Published in: Schader, M.; Korthaus, A.: The Unified Modeling Language. Technical Aspects and Applications. Heidelberg: Physica 1998, S. 14-31

Object-Oriented Modelling Languages: State of the Art and Open Research Questions

Ulrich Frank

"Language is my instrument, but at the same time my problem."

Maturana

Abstract

Object-oriented modeling is used in a growing number of commercial software development projects. But the plethora of approaches and corresponding CASE tools still prevents corporate users from migrating to object-oriented software development methods. Against this background the recent efforts of the Object Management Group (OMG) to standardize object-oriented modeling languages seem to promise substantial benefits: Not only will a standard allow to transfer a model from one CASE tool to another, it will also protect investment in training. However, at the same time it is questionable whether the state of the art in object-oriented modeling is mature enough to allow for standardization. In order to answer this question, we will briefly describe the proposals submitted to the OMG in January 1997. We will then show that there are still essential problems in designing modeling languages which have not been addressed yet.

1. Introduction

During the last decade, object-oriented software development has been adopted in the academic world with remarkable enthusiasm. This is different with corporate software development: Although there is a tremendous marketing push caused by vendors, consultants, and specialist journals, many companies are hesitating to introduce object-oriented methods[1]. This is for comprehensible reasons: At present time, there are still serious inhibitors of object-oriented software development to overcome. There is lack of *mature technology*. While there are object-oriented programming languages that come with reliable compilers - sometimes embedded in rather convenient environments, today's object-oriented database management systems are usually not suited to replace relational database management systems within corporate information systems. Furthermore there is *lack of competence*. In order to exploit the benefits offered by the object-oriented paradigm, and to avoid its pitfalls, it is necessary for the developers to gain a deep understanding of the essential concepts. It is certainly not too daring to assume that most software developers do not have these skills today. From a managerial point of view it is risky to provide for the training that is required to develop these skills: Not only that training is expensive, and its success is hard to predict; furthermore, there is still *lack of standards*. This is the case for object-oriented programming languages and database management systems, as well as for specialized development methods. For this reason protection of investments, both in professional training and in technology (CASE, compilers, etc.), is usual-

1. To our knowledge there is no representative study on an international scale. A recently conducted empirical study ([Sch97]) indicates that less than 10% of the german insurance companies use object-oriented software development methods.

ly not satisfactory.

Although many companies are still hesitating to introduce object-oriented methods and technologies, there is no doubt that the future (that is at least the next ten years) of corporate software development will be more and more object-oriented - simply because there is no alternative paradigm of similar relevance. In other words: The already big market for object-oriented concepts, training, and technologies can still be expected to grow at a fast pace. However, in order to encourage corporate investments, it is not sufficient to develop mature technologies. In addition it is crucial to provide reliable standards that foster interoperability and reusability, and protect investments at the same time. Only recently it became apparent that the various dialects of object-oriented modeling will eventually be replaced by an industry standard modeling language: The "Object Management Group" (OMG) issued a request for proposals for object analysis and design ([OMG96]). The submissions were due at January, 17th, 1997. The review process can be expected not to be completed before the end of 1997. The OMG's request for proposal was one cause for writing this article which will focus on the following questions:

- What are the particular benefits to be expected from standardized modelling languages?
- What is the OMG's intention?
- What are the characteristics of the official proposals to the OMG?
- Is the current state of the art mature enough to allow for standardization?

2. Background of Current Standardization Efforts

While it is no option to standardize a modeling method or even a modeling methodology (see [Fra97]), the standardization of object-oriented modeling languages promises substantial benefits for the development and utilization of software. Standardized modeling languages will allow for better protection of investments in technologies that depend on those languages - like modeling tools and ODBMS. This is the case for investments into training, too. Standardized modeling language will also improve the chance for exchanging models and interoperability in general, or - in other words - for system integration. Furthermore, standardized modeling languages will foster reusability of existing models, for instance of generic or reference models developed for certain problem areas or domains. Standardized modeling languages are also a prerequisite for the emergence of a market for standardized (implemented) classes for corporate information systems, often referred to as "business objects". It is not possible to establish a market for business classes/objects that have been specified/implemented independent from each other: Usually objects, which are to be used in a certain domain, will interact with other objects/classes. Furthermore, it is important to represent business objects in an illustrative way. For these reasons, the vision of providing standardized business objects in the long run requires application level object models together with a corresponding implementation - either as a class library or as a framework (see [LeRo95]). Standardizing application level object models implies the existence of a standardized modeling language.

