Characteristics of Object Oriented Modeling Methods

Roland Kaschek and Heinrich C. Mayr

Department of Business Informatics and Application Systems
University of Klagenfurt, Universitätsstrasse 65-67
A-9020 Klagenfurt, Austria

{kaschek | mayr} @ ifi.uni-klu.ac.at

Abstract

Object modeling is replacing classical data oriented modeling approaches. There is a variety of methods, the differences and similarities of which are not obvious as well as their suitability for a given purpose. This is mainly due to the lack of a common meta model which would allow for describing the different approaches in a uniform way. Consequently, it is a non-trivial task for practitioners to make a reasonable choice. The paper is motivated by this problem. Therefore we present a model based treatment of the notion method and specialize it to the notion of object oriented modeling method (OOMM). A set of high level characteristics of OOMM is presented that cover the relevant aspects of well known methods as will be exemplified by their application to OMT. Finally, we review the relevant literature by classifying recent approaches to the comparison of OOMM.

1. Introduction

Object orientation is the today's key paradigm for software development. Object oriented development methods promise to lead to substantial improvements of software development processes ([Bl94]). As a consequence, object oriented software development methods are 'in'. New methods or modifications of already established ones are continuously placed on the market. Practice, however, needs standards. This need led to unification attempts with respect to methods and notation (see, e.g. the COMMA and OPEN process [He96a,b, HB96, HF97, H*97a-c]) as well as the UML process (e.g. [Bu97, FS98, La98]), which lead to the OMG language standard.
UML initially was intended to be the basis for a unified development method (UDM) [BR95]. It was to improve and unify OOAD, OMT as well as OOSE ([Bo94, Ru91, Ja92]). OPEN on

the other hand, focuses more on the second wave methods (compare for this classification [He96a]) like MOSES, SOMA, Martin/Odell and BON (see [HE94, Gr95, MO95, WN95]). Clearly, a standard, regardless whether it is an 'official' or 'de facto', could significantly improve the state of the art in object modeling. But unifying concepts, techniques, procedures or notation of different methods is by no means a simple task.

There are the dangers that deficiencies will be inherited to the intended standard and that the mixture of concepts, notation and techniques is not well designed. A unified method regardless, whether it is intended to be a core method (OPEN) or an integration ([So97]), is a method and thus, following [Ru95], has: "...its own choice of concepts, notation and priorities, subject to the same evaluation criteria as any other new method."
Moreover, a core method might have to be tailored to the particular needs of a given user, a probably complex task that must be taken into account in the context of method selection.

Similar criticism also applies to the less ambitious UML project that concentrates mainly on notation, thus neglecting the dimension of pragmatics. Obviously the pragmatics of a certain symbol is not (at least not fully) determined by itself or by its semantics but by the way people deal with it in modeling. Thus the UML approach might suffer from non-harmonized pragmatics of the incorporated methods. We conclude that dealing with more comprehensive methods, their description, comparison and selection, is still an important issue though the actual trend to unification.

Within this paper we use the term *object oriented modeling* instead of the more common term *object oriented analysis and design*. For, the currently propagated object oriented analysis methods support modeling (of requirements, of a given universe of discourse (UoD) or of a software solution to the problem at hand) but nearly no model analysis. An analysis method, however, should also provide notions, techniques and guidelines for investigating the items at hand in order to generate propositions on the modeled system. Think, e.g. of the identification of invariants in Petri Nets (see [Ba90, p.120ff]), or of properties like reachability or safety, which - at least in principal - can be mathematically studied and used to derive knowledge about the modeled system. For a similar understanding of the notion analysis see [M*98, p.45].

In the following we will present an approach to a comprehensive description of OOMM. For that purpose, section 2 starts with discussing the challenges of method description and selection. We then proceed to a working definition of the notions *method* (section 3) and *object oriented modeling method* (section 4). Section 5 focuses on research objectives in OOMM comparison and prepares the ground for a classification of comparison approaches as found in the literature (section 6). The relationship between OOMM and tools for their application is investigated within section 7. The paper closes with a short summary and an outlook at further work to do (section 8). Appendices A and B[1] contain the complete hierarchy of method characteristics as well as the result of applying them to OMT.

We started our work by identifying limitations of object oriented approaches (see [K*93]) and by a case study (see[K*94a, K*94b, H*97]), which we ran as a 'multi analysis project' by in total 8 teams of students each applying one of the following OOMM: OOA [CY91], OOSE [Ja92], OMT [Ru91], RDD [W*90], SOM [FS93], BOOAD [Bo94]. Each method was applied to the same UoD, namely an extension of the well known IFIP conference organization problem [Ol86]. A subsequent study concentrated on the analysis and characterization of tools supporting that methods (see [KM96, H*97]).

2. Challenges in Method Description and Selection

Our work is triggered by the problem to select, for a given environment, a particular object oriented modeling method. This problem is complicated by certain aspects like the necessity to protect previous investments, to work with industry approved hard- and software and to comply with standard interfaces, to pay staff, tools and offices based on project outcome, as well as the necessity to organize for a homogeneous development style and for homogeneous product structure and quality. Our paper is intended to impact this complex decision by carefully analyzing the basic material, i.e. the modeling methods.

In practice, an OOMM would not be used without the support of a computerized tool for its application: Means for navigating through deliveries, providing for various views on them and for keeping deliveries consistent over the project duration are indispensable [W*98]. Since such tools are available in a huge variety, continuously changing due to new releases and method enhancements, the selection process increased its complexity. Therefore it is hard to choose, within a given environment, the method and tool that best fit into the local conditions (environment needs, tasks, development life cycle/proceeding model for software development).

We believe that a traceable and rational approach to method selection presupposes method comparison. Comparison in turn presupposes a comparison theory based on a comprehensive

[1] Due to space limitations we don't compare our characteristics to other such lists.

and uniform description of the methods to be compared. This may be done on the basis of a set of characteristic properties, i.e. *characteristics* w.r.t. which the methods may be compared. Characteristics are understood here like attributes, i.e. descriptive entities, the type of which in general is not restricted.

3. Methods

In order to get to a comprehensive description of OOMM's we take a model based approach, i.e. an approach focussing on a model of what usually is called method. Other approaches are possible but they don't necessarily focus on the above mentioned assumptions and thus might have a somewhat reduced significance w.r.t. method selection. We therefore discuss the general understanding of what methods are, i.e. (for the scope of this paper) what properties they have and consequently what can be done with them. The purpose of methods, i.e. the goals they are intended to support and the tasks they are to give guidance consequently are in our scope.

Following [Cl89, D*90] and [Bl94] a ***method*** is a *systematic goal driven procedure for gaining knowledge or practical results*. This definition more or less coincides with the one in Langenscheidt- Longman's Dictionary of Contemporary English. It furthermore is compatible with Webster's Dictionary, which according to [LG97, p. 74] defines the notion method with help of the phrase 'regular, orderly and definite procedure'. [Ru95] in his definition of the notion method does not mention, that it should be systematical. This may cause problems in enhancing, learning and applying the method. Furthermore that paper does not explicitly mention goals but just activities the enactment of which is guided by the method. We follow a goal or result oriented approach, since various activities may produce the same result and thus can be used interchangeably depending on the actual conditions. Our view is supported by [Sa90, SR92] and also by the meaning of the Greek word for 'method', which strictly speaking means 'following a way towards a specific location'. Other sources, see, e.g. [He95a,b, Fi95, C*96, So97, B*98, G*97], although dealing with modeling of OOMM don't define the notion method in abstract terms. Thus the way to deal with comparison issues is not necessarily convincing and important aspects might be ignored. Some sources, e.g. [G*97], immediately define the notion object oriented modeling method by means of a conceptual model, i.e. by enumerating their aspects and relationships among them. The relevance of them however is not discussed in that book.

