeling. In *TOOLS'37: 37th Int. Conference on Technology of Object-Oriented Languages and Systems (Sydney, Australia)*, pages 182–187. IEEE Computer Society Press, 2000.
5. Z. Diskin. The arrow diagram logic and the UML. Submitted, 2001.
6. Z. Diskin and B. Kadish. Variable set semantics for categorical sketches: Formal semantics for object identity and conceptual modeling. To appear in *Data and Knowledge Engineering*, manuscript is available from the authors
7. Z. Diskin and B. Kadish. A graphical yet formalized framework for specifying view systems. In *Advances in Databases and Information Systems, ADBIS'97*, 1st East-Europian Symposium, 1997.
8. Z. Diskin, B. Kadish, and F. Piessens. The Arrow Manifesto: Towards software engineering based on comprehensible yet rigorous graphical specifications. In *15th Int. Congress for Cybernetics. Namur (Belgium)*, 1998. (ftp://cs.chalmers.se/pub/users/diskin/Manifest/namur98.doc,namur98.ps).
9. Z. Diskin, B. Kadish, F. Piessens, and M. Johnson. Universal arrow foundations for visual modeling. In *Diagrams'2000: 1st Int. Conf. on the Theory and Applications of Diagrams*, volume 1889 of *Springer LNAI*, pages 345–360, 2000.
10. P. Drew, R. King, D. McLeod, M. Rusinkiewicz, and A. Silberschatz. Report on the workshop on semantic heterogeneity and interoperation in multidatabase systems. *SIGMOD Record*, 22(3):47–56, 1993.
11. J.A. Goguen. A categorical manifesto. *Mathematical structures in computer science*, 1(1):49–67, 1991.
12. M. Healy et al. On mathematics relevant to AI, SE, knowledge representation and other newly created domains. Discussion in "Categories" e-mail list (`categories@mta.ca`) run mainly under the title "Re: Mike Healy's question on mathematcis and AI", Winter 2001.
13. H. Kilov. *Business Specifications: The Key to Successful Software Engineering.* Prentice Hall PTR, 1999. ISBN: 0-13-079844-4.

A Appendix. About Metamathematics

Meta-mathematics is a part of mathematics aimed at modeling mathematics by mathematical means:



To see some structure in metamathematics, let's put some structure on mathematics itself. In a very general view, the domain of mathematics appears as a collection of mathematical structures (algebraic, relational, topological and their mixtures) defined on the base of some intuitive naive set theory, IntSet, and reasoned about in some strict yet intuitive classical logic, IntLog. In addition, the heart of mathematics (well, maybe it has a few hearts) is in manipulations with them like

– building a new structure from a structure (eg, by taking its quotient) or from a set of similar structures by some operator (e.g., Cartesian product), or
– amalgamating non-similar structures into a new type of structure (e.g., forming the structure of topological group from those of group and topological space), or
– extracting a structure from a structure of quite another kind (e.g., fundamental group of a manifold)
– ...

So, a good part of mathematics appears as a kind of *mathematical structure engineering* based on IntSet and IntLog. Correspondingly, three branches of metamathematics can be distinguished.

Metamathematics-I, often referred to as *foundations*: modeling (formalization) of set theory(ies) in which mathematical structures are (can be) built.
Metamathematics-II, often referred to as *mathematical logic and model theory*: modeling (formalization) of logic(s) in which mathematical structures are (can be) reasoned about.
Metamathematics-III, : modeling manipulations with mathematical structures, their relations and transformations in an abstract formal setting. It seems there is no a standard name for this part, we might call it *mathematical structure engineering* or *metamathematical engineering.*

Let's consider each branch in more details in the context of applications to SE and, in part, w.r.t. the opposition CT vs. ST (Set Theory) often underlying debates on mathematical foundations suitable for computer science.

A.1 Metamathematics-I: Toposes vs. formal set-theories.

The goal of metamathematics-I (so called *foundations*) is to model the domain of intuitive (naive) set theory, IntSet, by mathematical means.