Against this background it is not surprising that there is a growing demand for the standardization of object-oriented modeling languages in the industry. The OMG has addressed this issue in 1992 already by launching the "Object Analysis and Design Special Interest Group" ([KaCh92]). But this group did not succeed to meet its central objective: "To formalise the definitions of the concepts used for analysis and design" ([KaCh92], p. 12). Apparently for this failure, the OMG issued a request for proposals for object analysis and design in 1996 ([OMG96]). The aim of the initiative is to standardize "the interfaces and semantics needed to support the creation and manipulation of a core set of OA&D models that define the structure, meaning, and behavior of object applications" ([OMG96], p. 3). In particular, the standard should support the creation of "static models", "dynamic models", "usage models", and "architectural models". It seems that the OMG is primarily interested in CASE tool interoperability. Nevertheless, the OMG's statements about the purpose of the "OA&D facility" leave a number of questions:

What is the precise objective of the intended standard?

The explicit reference to tool interoperability indicates that the standard is to describe metamodels for those models managed by CASE tools. However, the OMG stresses the conceptual level as well: "An object analysis and design model (hereafter called simply model) is defined as an object that represents the semantics of an analysis or design of a system. A model is used to define and specify the semantics of interest in a particular aspect of the domain." ([OMG96], p. 13) Furthermore, the OMG requires notations which enable "a core set of OA&D diagrams, each of which describes a model, to be communicated between people without semantic loss." ([OMG96], p. 4). Note that tool interoperability and modeling languages as an instrument for system development require different specifications. While a model that is managed by a tool should be suited to incorporate the semantics defined in the corresponding modeling languages, it will also include additional information, like for versioning, user management, etc. Defining a language for communication between people on the other hand requires to take into account human perception and conceptualization.

What is the scope of the (meta) models to be specified?

While the request for proposals provides examples for a number of models that might be part of the standard ("class models", "instance models", "state-transition models", "use-case models", etc.), it does not tell the level of semantics they should incorporate. In other words: What are the stages of the software life-cycle to be covered - in part, or completely - by the modeling languages to be defined? There is evidence that the OMG does not plan a rigid standard anyway:

> "The OA&D facility should not constrain the processes and techniques that OA&D methods use to extract the information needed to determine the semantic content of OA&D models, to ensure the integrity and validity of the content of those models, or to evolve or optimize those models. Nor should the facility constrain OA&D methods from introducing models in addition to the core set." ([OMG96], p. 4)

How does the intended standard relate to other standards?

There is no doubt that the standard should be compliant to existing OMG specifications - such as the Interface Definition Language (IDL). However, there are at least two standardization efforts outside the OMG which are concerned with subjects that are obviously related to analysis and design. Firstly, there is the CASE Date Interchange Format (CDIF), defined by the CDIF division within the Electronic Industries Association (EIA). CDIF is dedicated to support the exchange of models managed by CASE tools. CDIF does not specify the metamodel a tool has to use itself. Instead, it is restricted to the external representations that a tool has to support with its interfaces. With the ISO "Information Resource Dictionary System" (IRDS, [ISO90]) and the "Portable Common Tool Environment" (PCTE, [Wak93]), an initiative launched by tool vendors, two major standardization efforts are committed to using the CDIF metamodel ([CDI96]).

Secondly, there are the activities of the Workflow Management Coalition (WfMC). Similar to the OMG the WfMC is an industry consortium. It aims at a set of standards that are to allow for "open" workflow management systems. In order to achieve this goal the WfMC intends to define a set of interfaces for the components to be integrated (such as office application, DBMS, and so called workflow engines) as well as a "Workflow Process Definition Language" (WPDL, [WFM96]). The idea behind this approach is similar to entity relationship modeling and SQL respectively: A workflow type can be modeled using an appropriate modeling language provided by a corresponding tool. The modeling language will then translate into the WPDL which can be interpreted by any workflow engine that has been certified by the WfMC. There is a clear relationship to the mission of the OMG's "Object Analysis&Design Facility": In order to model a workflow, you need to describe the information required and produced within that workflow - for instance by referring to an object model. While both, the EIA ([CDI96]) and the WfMC ([And96], p. 21) refer to the OMG, to our knowledge the OMG itself does not explain its relationship to those approaches, nor does it even mention them.