The literal meaning of the word method leads us to identify the following aspects, that somehow are present within a method: a task model, a product model[2], a producer model, a navigational model, and a tool model. We identify this aspects because a method somehow must tell, what kind of job it is good for, i.e. what kind of practical results or knowledge can be accomplished by following it. Additionally the form within which these results are to be obtained needs to be given. Furthermore the method must be directed to a potential user who is the one to produce the products, i.e. the practical results or knowledge described by the product model. Especially in case it is not a trivial job to follow the way prescribed by the method it must provide means for to navigate through the area of problem solving. With respect to the distance between two navigation points succeeding one another immediately the method must denote what tools can be used to bridge the respective gap and how they are to be used.

Having identified the most important aspects to be addressed by a method we now can care for a language to speak about methods. We only are interested in methods humans might benefit from. Thus we assume that the method's assumptions, hints and results or intermediate results can be presented to humans, which necessarily is done by means of a language. A further language aspect in methods is the necessity to teach and improve the method. Therefore to construct our method meta language we can consider the languages used by methods to guide human method users. Clearly this languages depend on the intended users, their organizational context and the desired level of precision and granularity. In practice there will be no universal method allowing to gain interesting knowledge or practical results concerning arbitrary tasks, i.e. we believe that the broader the method's task class the less interesting the possible knowledge and results. Therefore, we expect methods to have restricted domain of method application and universe of discourse (UoD) within it as well as a task class of importance with respect to the UoD.

Since the domain of method application, the UoD and the task class must be distinguished from other entities and described a method needs a language, the *method language*. E.g. the notions provided by OMT for describing static aspects (the object model) form such a language. Obviously for such languages not every syntactically correct phrase also is an appropriate or meaningful one. Therefore, a method should guide its users to build or select appropriate language phrases. For that we introduce the concept of *style*, i.e. the way things are done and represented following the method. This to some extent is realized in OPEN (see [G*97]) where there are recommendations to use or not to use certain phrases which in OPEN are called *deliveries*. A method then has to state what is *good* and what is *bad style*. Our

[2] It seems to be the case, that this term was introduced by C. Atkinson from the Max Planck Institute for Experimental Software Engineering at Karlsruhe, Germany.

concept of style thus generalizes the concept of architectural style discussed in [M*97]. The method style and related aspects are covered by the *proceeding model*[3].

A set of *goal templates* (e.g., 'ensure reusability', 'exploit parallelism', guarantee required 'performance', 'safety' or 'security' aspects) from which the user might choose should also be present. In case of simple methods the set of goal templates even may be empty or only implicitly present. Those methods which offer its users to select goal templates force them to judge about the current situation. The selection of goal templates then may drive the whole process of method application. As was observed by [G*97] one can distinguish between activity and product life cycle. Thus although the product life cycle might decompose into a predefined way by means of phases or stages (see [BM97, C*96]) this structure might not impact the activity life cycle. An iterative activity life cycle might result from goal selections since the judgements whether a particular goal is reached or not are rather problematic. Those modeling methods which provide for easy done model modifications and offer goal templates thus result in a highly iterative activity life cycle. The respective activities are put together in choice and sequence felt to be necessary by the designer, see, e.g. [Wi96, He98 p. 128f.].

Finally, *usage rules* should be provided telling the user what style or goal templates apply in what situation, as well as rules on how to build sentences of good style or how to reach a particular goal efficiently.

To summarize we identify the following main method ***constituents***:

- the *domain of method application* ***DoMA***: a setting wherein the OOMM typically is to be used to solve certain tasks,
- the method's *task class* ***TC:*** denotes the tasks to solve the method is designed for,
- the *method language* ***ML***: allows to state everything important for method application,
- the *proceeding model* ***PM***: guides the method user to proceed during method application. It consists of the following parts:
- ❖ a set of *statements* on *method style* ***MS***: states what kind of ML-statements are to prefer or to reject,
- ❖ a set of *method goal templates* ***MG***: lists objectives a method user might or should try to reach,
- ❖ a set of *usage rules* ***UR***: specifies what style or goal is suitable for what state of affairs within the DoMA,

[3] Since process might be a modeling notion utilized by modeling methods we don't use the more common term process model to prevent from confusing object and meta level concepts.

- a set of *construction rules CR*: tells how to produce expressions of desired style or to reach a desired goal efficiently.

4. Object Oriented Modeling Methods

We consider an OOMM to be a method for the development of object oriented models. These correspond to the stages or phases proposed by the product life cycle of the respective method. They may, depending on the method, range from conceptual to logical or physical models[4]. Furthermore they may, also depending on the method, offer various views on systems. We believe that the domain of method application of an OOMM is systems modeling without any further restriction of the UoD herein. The task class is the production of the respective models. In the following the method constituents introduced above will be specialized to what we call high level OOMM characteristics. A refinement of these characteristics is presented in Appendix A.

4.1 High level characteristics of OOMM

Conception

Additional to the specification of its DoMA an OOMM must state what kind of development process is to be instantiated, i.e. what product and activity life cycle a method user should follow. Connected with these cycles the deliveries to be produced must be specified. Additionally effective techniques to produce them should be explained. The roles, persons should or might take in the modeling process together with their responsibilities, are a special to OOMM compared to methods in general. The producer model additionally should lead the potential method users to have a clear understanding of what they expect from using the method and what their needs are. Also the users should be supported in finding out whether their expectations and needs were met by method application. Furthermore the philosophical assumptions underlying the modeling method should be specified. These assumptions concern the world view incorporated in the method, e.g. what view of concepts like truth or meaning the method presupposes. All this together we call *conception* of the OOMM, i.e. the way the OOMM looks at the production of object oriented models from a high level perspective.

[4] A discussion of this concepts can be found, e.g. in [B*92a, G*97].

Modeling system

We comprehend a model as an imagination that an individual establishes from a thing or process in its environment, see [LM78]. Thus, models reflect judgements on the observed environment by notions and relationships between them. For a systematic treatment of judgements and notions see, e.g. [Pf21]. Since judgements in that source are understood to be certain triplets of notions the method language of an OOMM must offer a system of modeling notions[5]. A modeling notion supports an abstract view at some entity (, e.g. a UoD, some piece of software or of a software development process) which is dealt with during OOMM application.

Furthermore the method language must allow to distinguish, from each other, entities within the UoD, modeling notions and modeling notions instantiated to cover UoD entities. The language also should have means to address the goal templates as well as the method style. In addition to the notions, their representations (words, symbols) are in the realm of the language, too. Important properties of the provided representation concepts, stating how to represent modeling notions instantiated to the UoD, see [LM78], are their practicability, well-definedness and intuitiveness. This leads to the notion of *modeling system*, which is the method's set of modeling notions together with an adjusted set of representation concepts and a modeling concept (see [LM78]) which states how to apply the modeling notions in order to construct the deliveries prescribed by the product life cycle.

Typical OOMM modeling notions are:

S *object, class, value, message, association, object link,*
S abstraction concepts like *aggregation, generalization, classification, clustering,*
S *event, state, state transition, guard, process, actor, activity, action.*

Modeling notions we believe to be necessary in OOMM deal with:

S *dependency* and *compatibility* with respect to the relationships between instances of (the same or different generic views,
S *quantitative UoD aspects* (e.g. timing, money, amount of data, or other sorts of constraints),
S *architectural concepts* describing types of system structure, behavior or constraints,
S *patterns* describing types of connection of system components at a level between classes or objects and architectural concepts,
S *meta notions* concerning developed models and schemas or the development process itself (e.g. *quality, reuse, safety, security* and *management* related notions).