Two major approaches were invented. One is a system of formal set theories, FrmST, the other is a categorical set theory, CatST. A peculiarity of FrmST is that mappings - a component of IntSet - in the formal treatment are reduced to special sets (of ordered pairs). In contrast, in CatST both ingredients remain first class citizens but the membership predicate – the basic ingredient of FrmST – becomes a derivative. This difference essentially influences the two approaches, both methodologically and technically, and actually leads to different philosophies of set universes. However, it was proven that FrmST and CatST

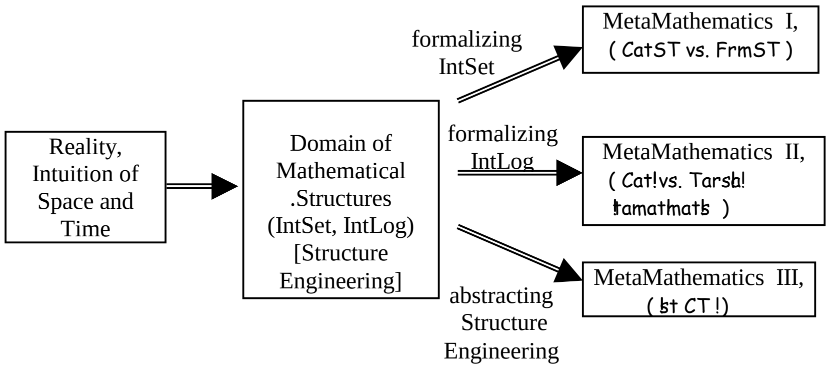

Reality, Intuition of Space and Time
Domain of Mathematical .Structures (IntSet, IntLog) [Structure Engineering]
formalizing IntSet
formalizing IntLog
abstracting Structure Engineering
MetaMathematics I, (CatST vs. FrmST)
MetaMathematics II, (Cat!vs. Tarsk! tamathmatk)
MetaMathematics III, (kt CT !)

Fig. 2. "Business schema" of mathematics

are mutually-inverse interpretable and thus equivalent (and in some precisely defined sense too).

An important achievements (maybe, the most important) of metamathematics-I is the discovery of qualitatively different universes of sets having different nature and inducing different logics of reasoning with sets so that the question of what is *the* set universe is considered incorrect. This crucial idea was recognized and elaborated in both FrmST and CatST but it seems in CatST it has much more manageable form and machinery associated with the notion of topos.

In a quite general view, a topos is a categorical structure whose intended interpretation is a universe of set-like objects and mappings between them, over which certain diagram operations are defined. Actually topos is an essentially algebraic structure if you consider the operations as operations over graphs in some precise sense. This allows to manage toposes algebraically and, in particularly, apply the effective machinery of diagram chasing – the graph-based analog of term rewriting. In contrast, set universes built in FrmSet are relational structures with rather complex axiomatics (actually, very complex in comparison with typical mathematical structures).

A key peculiarity of the categorical treatment of the notion of universe is that, in the topos framework, 'sets' are variable sets and their elements are also variable entities so that equality and other basic logical notions are considered w.r.t. some scale of logical time. This changes the very logic of reasoning about sets and their elements and makes the very framework flexible and open: one may easily develop the core structure towards one or another type of universe, regulated by one or another logic by setting the corresponding logical time scale. Such a methodology of open set universe seems to be in perfect match with RM-ODP's spirit. In contrast, universes of formal sets theories are hard to manage in this sense, they are static and closed.

A.2 Metamathematics-II: Categorical logic vs. Tarskian metamathematics.

The starting point in metamathematical-II studies is the answer to the question: "What is a mathematical structure?" In the well-known classical formalization of first-order logic (originated by Tarski and Mal'cev and known as Tarski's formalization of the notion of truth) , a mathematical structure is a set (or a family of sets indexed by sorts) together with relations and operations defined over it (them) and subjected to certain conditions (axioms, constraints). The latter are specified in a string-based notation with logical connectives and quantifiers. So, a semantic universe U is a set (family of sets) consisting of uninterpreted elements, and syntax-semantics interpretations are assignments of elements from U to variables. An essential point is that similar structures have the same sorts and the same relations and operations.

In categorical logic (CatLog), a mathematical structure is a collection of sets and mappings between them subjected to certain conditions. The later are specified in a graph-based notation with diagram predicates and operations: the carrying collection is specified by a directed graph G (whose nodes denote sets and arrows denote mappings) and constraints are specified as predicates declared for some diagrams in G (note, these predicates are for sets and mappings, not for elements). An important discovery made in CT is that it's possible to set an internal structure $\mathcal{S}$ of object X to be though of as mathematical structure via declaring a corresponding property $P_{\mathcal{S}}$ for some arrow diagram D including X (and other objects, all are connected by arrows – object morphisms). So, the predicate declaration $P_{\mathcal{S}}(D)$ entails that X can be thought of as endowed with the structure $\mathcal{S}$ and really behaves itself so.

The pair $S = (G,$ a set of predicate declarations like above$)$ is called a *sketch* (terminology goes back to Ehrismann). Thus, in CatLog, (i) mathematical structures are specified by sketches, (ii) a semantic universe U is a graph of sets and mappings and (iii) a syntax-semantics interpretation is a mapping sending G-nodes to sets in U and G-arrows to mappings in U s.t. predicate declarations are respected. Importantly that in contrast to TarMM, similar structures in CatLog may well have different number of sorts and morphisms between structures include non-trivial mappings of sorts when a basic sort of one structure is mapped into a derived sort in the other. But of course, similar structures have the same vocabularies of diagram predicates and operations.