3. Object-Oriented Modelling Languages: Is there a State of the Art?

Standardizing technologies or concepts requires a mature level of corresponding research activities. With object-oriented modeling, it is impossible to identify a unique scientific community that is dedicated to this subject. This is for a number of reasons. Firstly, object-oriented concepts have various roots, including programming languages, database design, and artificial intelligence. Secondly, there are contributions from authors with either a commercial or a more academic background. Therefore, the selection we made, is a compromise: We will primarily focus on what you could call the "dominating" state of the art. For this purpose, we will have a look at the languages submitted to the OMG early this year. By the time of writing this, some of the proposals have apparently been withdrawn. Nevertheless, we will consider them briefly, since they give an impression of the current state of the art.

3.1 The Proposals to the OMG

Until the deadline at January 17th the following six proposals had been submitted to the OMG:

Company/Consortium	References	Extent (pages)
Taskon; Reich Technologies; Humans and Technology	[Tas97]	101
IBM; ObjecTime Limited	[IBM97]	196
Softeam	[Sof97]	37
Platinum Technology	[Pla97]	318
Ptech	[Celb97]	58
Rational Software; Microsoft; Hewlett-Packard; Oracle; Texas Instruments; MCI Systemhouse; Unisys; ICON Computing; IntelliCorp.	[Rat97a] ... [Rat97d]	>550

Taskon; Reich Technologies; Humans and Technology

This submission does not claim to describe a complete set of object-oriented modeling languages. Instead, its authors intend to provide an "extension of the UML, and the OML object models" ([Tas97], p. 1). It originates from a number of previous approaches (like [Ree95], [WiWi90]). Its main emphasis is on three additional basic concepts: *role model*, *role*, and *port*. Roles emphasize another level of abstraction than classes. A role represents a particular real world object's role within a certain activity. A role model describes the interactions of roles within an activity. Those interactions are invocations of operations in other roles/objects. Apparently role models are to be used mainly during analysis, in order to record systematically the features of classes within a corresponding object model ([Tas97], p. 15). A port is described as "an abstraction on a variable, permitting the object represented by the adjoining role to execute operations in the object represented by OtherRole." ([Tas97], p. 50) This concept is not sufficiently explained. We assume that it is similar to - although not equivalent - to delegation, where an object may transparently access operations of its role object ([FrHa97]). The metamodel itself is defined using the IDL standardized by the OMG. Although this has been encouraged by the OMG, there is no doubt that IDL is not suited to serve as a modeling or specification language, since it lacks relevant concepts (for instance: it is not possible to express cardinalities or more specific constraints).

There is one other suggestion within this proposal that we would like to mention: "system inheritance" is to provide for specializing from "the overall system structure and behavior properties" ([Tas97], p. 14). As the authors state, such a concept would certainly be helpful for designing reusable frameworks. That, however, would require a precise definition of this type of inheritance - especially how to modify and enhance an inherited system. We could not find such a definition within the proposal. Although Reich claims to be a tool vendor, it seems that this consortium is focusing on the applications of modeling languages rather than on metamodels which would promote tool interoperability. The authors have rather elaborated ideas about the particular models to be used within a software development project. They suggest not less than nine partial views (like "area of concern view", "stimulus-response view", "role list view", "interface view", etc.) together with corresponding modeling notations. The models are described in an informal way ([Tas97], pp. 88).

IBM; ObjecTime Limited

This proposal, submitted by IBM and ObjecTime, covers what may be called the "full range" of common object-oriented modeling: static properties, behavior, and dynamics. Nevertheless, the authors apparently do not intend to define a complete set of languages for object-oriented modeling. While they completely abstract from modeling notations, they put strong emphasis on language semantics and pragmatics. It is characteristic for this outstanding submission that it starts with analyzing the requirements related to modeling languages in general. The authors make the assumption that models may serve a wide range of purposes, which can hardly be determined in advance. To illustrate this point, they distinguish between "general purpose modeling languages", "domain specific modeling languages", and even "application specific modeling languages". Furthermore, they state that modeling languages should take into account concepts which are appropriate for a particular purpose (and particular people) rather than reconstructing programming languages. On the other hand they argue that modeling applied to software development at some point requires formal definitions of the modeling languages. From these assumptions they conclude that there is "need for extensibility", "for language and method independence", and "for formality and interoperability" ([IBM97], pp. 12).