[5] Note that a method thus has a tendency to make the method user blind for things beyond its modeling notions. Consequently each method has principal limitations w.r.t. the knowledge which may be elaborated by following it.

An OOMM should offer certain specifications concerning tool support. It should specify:

— the *minimum set of deliveries to be supported* as well as possible substitutes for symbols or diagram types,
— the minimum *requirements for methodology support,*
— the minimum *requirements for project management support*

Clearly, all the modeling notions should conform to the OOMM conception. Concerning the representation concept of an OOMM it has to be considered, that the intended users or customers should be able to validate the produced model. Thus the representation concept should also be adjusted to the capabilities of the users that are typical for the DoMA of the OOMM. If necessary, several representation concepts could be provided for different user groups.

Methodology

Object oriented modeling is a design activity, often performed in teams (as a source to design theory, see [BM97]). Consequently, in contrast to methods in general OOMM need concepts for team support. Design activities often employ design primitives (i.e. elementary operations modifying deliveries, which are the building blocks for complex operations on deliveries, see, e.g. [B*92]), so that OOMM should have means for dealing with such primitives, too. Furthermore, the analysis of design artifacts, though not a standard task in practice, should be supported by high quality OOMM. In total, these topics form, together with the OOMM proceeding model, what we call the OOMM *methodology*, denoting, similar to [Cl89,Bl94,Sa90], a theory of techniques, procedures and joint goal templates, the aim of which is to guide the OOMM user in applying the modeling system to solve development tasks as addressed by the OOMM conception. It supports

- the selection and instantiation of goal templates that fit into the actual development situation,
- the instantiation of the proceeding model to a development process adequate to the realization of the selected goals,
- realization of the instantiated goal templates under the instantiated development process,
- gaining help in the case that the OOMM is not (or no further) applicable.

Possible goals templates among others are

- complexity treatment,
- reuse of all kinds of former development results (model components, patterns {see e.g. [Fo97, La98, BP98]} or programs)
- organizing all types of reuse (sewing and earning, ...), see [J*97],

- exploitation of UoD inherent parallelism,
- ensuring performance requirements, safety, security,
- ensuring quality standards or improving quality (for a treatment of quality aspects in conceptual models see, e.g., [L*94, K*95b]),
- a controlled design document evolution,
- validation and verification of deliveries,
- reduction of development costs and economical resource usage,
- support of (various kinds of) prototyping.

Note that modeling notions are required for each goal template in order to address related UoD aspects as well as guidance to reach the instantiated goal. In turn modeling notions and guidance each depend on adequate goal templates. Clearly neither goal templates nor modeling notions or guidelines should conflict the OOMM conception. The goal templates together with the respective selection and realization support contain most of the knowledge incorporated into the OOMM. Thus it is a good rule of thumb to concentrate on this part of an OOMM during selecting for an OOMM.

Documentation

Especially with respect to method selection in practice we add *documentation* as a further constituent for OOMM. It covers the presence and distinction of the method in literature and practice, the kind of available literature as well as the clearness of the available method descriptions and case studies. Further topics are training material and courses, certificates.

There is an increasing amount of literature on OOMM in general and of specific documentation material in detail. The OPEN method, e.g. requires several books for its documentation, see [G*97]. Clearly this might lead to problems in understanding and applying the resp. method.

For more details of the OOMM characteristics see appendix A.

For a specific OOMM the proceeding model aspects might be empty or not fully elaborated. Especially its language may lack of notions to reason about the UoD. We could view, e.g., the classical entity relationship model (ERM) (see [Ch76]) as an OOMM for UoD having a significant time invariant, i.e. static aspect. It has practically no proceeding model.

There are several investigations in the domain of semantic modeling, see e.g. [PM88, HM90, UD86] and [KL85]: They concentrate to a large extent on the language means that are offered by the respective data model (viewed as a modeling method) and on its semantics. [B*92] and others try to give a proceeding model (rules for obtaining 'good' schemas in the above sense) for an ERM-based software development method. [Ei91] proposes style preferences and usage rules for a specific ERM.

With respect to the domain of application the characteristics concentrate on specific aspects of the OOMM, which we believe to be rather independent. We furthermore believe, that methodology and conception are the most important characteristics of an OOMM.

5. Research objectives in OOMM comparisons

OOMM comparisons necessarily are strongly influenced by their objectives. We present some objectives we found in the literature:

- [Be92] points out three objectives:
  (1) to improve the *understanding* of object orientation in general and of OOMM in particular,
  (2) to improve the process of *selecting* among OOMM and
  (3) to *identify* problems of OOMM that might be used as arguments against the introduction of such a method into the 'comparers' environment.

- [SO94a, SO94b] and [Gi93] use comparison as a means for *integrating* various OOMM into one (new) OOMM. [Gi93] and [SO94b] take the construction of a case tool supporting various OOMM as a reason for the integration of the respective methods. This objective to some extent also applies to the UML process. Additionally UML tries to provide for an OOMM language standard.

- The proponents [H*97b] of the OPEN process try to provide for a standard OOMM, which is a core method to be extended if necessary. This project poses the new questions how to find out what tasks cannot efficiently be mapped into the task class of the core method and how to find and instantiate the appropriate extension.

6. A classification of comparison approaches

Some work has been conducted recently focussing on the comparison of OOMM or SDM's in general, see [St94, SO94a, FK92, MP92, Ii94, SO94b, Bl94, Be92, SC93, DTLZ93, KH87, Gi93 and Hu94]. Methodological differences may be observed in these approaches. [SO92] give a classification of possible approaches based on [So83]; it turns out that the differences between the approaches consist mainly in the way these approaches define a *meta-language* for reasoning about the OOMM at hand. This is not very striking since comparing OOMM, in first, affords to speak about them. Taking a more sophisticated view we observe, that there are the following (possibly overlapping) classes of approaches to method comparison:

(1) ***Case based approach***: Determined by

  (a) the *definition of a new OOMM* which is to serve as a reference model for comparison; or

  (b) by the *derivation of a set of OOMM characteristics* from the OOMM that are to be compared; the elements of that set then become the basic notions of the meta-language; or

  (c) by the *definition of some meta-language* from scratch; or

  (d) by the *integration of the OOMM* to be compared on the basis of a formalized description. The meta-language then consists of the integration result together with the description means (e.g. a data model). This approach has been introduced by [SO94a] and [SO94b].

(2) ***DoMA centered approach***: Determined by relating OOMM to DoMA types and by defining an appropriate meta-language for a DoMA specific comparison. Examples for more specific DoMA are: process control systems, information systems, multi media or virtual reality systems.

(3) ***Model based approach***: Determined by identifying those characteristics of OOMM that are essential with respect to the comparison purpose, and by matching concrete methods against these characteristics. The challenge of this approach consists in determining of what is to be considered as essential thus getting a clear understanding of OOMM. Our approach as presented in the foregoing sections belongs to this class.

Most of the approaches mentioned so far suffer from the lack of assessing advantages a certain OOMM has compared to others (with respect to the given DoMA), a point that is essential for method selection. [D*93] report on a case study where they employed a human expert for to assess schemas that were developed by several groups of students. Each group used a specific OOMM. The expert assessed for each group two schemas: the initial one (first result) and the final one (last result after enhancements). Each method then was ranked according to the scores of their respective schemas.