A.3 Metamathematics-III: abstract mathematical structures engineering.

The issues to be managed here are as follows. Is it possible to consider an abstract notion of Cartesian product so that those of sets, groups and topological spaces would be just special cases of the general abstract notion? Similarly, there are operations of amalgamation/integration of sets (merely their union), of groups, of topological spaces; is it possible to find an abstract framework where some operation of integration could be specified in abstract way so that special operations above would be particular realizations (implementations) of the general format?

Similar questions arise in Metamathematics-II when we consider quite different logical systems and interpretations/translations between them. Is it possible to formulate a general notion of logical system, an abstract logic with as few specifics as possible but of course including abstract notions of syntax, semantics, interpretation? How to build logics from logics, or to include/incorporate a logic into another logic, or interrelate logics in a required way? In what mathematical framework can these problems be made manageable?

And, in general, mathematical structures don't live in isolation, their world is full of their mutual translations/interpretations/interplays (and the usefulness of mathematics is often in just these interplays). In contrast to mathematics of the ancient Greeks focused on theories intended to have only one semantic instance/realization up to isomorphism (a tradition that had been held till 19th century), modern mathematics focuses on mathematical theories/specifications that have multiple non-isomorphic semantic realizations. These theories are considered as elementary specification blocks from which complex structures are built. So, the real line that was something whole for the ancients, for a modern mathematician appears as a complex structure designed from elementary algebraic and topological blocks. The results is that large pieces of the domain of mathematical structures are themselves complex structures designed in a way somewhat similar to how large software systems are designed.

So, to summarize, manipulations with mathematical structures and their interrelations is an important component of modern mathematics, which might be called *mathematical structure engineering*, and there are serious reasons for specifying this stuff in an abstract way. Then building mathematical models for this component of mathematics is really important and we come to metamathematics-III. The only framework that makes such a modeling possible (at least, manageable) is without doubts CT (originally invented by Eilenberg and Mac Lane just for precise and manageable formalization of a particular situation of this kind - passing from topology to algebra). Thus, metamathematics-III's *ModDom* consists of categorical structures and a natural mechanism for manipulations with them is diagram chasing.

B Appendix. Some peculiarities of applying mathematics to natural sciences

B.1 Semi-formal deduction.

A typical reasoning in theoretical studies in natural sciences is a mixture of formal derivation steps and steps (explicitly or implicitly) justified by reference to *AppDom*, that is, involving intended physical interpretation. In fact, such intuition invocations in the process of derivation can be treated as entering new basic concepts and axioms into the formal system. So, in contrast to pure formal reasoning where the list of basic terms/notions and axioms which may be used is stated explicitly from the very beginning and cannot be changed later, in semi- formal reasoning one may add new ones during the derivation. A precise formal explication of these new concepts may require rebuilding of the entire formal framework and so normally they remain (though precise in some sense yet) not formalized. However, these references to *AppDom* are normally regulated by a certain discipline and, maybe, the entire process still could be somehow formalized. So, this semi-formal style of reasoning would be converted into a formal one but within another logic.

A good example here is the non-standard analysis of Abraham Robinson: for many years mechanical and electrical engineers manipulated symbols dy/dx like fractions while mathematicians considered that as just an informal syntactic sugar over strict delta-epsilon language until these fractions became quite legitimate and formal in the non-standard analysis. Similar examples are Heaviside's operational calculus and Dirac's delta-function, which were invented and used in a quite semi-formal way, and only later were precisely formalized within modern functional analysis; these lessons are instructive.

B.2 "Unreasonable effectiveness".

The combination of a short list of axioms (about short relations and operations) and deep theorems deduced from them pertaining to classical mathematical theories turned out extremely useful for applications: small width and length make checking applicability of a theorem an easy task while real deepness provides getting non-trivial and precise applied results far from being evident and far from being precise without mathematics (especially if we deal with situations essentially depending on topological aspects of space and time since precise reasoning about them in a naive way is really hard). It looks like that in some mystical way the nature is governed by the same logical rules that provide derivation of theorems from axioms (and a Nobel Prize Laureate in physics, Eugene Vigner, once even wrote a paper about "an unreasonable effectiveness" of mathematics in natural sciences). This quality of classical mathematics and its coordination with nature provided "the supreme respect for mathematics expressed by champions of reality like Galileo, Maxwell, and Heaviside", as Bill Lawvere wrote in his posting to [12].