The consortium presents a "Core Meta-Model (CMM)" which is the foundation for the models that can be built within this approach. The CMM contains generic concepts which allow to express static, dynamic, and interaction semantics. It is characteristic for the level of abstraction suggested by this proposal that the CMM does not explicitly contain core concepts of object-oriented modeling such as classes or objects. Instead the CMM provides abstractions which may be specialized further. One of the core abstractions is called "specification". Specification is an abstraction of concepts like type and class: "Specifications representing pure interfaces will have all Features unimplemented, those representing concrete classes will have all Features implemented, and further possibilities exist in between." ([IBM97], p. 37) Compared to other submissions, there is a stronger emphasis on formalization. For this purpose the authors introduce and define a formal language called "Object Constraint Language" (OCL) that is used to specify the CMM. It also helps with the specification of additional modeling languages, since the OCL in general is used to specify invariants within model schemes. Additionally, each "model scheme may define a Grammar for each type of Expression" ([IBM97], p. 72) - like invariants or conditions. The proposal includes a few examples of how to specify a model scheme. Smalltalk "design models", for instance, can be introduced by a scheme which includes - among other things - invariants to constrain generalization to single inheritance, or to express that every element within a class has exactly one name. The ability to introduce specialized languages which are defined by concepts in more general languages is a promising approach - last but not least to support the idea of standardized business object models. Notice, however, that extensibility - like the definition of new modeling languages with existing meta concepts - is not sufficient to allow for a straightforward interchange of models expressed in such a new language: Standardized meta concepts do not imply that the special concepts of a particular modeling language will be standardized, too. For instance: If you define a language scheme for a business modeling language, you may want to introduce a concept like "Organisational Unit". It is probably very hard to define such a concept in a way that everybody is willing to accept. Nevertheless a standardized metalanguage allows to determine precisely how a specialized concept was defined - even if you do not agree with it.

Although the authors point to the fact that modeling languages should suit a particular purpose, providing particular modeling languages is not their main concern. Instead they refer to approaches such as the UML or the OML ([IBM97], pp. 131) which they consider as possible applications of their specification. From our point of view, this is a very ambitious, yet substantial approach. To briefly summarize our impression, we think that the strength of this approach marks a possible weakness at the same time. Taking into account the realistic assumptions on requirements for modeling languages, it is certainly a good idea to provide for a high degree of extensibility: Thereby developers (and users) of specific modeling languages will not be restricted to concepts which do not fit their needs. However, by allowing for almost arbitrary specific modeling languages, standardization does not happen on the level of those languages.

Softeam

This submission is based on the "Class Relation method" which was introduced in 1989, and revised later ([Des94]). Softeam's proposal is certainly not as complete as for instance the previous one. In fact, it seems that the authors primarily intend to introduce a few concepts as alternatives to corresponding concepts of the UML. In particular, they suggest an "object flow model" that is to provide a representation of operations behavior. Most remarkable, from our point of view, is the introduction of a special state diagram that is based on the "Hygraph" principle suggested by Harel. It is motivated by the fact that the state charts often used in object-oriented modeling neither allow for a well defined integration with object models, nor do they allow to express generalization/specialization hierarchies. The suggested state diagrams can be decomposed into two different categories ([Sof97], pp. 7). "Control state diagrams" focus on the usage of a class, describing what is to be expected from an instance of this class from an external point of view. The authors state - referring to [Des94] - that control state diagrams can be completely translated into pre- and postconditions. "Trigger state diagrams" represent a particular implementation of a control state diagram. In other words: They describe the actual dynamics a class has to implement in order to satisfy the requirements specified in a corresponding control state diagram. Figure 7 gives an overview of the control state diagram metamodel.

The specialized state charts suggested by Softeam promise both a better integration with other partial models, and to take advantage of generalization/specialization. Notice, however, that the corresponding concepts are not defined in the proposal itself - which comprises only about 30 pages. Instead it is referred to [Des94].

Platinum Technology

Platinum used to be a provider of tools for the development and maintenance of relational DBMS. With the acquisition of Protosoft, the developer of an object-oriented modeling tool called "Paradigm Plus", Platinum became one of the major vendors in this market. "Paradigm Plus" is claimed to support various popular modeling approaches, such as OMT ([Rum91]), OOSE ([JaCh92]), and Booch ([Boo94)]. Against this background it is not surprising that Platinum's submission puts strong emphasis on tool interoperability. The proposal includes seven modeling languages, each of which is assigned to a so called "subject area". Each subject area represents a certain view on a system - such as object models, dynamic models, architecture models, etc. The subject areas are defined by seven corresponding metamodels which are tightly coupled. The concepts shared by all seven metamodels are assigned to an additional subject area called "Foundation and Common Subject Areas", which in turn is defined in a corresponding metamodel.