[SC93] use software metrics to assess the schemas that were produced in a case study. They again take the resulting score as the (multi dimensional) method score. Both rankings however, do not take into account that they depend (among other factors) on the UoD.

Another approach is to use metrics for assessing methods themselves. I.e., method characteristics are instrumentalized to criteria by providing a value range for each of them. All that leads to a fourth class of comparison approaches. It focuses on ranking OOMM according to a quantifying procedure, that does not take into account organizational needs:

(4) ***Method ranking approach.***

Though not limited to OOMM comparisons also [B*98] is worth to being mentioned, since it attempts to develop a set of benchmark problems. Method selection then may be done by comparing the problem at hand with the benchmark problems and the choosing the method which performed best w.r.t. this problem. This approach is related to the DoMA-centered approach and the ranking approach. We create a new class for approaches like this because within a given DoMA methods might perform differently on benchmark problems. Additionally no formal ranking mechanism needs to be given for this approach. The designers taste might be viewed as sufficient. The class crated is called:

(5) ***Benchmark problems approach***

Like every kind of investigation OOMM comparison affords a certain underlying philosophy, i.e., a conception of the real world and the role of OOMM within it. Think, e.g., of ontology, epistemology, subjectivism, objectivism. Some of the most common approaches for a theory of truth are discussed in [An92]. Various approaches to a theory of meaning are documented in [HK95]. The comparison approaches cited above don't deal with questions like these. The same holds for the methods that were under our investigation (see introduction). Nevertheless the topic has been discussed by [KH87] with respect to data models. Further research concerning OOMM and their philosophical assumptions is done in [KO94], [TW91] and in [Ka88]. More research concerning this topic seems to be necessary. We therefore create a further class focussing on the philosophical assumptions made by OOMM to gain practical results or knowledge:

(6) ***Epistemological approach.***

At the first glance it might seem that the above classification of approaches is thought to be non-overlapping. We explicitly mention that this is not the case. E.g., if one wants to deal with measurement issues, class (4) seems to be appropriate. But since measurement implies the identification of the things to be measured also class (3) is affected. Thus, the classes more likely correspond to clusters of comparison aspects.

[Be92] describes formally, as an essential ingredient of an OOMM, the fundamentals of the proceeding model proposed by the methods he investigated. These descriptions are organized according to a software life cycle model. However, this approach does not help to match the methods against the customers needs. This weakness can be solved by focussing on the goals the proceeding model of the methods suppose and the selectors needs. The present paper therefore contributes to this aim and is intended to continue the work in [Be92].

[SO94] mention as a deficiency of recent OOMM comparisons that they are not at a sufficient level of detail. [Gi93], e.g., describes OMT without any reference to its proceeding model. This is in contradiction with [Ii94] where OMT is analyzed at a higher level of detail and is estimated because of its good (modeling) instructions. A further deficiency is mentioned in [Be92], who criticizes that some method comparisons are suffered from untraceability.

Clearly, concerning the selection problem, only those comparison approaches are adequate that help to focus on the goals that are relevant with respect to the enterprise policy and needs. Furthermore, when designing a comparison method one has to take into account that weaknesses of some parts of an OOMM language can be compensated by other parts and by the process model (see e.g. [SO94b]).

7. The Relationship Between Methods and Tools

As has been mentioned before our long-term objective is method selection. In practice this nearly implies the selection of a tool since OOMM application without support by a tool is not the practitioners work and life.

Methods are (abstract) means for managing a certain class of tasks. In contrast to that, we consider a tool to be a (concrete) means that supports the use of a method for to solve concrete tasks. i.e. instances of that method's task class. Consequently, an OOMM tool is to support the use of the respective OOMM. Therefore in principle we suppose to first choose the OOMM and then a supporting tool. Occasionally an OOMM may be defined by a tool. But this has no impact on their relationship.

There are tools supporting several OOMM (see, e.g. Object Maker [OM93], Paradigm Plus[PP93] or Rational Rose/C++ 3.0 [RR30]); we may see them as a tool box containing tools for more or less similar purposes within the framework of OOM (think, e.g. of a box with different screw drivers). OOM tools usually are software systems so that besides of characteristics that focus on tool aspects also software characteristics have to be considered.

According to the precedence of methods over tools their characteristics mainly have to fit to method characteristics. E.g., the modeling notions and representation concepts of the method's modeling system should be supported appropriately and completely; the achievement of methodology goals should be ensured. Not all of the method characteristics must give raise for a tool characteristic. This especially holds for the philosophical assumptions as part of method conception (a tool may not force its user to follow it's method's philosophy: think, e.g., of a craftsman driving a nail into a wall by punches with a screwdriver) and to some extent also for some of the method documentation characteristics. But with respect to certain important aspects the method should prescribe properties a supporting tool has to observe.

In case of OOMM tool boxes (see above) another characteristic is that they offer means for the translation of deliveries from the language of one method into the language of another method they support. Concerning this we point out, that the meta model of such tools must be essentially enlarged compared to those of the methods since layout information must be kept as far as possible during translations. Our experience with some products indicates that this still is a challenge for tool producers.

To sum up, we introduce again four top level characteristics: **methodology support**, **general software characteristics**, **documentation** and **environment**. For more details concerning tool characteristics see [KM96].

8. Summary And Outlook

Starting with the objective of OOMM selection we identified OOMM comparison to be a fundamental issue. Starting from the Greek origin of the word method we analyzed the method concept and derived characteristics of OOMM. We discussed approaches to method comparison known from the literature, to classify and analyze them with respect to their relevance for method selection.

It turned out that to deal with selections of OOMM their methodology in general and the goal templates in particular have to be investigated since one of the promises of object orientation is simplicity of model modification. To do so we developed, based on the results of a multi-project study, a hierarchy of characteristics, the most important of which were conception and methodology. These characteristics may be used to describe OOMM.

A tool is to be constructed that will help users to compare OOMM. It can be based on benefit analysis as was done in [St94]. Clearly user needs and expectations have to be taken into account for this analysis. These needs and expectations need to be assessed according to two dimensions. The first is the relevance or importance of the respective item. The second is its relative amount of fulfillment.

Additionally a procedure is needed to find out the needs and expectations the method is to meet. We intend to approach this task by observing the capability maturity model, see [Hu89], and relate the selection to the specific maturity level of the selecting organization.

Finally, a comparison and analysis of the known lists of OOMM characteristics will have to be done. The respective investigation will give an improved insight in the value of our model based approach to method selection.

References

[An92] Andersson, G.: Wahr und falsch; Wahrheit. In [SR92,pp. 369-375].

[A*91] Van Assche, F., Moulin, B., Rolland, C. (eds.): Object Oriented Approach in Information Systems. North Holland 1991.

[Ba90] Baumgarten, B.: Petri-Netze. BI-Wissenschaftsverlag. Mannheim, Wien, Zürich. 1990.

[Be92] Berard Software Engineering, Inc.:A Comparison of Object-Oriented DevelopmentMethods. Berard Software Engineering, Inc. 1992.

[Bl94] Blum, B.I.:A Taxonomy of Software Development Methods. In: Communications of the ACM Vol. 37, No 11, 1994, pp. 82-94.

[Bo94] Booch, G.: Object-Oriented Analysis and Design with Applications. Benjamin/Cummings Publishing Company, Inc.. Redwood City, Cal. et al.. 1994.

[Bu97] Burkhard, R.: UML- Unified Modeling Language. Addison-Wesley. Bonn et al.. 1997.

[BM97] Braha, D., Maimon, O.: The Design Process: Properties, Paradigms, and Structure. IEEE Transactions on Systems, Man, And Cybernetics, Part A: Systems and humans, Vol. 27,2(1997):146-166.

[BP98] Blaha M., Premerlani W.: Object-Oriented Modeling and Design for Database Applications, Prentice Hall, Inc., Simon & Schuster / A Viacom Company, Upper Saddle River, New Jersey, 1998.

[BR95] Booch, G., Rumbaugh, J.: Unified Method for Object-Oriented Development. Rational Software Corporation. Santa Clara, CA. 1995.

[B*92] Batini C., Ceri S., Navathe S.: Conceptual Database Design: An Entity Relationship Approach. Benjamin Cummings Publishing Company, Inc.. Redwood City, Cal. et al.. 1992.

[B*98] Bahill A.T., Alford M., Bharatan K., Clymer J.R., Dean D.L., Duke J., Hill G:, LaBudde E.V., Taipale E.J., Wymore A.W.: The Design- Methods Comparison Project, IEEE Transactions on Systems, Man And Cybernetics Part C 28,1(1998):80-103.

[Ch76] Chen, P.: The Entity-Relationship Model- Toward a Unified View of Data. ACM Transactions on Database Systems, 1(1976):166-192.

[Cl89] Claus, V.: Entwicklung von Informatik-Methoden. In: Handbuch der modernen Datenverarbeitung. 150, 1989, pp. 25-44.

[CY91] Coad, P., Yourdon, E.: Object Oriented Analysis. Prentice Hall.Englewood Cliffs. 1991.

[C*96] Chonoles M.J., Schardt J.A., Magrogan P.J.: Speaking of Methods, ROAD Mar.-Apr. 1996:8-12,15.

[D*90] Drosdowski G., Müller W., Scholze-Stubenrecht W., Wernke M. (Eds.): Duden Fremdwörterbuch, Bibliographisches Institut & F.A. Brockhaus AG, Mannheim, 1990.

[D*93] Drake, J., Tsai, W.T.,Lee, H.J., Zualkernan, I.: Object-Oriented Analysis: Criteria and Case-Study.International Journal of Software Engineering and Knowledge Engineering 3,3(1993), pp. 319-350.

[Ei91] Eick, C.F.: A Methodology for the Design and Transformation of Conceptual Schemas. In: Proceedings of the 17th International Conference on Very Large Data Bases. pp. 25-34.

[EN94] Elmasri, R., Navathe, S.: Fundamentals of Database Systems. The Benjamin/Cummings Publishing Company, Inc.. Redwood City, CA et al.. 1994.

[FS93] Ferstl, 0., Sinz, E.: Der Modellierungsansatz des Semantischen Objektmodels (SOM). In: Bamberger Beiträge zur Wirtschaftsinformatik, Nr. 18(1993).

[FK92] Fichman, R.G.,Kemerer, C.F.:Object-Oriented and Conventional Analysis and Design Methodologies. IEEE Computer. October 1992. pp. 22-39.

[FS98] Fowler M., Scott K.: UML konzentriert, Addison-Wesley Longman Verlag GmbH, Bonn et al. 1998.

[Gi93] Gilpin,M.:A Comparison of Object Oriented Analysis and Design Methods. Mike Gilpin, INTERSOLV. CASE WORLD, March 8-10, 1993.

[Gr95] Graham I.M.: Migrating to Object Technology, Addison-Wesley, Wokingham, UK, 1995.

[G*95] Györkös, J., Krisper, M., Mayr, H.C. (eds.): Re-Technologies for Information Systems. Oldenbourg Verlag, 1995.

[G*97] Graham I.M., Henderson-Sellers B., Younessi H.: The OPEN Process Specification, Addison Wesley Longman Ltd., Edinburgh Gate Harlow, England, 1997.

[He96a] Henderson-Sellers B.: Convergence Is in the Air, ROAD Mar./Apr. 1996:47-49,54.

[He96b] Henderson-Sellers B.: The COMMA Project: First Steps, ROAD May/June 1996:49-52.

[He98] Hesse W.: Vorgehensmodelle für objektorientierte Software- Entwicklung, In: Kneuper R., Müller-Luschnat G., Oberweis A. (eds.): Vorgehensmodelle für die betriebliche Anwendungsentwicklung, B.G. Teubner, Stuttgart, Leipzig, 1998, pp. 110-135.

[Hu89] Humphrey, W.S.: Managing the Software Process. Addison-Wesley, Reading, MA. 1989.

[Hu94] Hutt, A.: Object Analysis and Design, Description of Methods. John Wiley & Sons, N.Y.. 1994.

[HB96] Henderson-Sellers B., Bulthuis A.: COMMA: Sample metamodels, JOOP Nov./Dec. 1996:44-48.

[HE94] Henderson-Sellers B., Edwards J.M.: Book two of Object-Oriented Knowledge: The Working Object, Prentice Hall, Sydney, 1994.

[HF97] Henderson-Sellers B., Firesmith D.G.:COMMA Proposed core model, JOOP Jan. 1997:48-53.

[HK95] Holm, P., Karlgren, K.: Theories of meaning and different perspectives on information systems. In [FHO95, pp. 3-35].

[HM90] Hong, S., Maryanski, F.: Using a Meta Model to Represent Object Oriented Data Models. Proceedings of the 6th. International Conference on Data Engineering. February 5-9, 1990. Los Angeles, CA, pp. 11-19.

[H*96] Hochmüller, E., Kohl, C., Mayr, H.C., Mittermeir, R.: CASE-Tools - Perspektivenwandel am Weg vom Anfänger zum Experten Informatik/Informatique, SI, Nr. 3 June 1996, pp. 47 – 50.

[H*97a] Henderson-Sellers B., Firesmith D.G., Graham I.M.: OML metamodel: Relationships and state modeling, JOOP Mar./Apr. 1997: 47-51.

[H*97b] Henderson-Sellers B., Firesmith D.G., Graham I.M.: Methods Unification: The OPEN methodology, JOOP May 1997: 41-43,55.

[H*97c] Henderson-Sellers B., Firesmith D.G., Graham I.M.: The Benefits of Common Object Modeling Notation, JOOP Sep. 1997:28-34.

[Ii94] Iivari, J.:Object-oriented information system analysis: A comparison of six object-oriented analysis methods. In: [VO94]. pp. 85-110.

[Ja92] Jacobson, I., Christerson, M., Jonsson, P., Övergaard G.: Object-Oriented Software Engineering - A Use Case Driven Approach. Addison Wesley. 1992.

[J*97] Jacobson I., Griss M., Jonsson P.: Making the Reuse Business Work, IEEE Computer Oct. 1997:36-42.

[Ka88] Kangassalo, H.: Foundations of conceptual modeling: A Theory construction view. In: Information Modeling and Knowledge Bases. IOS Press. Amsterdam, Washington DC, Tokyo. 1990. pp. 19-35.

[KH87] Klein, H.K., Hirschheim, R.A.: A Comparative Framework of Data Modeling Paradigms and Approaches. The Computer Journal, Vol. 30, No. 1, 1987. pp. 8-15.

[KL85] King, R., McLeod, D.: Semantic Data Models. In: [Ya85], pp. 115-150.

[KM96] Kaschek, R., Mayr, H.C.: A Characterization of OOA Tools. In: 2nd International Symposium on the Assessment of Software Tools, held in Toronto, Canada in May 1996. IEEE Computer Society Press. 1996.

[KO94] Kristensen, B.B., Osterbye, K.: Conceptual Modeling and Programming Languages. ACM SIGPLAN Notices, Vol. 29. No. 9. 1994, pp. 81-90.