Notice that the COMMA ("Common Object-oriented Methodology Metamodel Architecture") metamodel, proposed by the OPEN consortium ([FiHe96), "had a major influence" ([Pla97], p. 13) on those metamodels. Furthermore the metamodels are defined within a common "meta meta model". It is based on the CDIF meta metamodel ([EIA93], in other words: it uses the concepts defined in this reference model). Therefore the metamodels can be exchanged between all tools which are capable of using the CDIF interchange standard formats.

The meta metamodel provides a few core concepts which are "totally immutable" ([Pla97], p. 16). "Extensibility" of subject areas can be accomplished by modifying the corresponding metamodels. Notice, however, that extensibility is restricted to the expressive power of the concepts defined in the meta-metamodel. Since those concepts are essentially restricted to data structures it seems that there is no way to specify constraints that go beyond the description provided by such structures - as it is possible in the metamodel suggested by IBM and ObjecTeam.

The submission includes an extensive description of the meta-metamodel as well as of the metamodels (about 2/3 of the whole document). By referring to an existing standard for exchanging information between CASE tools (CDIF), it is probably of special value for tool providers. However, from our point of view the proposal neglects to consider the requirements of conceptual modeling: A modeling language is not only thought to define models for the purpose of exchanging them between programs. Instead, it is also a tool to support to an intellectual endeavor. In other words: It includes pragmatic aspects as well. It seems that these aspects are not Platinum’s primary concern - on the contrary: In the past Platinum has gained its reputation as a tool vendor by supporting many modeling languages. In order to hold on to this tradition, the proposal is restricted to metamodels which can be used to define a set of particular modeling languages. To emphasize this point of view, Platinum does not provide a specific notation. Instead it is argued that the proposed specifications "are fully compatible with the current state of the art in object modeling notations" - a statement that is illustrated by small examples. They refer to other approaches which put more emphasis on notation ([Pla97], pp. 305). Although the "current state of the art" is hard to identify, such a statement is rather daring: Notations only make sense with associated concepts. We doubt that the meta-metamodel allows to express the semantics of any concept that may be useful within object-oriented modeling. For instance: How would you express the specific semantics of delegation (see [FrHa97]).

Ptech

The authors of this proposal first make a few assumptions on the purpose of modeling languages. They emphasize both, the need to offer a medium to foster effective communication between humans with different professional backgrounds, and the need for tool interoperability ([CeIb97], pp. 1). They provide metamodels for five types of models: "structural models", "architectural models", "behavioral models", "distributed processing models", and "usage models". It is remarkable that the authors seem to be very much inspired by relational theory. In order to define the concepts they suggest for object-oriented modeling, they often refer to notions such as "set", "domain", "range", or "relation" - for instance: "The associations that implement the relation are named customer and order." ([CeIb97], pp. 3).

To some degree this proposal benefits from the precision of relational theory. It lacks, however, a specific object-oriented point of view - in other words: The authors seem not to have overcome the separation of function and data. For instance, they speak of "... an operation that can be performed on instances of a class." ([CeIb97], p. 14) Although the authors state that their metamodels "guarantee that all Ptech models are logically consistent." ([CeIb97], p. 1), the documentation of the metamodels does not seem to be complete. While the metamodels are said to be extensible ([CeIb97], p. 1), it is not demonstrated how an extension can be accomplished. Furthermore, there is no language included that would allow to specify additional constraints. From our point of view, the most remarkable aspect of this proposal is the background of the Ptech method: It has been used for "understanding, analyzing, capturing, validating, and documenting business processes and systems." ([CeIb97], p. 1) It is surprising that the authors did not add specific requirements for object-oriented modeling languages to be applied in the area of process modeling.

Rational et al.