[K*93] Kaschek, R., Kohl, C., Mayr, H.C.: Grenzen objekt-orientierter Analysemeth oden am Beispiel einer Fallstudie mit OMT. In: [MW93] pp. 153-154.

[K*94a] Kaschek R., Mayr H.C., Kop C., Kohl C.: Characteristics of OOA Methods and Tools – A First Step to Method Selection in Practice, Technical Report 01/1994, University of Klagenfurt, Klagenfurt, 1994.

[K*94b] Kaschek R., Khouri B., Kohl C., Kop C., Mayr H.C.: A Hierarchical Classification System for Comparing OOA Methods and Tools, Poster Presentation at CRIS 94, University of Limburg, Maastricht, The Netherlands, September 26-28, 1994.

[K*95a] Kaschek, R., Kohl, C., Mayr, H.C.: Cooperations-An Abstraction Concept Suitable for Business Process Reengineering. In: [G*95], pp. 161-172.

[K*95b] Krogstie, J., Lindland, O.J., Sindre, G.: Defining quality aspects for conceptual models. In [FHO95,pp. 216-231].

[La98] Larman C.: Applying UML And Patterns, Prentice Hall PTR Prentice Hall, Inc., Upper Saddle River, NJ, 1998.

[LG97] Luqi, Goguen J.A.: Formal Methods: Promises and Problems, IEEE Software Jan./Feb. 1997:73-85.

[LM78] Lockemann P.C., Mayr H.C.: Rechnergestützte Informationssysteme. Springer Verlag. Berlin et al..1978.

[L*86] Langefors, B., Verrijn-Stuart A.A., Bracchi, G. (Eds.): Trends in Information Systems. North Holland. 1986.

[L*94] Lindland, O.I., Sindre, G., Solvberg, A.:Understanding Quality in Conceptual Modeling. IEEE Software. March 1994. pp. 42-49.

[MO95] Martin J., Odell J.J.: Object-Oriented Methods: A Foundation, Prentice Hall, Englewood Cliffs, NJ, 1995.

[MP92] Monarchi, D.E., Puhr, G.I.:A Research Typology for Object-Oriented Analysis and Design. Communications of the ACM Vol. 35, No. 9, 1992. pp. 35-47.

[MW93] Mayr, H.C., Wagner, R. (Eds.): Objektorientierte Methoden für Informationssysteme. Informatik aktuell. Springer Verlag. 1993

[M*98] Monroe R.T., Kompanek A., Melton R., Garlan D.: Architectural Styles, Design Patterns, and Objects, IEEE Software Jan./Feb. 1997:43-52

[Ol86] Olle, T.W.: IFIP Comparative Review of Information Systems Design Methodologies: Problem Definition. In: [L*86], pp. 55-56.

[OMT93] OMTool User Guide, Februar 1993 for OMTool Release 1.2

[OM93] ObjektMaker Tutorial Version 2.1 for UNIX/VMS, Objekt Maker User's Guid Version 2.1 for UNIX/VMS .

[O*83] Olle T., Sol H., Tully C. (Eds.): Information Systems Design Methodologies: A Feature Analysis. North Holland. Amsterdam et al..1983.

[Pf21] Pfänder A.: Logik. Verlag von Max Niemeyer. Halle an der Saale.1921.

[PM88] Peckham J., Marjanski F.: Semantic Data Models. ACM Computing Surveys 20,3(1988), pp. 153-189.

[PP93] Paradigm Plus Reference Manual, Release 2.0 for UNIX and Windows Workstations. Paradigm Plus Methods Manual Release 2.0 for UNIX and Windows Workstation.

[RR30] Rational Rose/C++ 3.0

[Ru91] Rumbaugh J., Blaha, M., Premerlani, W., Eddy, F., Lorensen W.: Object Oriented Modeling and Design. Prentice Hall 1991.

[Sa90] Sandkühler H.J. (Ed.): Europäische Enzyklopädie zu Philosophie und Wissenschaften, Felix Meiner Verlag, Hamburg, 1990.

[So83] Sol, H.: A Feature Analysis of Information Systems Design Methodologies: Methodological Considerations. In [O*83], pp. 1-9.

[St94] Stein, W.: Objektorientierte Analysemethoden - Vergleich, Bewertung Auswahl. Reihe: Angewandte Informatik, Band 12. Herausgegeben von Balzert, H. Wissenschaftsverlag, Mannheim, Leipzig, Wien, Zürich. 1994.

[SC93] Sharble, R.C., Cohen, S.S.: The Object-Oriented Brewery: A Comparison of Two Object-Oriented Development Methods. ACM SIGSOFT Software Engineering Notes 18,2(1993), pp. 60-73.

[SO92] Song, X., Osterweil, L.J.:Toward Objective Systematic Design-Method Comparisons. IEEE Software. May 1992, pp. 43-53.

[SO94a] Song, X., Osterweil, L.J.:Experience with an Approach to Comparing Software Design Methodologies. IEEE Transaction on Software Engineering 20,5(1994), pp. 364-384

[SO94b] Song, X., Osterweil, L.J.:Using Meta-Modeling to Systematically Compare and Integrate Object-Oriented Modeling Techniques. Manuscript 1994, pp. 1-20.

[SR92] Seiffert, H., Radnitzky, G.(Eds.): Handlexikon zur Wissenschaftstheorie. Deutscher Taschenbuch Verlag GmbH & Ko KG. München. 1992.

[TW91] Tagaki, K., Wand, Y.: An Object-Oriented Information System Model based on Ontology. In: [A*91], pp. 275-296.

[UD86] Urban, S.D., Delcambre, L.M.L.: An Analysis of the structural, dynamic and temporal aspects of semantic data models. In: Proceedings of the International Conference on Data Engineering. 1986, pp. 382-387.

[VO94] Verrijn-Stuart A.A., Olle T.W.: Methods And Associated Tools for the Information System Life Cycle. North-Holland. Amsterdam et al.. 1994.

[Wi96] Williams J.D.: Managing Iteration in OO Projects, IEEE Computer Sep. 1996:39-43.

[WN95] Walden K., Nerson J.-M.: Seamless Object-Oriented Architecture, Prentice Hall, Englewood Cliffs, NJ, 1995.

[W*90] Wirfs-Brock, R., Wilkerson B., Wiener, L.: Prentice Hall, Englewood Cliffs, NJ, 1990.

[W*98] Weidenhaupt K., Pohl K., Jarke M., Haumer P.: Scenarios in System Development: Current Practice, IEEE Software, Mar./Apr. 1998:34-45.

Appendices:

Appendix A: A Hierarchy of method characteristics

I. **OOMM Conception**

1. intended domain of method application
   - A. real time applications
   - B. dialogue intensive applications
   - C. applications with intrinsic concurrency
   - D. data intensive applications
   - E. knowledge intensive applications
   - F. computation intensive applications
   - G. planning systems
   - H. configuration management systems
   - I. decision support systems
   - J. distributed systems
   - K. CSCW
   - L. diagnosis systems
   - M. CAD
   - N. embedded systems
   - O. life critical systems
   - P. enterprise modeling
   - Q. others
2. software life cycle (SLC) model supposed and SLC-phases to be instantiated
3. basic philosophy
   - A. ontological: basic view/model at/of UoD (a priori semantics, basic notions)
   - B. epistemological: basic view/model at/of instantiated knowledge gaining processes; prototyping
   - C. basic principles describing the roles and interactions of humans and enterprises in the real world
4. intended design products (documents)
   - A. types and instances
   - B. contents
   - C. usage operations
5. starting point of the OOMM
   - A. prerequisites
   - B. pre-phases
6. involved persons with their roles, qualification profiles and responsibilities

II. **Modeling System of the OOMM**

1. provided set of modeling notions
   - A. for *static* UoD *aspects,* e.g., object, class, value, value set, relationship, attribute, role, cardinality
   - B. for *dynamic* UoD *aspects*, e.g., state, event, state transition, operation, condition, time, concurrency, constraint
   - C. for *functional* UoD *aspects,* e.g., process, data flow, data store, actor, process specification
   - D. for *communicational* UoD *aspects*, e.g., message, communication role

(master, slave, etc.), communication type

E. for *organizational* UoD *aspects*, e.g., business process, cooperation, job-unit, task

F. for *architectural* UoD *aspects*, e.g., system, subsystem, module, layer

G. for *temporal* UoD *aspects*, e.g., before, after, simultaneously

H. for *operational* UoD *aspects*

- a. simple, complex
- b. short or long running
- c. consistency preserving and/or checking
- d. extracting (queries) or generating
- e. others

I. for *relational* UoD *aspects*

- a. aggregation
- b. generalization
- c. inheritance
- d. uses
- e. has
- f. instance of
- g. communication:
  - synchronous,
  - asynchronous or
  - synchronizing
- h. existence or identification dependency, qualification
- i. clustering
- j. others

J. others

2. defined sub models (i.e. coherent modeling subsystems)

A. their scope with respect to the universe of discourse

B. interdependencies between sub models

C. compatibility of sub models

D. integration of sub models

3. independence of modeling notions

A. selectivity of notions w.r.t UoD aspects

B. overlapping semantics

C. restriction of modeling notions to certain UoD types or UoD parts

D. practicability for modeling

4. adequacy of the modeling system with respect to the conception

5. quantitative aspects

A. concepts for assessing data volumes

B. concepts for assessing quality

C. concepts for assessing performance

D. concepts for assessing validity of design documents

E. concepts for assessing consistency of design documents

F. concepts for assessing correctness of design documents

6. illustration of modeling notions by examples

A. number of examples

B. illustrated modeling notions

C. ignored modeling notions

- D. number of examples containing all modeling notions
- E. incorrect defined modeling notions
- F. modeling notions with illustrations that conflict their definition

7. sets of representation concepts; per set:
   - A. correspondence between representation concepts and modeling notions
     - a. completeness of the set of representation concepts
     - b. uniqueness in addressing modeling notions
     - c. complexity of the representation concepts
     - d. minimality of representation concepts
   - B. extendability of representation concepts
   - C. practicability of representation concepts
     - a. reasons for practicability
     - b. reasons for impracticability
   - D. novelty of the representation concepts
     - a. similarity with well known representation concepts
     - b. necessary learning effort
8. quality criteria
   - A. local quality criteria with respect to sub model design documents (sub schema)
   - B. global quality criteria for the whole schema
   - C. understandability and persuasion power of quality criteria

III. Methodology

1. process model
   - A. covered parts of the application domain
   - B. covered life cycle phases
   - C. completeness with respect to the defined sub models
   - D. goals
     - a. intention
       - to reach security, safety, ergonomy, quality,
       - to describe concurrency/ parallelism or timing aspects in the UoD,
       - to cope with complexity,
       - to validate or verify the schemas (design products),
       - to reuse former thought and design results,
       - to do inter or intra sub schema integration,
       - to assess data volumes or performance,
       - to produce executable and/or animatable specifications,
       - to cope with vague knowledge

- to focus on real business processes in the enterprise to encourage BPR
- other intentions

b. evidence of goals to the involved persons

c. power of persuasion

d. distinctness of goal description

E. goal realization support

a. covered goals

b. reachable goals

c. quality of realization support

F. goal selection support

a. covered goals

b ignored goals

c. quality of selection support

G. flexibility of schema-modification

a. restrictions

b. articulation of sub schema-interdependencies

c. support in handling sub schema-interdependencies

H. user participation

a. user roles

b. selection support concerning user roles

2. design primitives (schema or design process 'molecules')

A. proposed design primitives

B. adjustment to the proceeding model and to the representation concepts

C. advantages

D. support of intuitive use

E. available guidelines when to use which primitive

F. controlled schema evolution

G. others

3. means for design product analysis, e.g., reachability analysis, invariants

4. method style support

A. for the use of modeling notions, techniques or procedures in special situations

B. document making

C. usage of data dictionary

5. applicability

A. appropriate support for every process model step

B. feasibility of proposed process model steps

C. transition between process model steps

a. definedness of transitions

b. order of proposed steps

D. conclusiveness of process model steps

a. intuitive

b. reasonable

c. convincing

E. adequacy with respect to the conception

F. learnability of the process model

6. management

A. single team work support

B. multiple team work support

C. adjustment to conception, modeling notions and methodology

IV. Documentation

1. references

   A. publicly available definition of the OOMM

   B. completeness

   C. availability of completely documented case studies

   D. reports of experience

   E. presence in practice

   F. others

2. education

   A. textbook

   B. courses

   C. certificates

3. clearness

   A. distinction from other OOMM in literature

   B. tolerance against variations of

      a. conception

      b. methodology

      c. process model

   C. description of scope of the OOMM

Appendix B: Application of method characteristics to OMT[6]

I. *OOMM Conception*

1. *intended domain of method application*

OMT is a general purpose method which was applied by Rumbaugh et al. [Ru91, p. 9] at General Electrics among others to the following tasks: compiler construction, graphic applications, user interfaces, database applications, CAD systems, control systems, simulation and metamodels.

2. *software life cycle (SLC) model supposed and SLC-phases to be instantiated*

SLC model consists of the phases analysis, system design, object design (in this order).

3. *basic philosophy*

No philosophical assumptions are stated.

A. *ontology: basic view/model at/of UoD (a priori semantics, basic notions)*

OMT is an object oriented method and thus uses object identity as a modeling notion. Therefore OMT assumes that different objects may not be distinguishable by their attributes. Further OMT assumes that all UoD can be understood as a system of communicating objects. Objects in general are thought to be independent.

B. *epistemology: basic view/model at/of instantiated knowledge gaining processes; prototyping*

No such assumptions are stated.

C. *basic principles describing the roles and interactions of humans and enterprises in the real world*

No such principles are stated.

4. *intended design products (documents)*

A. *types and instances*

types: PROBLEM STATEMENT, OBJECT MODEL DIAGRAM, STATE DIAGRAM, GLOBAL EVENT FLOW DIAGRAM, SCENARIO, EVENT TRACE, DATA FLOW DIAGRAM, DATA DICTIONARY

instances: OBJECT MODEL DIAGRAM, STATE DIAGRAM, EVENT TRACE, GLOBAL EVENT FLOW DIAGRAM and DATA FLOW DIAGRAM each take a graphical form whereas PROBLEM STATEMENT, SCENARIO and DATA DICTIONARY take a textual form.

B. *contents*

[6] This study is based on the material that was accessible to us.

PROBLEM STATEMENT: a description of the universe of discourse and the customers needs;

OBJECT MODEL DIAGRAM: description of the static aspects [Ru91, p. 21], i. e. to what something happens to;

STATE DIAGRAM: description of the flow of control [Ru91, p.84], i. e. causality of the things happening.

DATA FLOW DIAGRAM: specifies the result of computations without specifying how or when they are computed, i. e. what happens [Ru91, p.123];

SCENARIO, EVENT TRACE: show communication between specific objects;

GLOBAL EVENT FLOW DIAGRAM: shows the collection of all messages exchanged between all or a specific set of objects;

C. *usage operations*

STATE DIAGRAMS are constructed out of EVENT TRACES each of which is derived from a SCENARIO. GLOBAL EVENT FLOW DIAGRAMS are used for testing whether all objects do have the necessary methods. The DATA FLOW DIAGRAMS are used as

B. for *dynamic* UoD *aspects*,

Additionally to the notions mentioned OMT offers TIMING CONSTRAINTS, NESTING, ENTRY- and EXIT ACTIONS, (operations are detailed into ACTIVITY and ACTION), SPLITTING and SYNCHRONISATION of control

C. for *functional* UoD *aspects*

basis of algorithm design in object design, a method phase belonging to implementation, which in general is out of the scope of OOA. EVENT TRACES may be used to validate STATE DIAGRAMS.

5. *starting point of the OOMM*

No special assumptions are made. The method starts with the production of a PROBLEM STATEMENT. OMT does not give hints how to prepare the problem statement.

6. *involved persons with their roles, qualification profiles and responsibilities*

Requestor, analyst, designer, developer.

II. ***Modeling System of the OOMM***

1. *provided set of modeling notions*

A. *for static UoD aspects*

Additionally to the notions mentioned OMT offers NESTING and ORDERING. The method authors emphasize that one should not use associations between more than three classes.

Additionally to the notions mentioned

OMT offers nesting

F. for *architectural* UoD *aspects*

OMT offers SYSTEM, SUBSYSTEM, MODULE and SHEET.

G. for *temporal* UoD *aspects*

No modeling notions are provided. But with respect to timing conditions it is assumed that the method users have an naive understanding of time. Especially BEFORE, AFTER, SIMULTANEOUSLY have to be understood.

H. for *operational* UoD *aspects*

a. simple, complex, these are action and activity respectively. A further modeling notion for operational aspects could be 'business process', see e.g. [K*95a].

d. extracting (queries) or generating, object generating operations are supported

I. for *relational* UoD *aspects*

A. AGGREGATION

B. GENERALIZATION

C. INHERITANCE: also multiple inheritance is offered.

h. existence or identification dependency: existence dependency may be expressed with MULTIPLICITIES which are the OMT corresponding to cardinalities. Identification dependency may be expressed with QUALIFICATION.

2. *defined sub models (i.e. coherent modeling subsystems)*

A. *their scope with respect to the universe of discourse*

STATIC - DYNAMIC - FUNCTIONAL sub model. The static sub model describes structural UoD aspects which don't change over system evolution. The dynamic sub model describes the communicational behavior. The functional sub model describes the input/output behavior of the UoD.

B. interdependencies between sub models

The three sub models are heavily interdependent.

C. compatibility of sub models

They are compatible.

D. integration of sub models

The sub models in general are not explicitly integrated. They are just virtually integrated by the idea that each of the sub models gives a view at a real or at a software system.

3. *independence of modeling notions*

A. selectivity of notions w.r.t UoD aspects

It is relatively clear which modeling notions are to be applied to given UoD aspects. Of course a certain semantic relativism is inherent to OMT.

B. overlapping semantics

Certain UoD aspects can be modeled with in the static sub model as well as within the dynamic sub model. The same holds true for the connection between the functional and the dynamic model (see [K*93]).

D. practicability for modeling

The system of modeling notions is practicable because it offers powerful abstraction concepts and is relatively closed with respect to them, i.e. abstraction concepts may be applied to modeling notions.

4. *Adequacy of the modeling system with respect to the conception*

The modeling system is not complete with respect to the conception, i.e. there is no modeling notion for handling time. But the notions provided are adequate.

5. *quantitative aspects*

none

A. *correspondence between representation concepts and modeling notions*

a. *completeness of the set of representation concepts*

yes

b. *uniqueness in addressing modeling notions*

Probably not. It seems, e.g. to be the case that some identification dependencies, can be expressed with help of multiplicities or qualifiers. Another example is the placement of attributes of relationships.

c. *complexity of the representation concepts*

Generally no, however nested state diagrams (with concurrency) tend to become complex

d. *minimality of representation concepts*

6. *illustration of modeling notions by examples*

A. *number of examples*

sufficient in the accessible literature

B. *illustrated modeling notions*

all modeling notions appear within some examples, however, for to illustrate the notions of the dynamic model mostly technical and not information system oriented examples are given.

7. *sets of representation concepts; per set:*

yes

B. *extendability of representation concepts*

is not recommended by the method authors, but also not prohibited.

C. *practicability of representation concepts*

The representation concepts are practicable.

D. *novelty of the representation concepts*

a. *similarity with well known representation concepts*

Similarities are to the ER model, to structured design notation and to harel state charts.

b. *necessary learning effort*

low

8. *quality criteria*

A. local quality criteria with respect to sub model design documents (sub schema)

There are some hints concerning how to develop the analysis model and what has to be observed in analysis (see [Ru91 p. 148f]).

III. ***Methodology***

1. *proceeding model*

A. *covered parts of the application domain*

all.

B. *covered life cycle phases*

all. But the life cycle model is not complete. Maintenance is missing.

C. completeness with respect to the defined sub models

all.

D. goals

a. intention: OMT does not explicitly mention goals for the conceptual sphere. Implicitly it -as an object oriented method- more or less helps in doing reuse, exploiting UoD inherent parallelism and increasing maintenance although the latter is not part of the development life cycle.

Because of the powerful modeling system and the adjusted representation concepts OMT helps in coping with complexity.

The analyst is encouraged together with the customer to do a review of analysis documents in order to validate them.

E. *goal realization support*

does not apply

F. *goal selection support*

does not apply

G. flexibility of schema-modification

a. restrictions:

only a single integrity condition has to be observed, name and structure conflicts have to be avoided.

b. articulation of sub schema-interdependencies: Such interdependencies very rarely can be made explicit because there are no notions for that purpose.

c. support in handling sub schema-interdependencies:

the only support is via the global EVENT FLOW DIAGRAMS. It allows to check whether the messages appearing in the dynamic models may be understood by the requested objects, i.e. whether they have the necessary methods.

2. *Design primitives and configuration rules*

The method does not offer any design primitives. Configuration rules are restricted to the modeling notions offered.

3. *means for design product analysis, e.g., reachability analysis, invariants*

None.

4. *method style support*

A. for the use of modeling notions, techniques or procedures in special situations

a lot of good hints.

5. *applicability*

A. appropriate support for every process model step

yes.

B. *feasibility of proposed process model steps*

yes.

C. transition between process model steps

a. definedness of transitions

yes.

b. order of proposed steps

yes.

D. conclusiveness of process model steps

a. intuitive

yes

b. reasonable

yes.

c. convincing

yes

E. *adequacy with respect to the conception*

yes.

F. learnability of the process model

easy.

6. *management*

A. single team work support

no.

B. multiple team work support

no.

C. adjustment to conception, modeling notions and methodology

does not apply.

IV. *Documentation*

1. *references*

A. publicly available definition of the OOMM

yes [Ru91].

B. completeness

yes.

C. *availability of (completely) documented case studies*

For case studies see various issues of the 'Journal of Object Oriented Programming' and e.g. [K*93].

D. reports of experience

yes.

E. presence in practice

yes, OMT seems to be one of the most applied OOMM.

F. others

Within the Journal of Object Oriented Programming; there often is published material concerning OMT.

2. *education*

A. textbook

yes [Ru91].

3. *clearness*

A. distinction from other OOMM in literature

yes.

B. tolerance against variations of conception, methodology and process model is given.