This submission is backed by a rather impressive consortium including Rational Software, Microsoft, Hewlett-Packard, Oracle, Texas Instruments, and Unisys. The proposed "Unified Modeling Language" (UML) has mainly been developed by Rational Software and has its roots in three well known object-oriented modeling approaches ([Boo94], [JaCh92], [Rum91]). The material proposed by the consortium is relatively extensive: It includes 13 documents which cover more than 550 pages. Furthermore, the background of the UML is explained in various textbooks ([Boo94], [Jac94], [JaCh92], [Rum91], [Der95], [Whi94]), and in numerous articles. But it is certainly not the sheer volume, why the UML is regarded as a reference already - often without discussing its qualities. Even within the competing proposals to the OMG the UML seems to have gained an outstanding position. This may be due to the fact that it is associated with three protagonists of the object-oriented modeling arena. Furthermore, it seems that most of the other consortia have accepted the relevance of this proposal, since they describe how their metalanguages, or metamodels respectively, would allow to specify parts of the UML (for instance [IBM97], [CeIb97]) - or they restrict their efforts to providing extensions to the UML (like [Sof97]).

Compared to the previous approaches, the UML is based on, the language description submitted to the OMG is much more comprehensive (fig. 12 gives an impression). Although some concepts of the original approaches, such as data flow diagrams, which are part of OMT, are not supported any more, the UML almost appears like a superset of the languages it originates from. It supports nine types of diagrams. Static aspects can be rendered on various levels of abstraction. In addition to *class* or *object diagrams*, *component diagrams* serve to map existing software components and their interrelationships. *Deployment diagrams* can be used to render dependencies between runtime systems across various platforms. Collaboration diagrams allow to express message flows between objects. *Sequence diagrams*, *state diagrams*, and *activity diagrams* serve to render various dynamic aspects. Furthermore, the UML provides *use case diagrams* as suggested in [JaCh92]. While the variety that comes with those diagrams improves the chances to find a diagram type which is appropriate for a particular view, it can be a burden at the same time, since some of the diagrams (like sequence diagrams, state diagrams, and activity diagrams) are overlapping in a subtle way. For a detailed evaluation of the UML see [FrPr97]. The 1.1 version of the language, released in July 1997, has adopted the OCL from the IBM/ObjecTime proposal. In addition to the UML and natural language, the OCL is used to specify the abstract syntax and semantics of the UML ([Rat97d]).

Without any doubt, the UML marks a clear progress compared to its direct predecessors. Nevertheless, it is not completely convincing in the end. It seems questionable whether the extent and variety of concepts provided with the UML will not confuse the prospective users. It is by no means easy to learn and use.

3.2 Resume

Our brief analysis was mainly motivated by the following question: Do the proposals give the impression of a converging and mature field? Considering the different backgrounds and goals of the various contributors, it is certainly not appropriate to speak of a coherent state of the art. Furthermore, we do not think that the submissions are a representative selection of object-oriented modeling languages in general: There are numerous other approaches that contribute to the state of the art as well - from various perspectives (for instance [Mey97], [Coa95], [GoRu95]). One of them deserves particular attention: The OPEN Modeling Language (OML) which has evolved from a joint research effort called OPEN ("Object-oriented Process, Environment, and Notation"). Different from the companies that submitted their proposals to the OMG, OPEN is an initiative launched solely within an academic context. OPEN has been developed further from a number of previous approaches, like [Col89], [Des94], [Fir92], [Gra91], [HeEd94], [Hen92]. As it is indicated by its name, the members of the initiative clear-ly want the OPEN specifications to become a standard ([FiHe96], p. 4). Nevertheless, they did not submit the already existing specifications, because they were not able to fulfill some requirements defined by the OMG: A submitting organization has to be a "contributing member" of the OMG. Additionally, it has to provide a statement about its willingness to make the proposed "technology ... commercially available within 12 months of adoption" ([OMG96], p. 24) - whatever that means for modeling languages.

Independent from the question how well the OML compares against other approaches (for a comparison of the UML and the OML see [FrPr97]), the regulations imposed by the OMG have rather disturbing consequences: They exclude many potential contributors, especially those with academic intentions, no matter how valuable their proposals might be. Such a procedure does not only jeopardize the quality of the standard to be established. In addition, it may also hinder the acceptance of the standard by those who were not able to express their ideas.

4. Open Research Questions

Assuming that modeling languages are of pivotal importance for the way people perceive and conceptualize real world domains, and how they design software, the design of a general modeling language is a challenging task. This is for various reasons. Firstly, there is *variety* - both in domains and in software systems. For instance, there are certainly essential differences between traffic control systems, vending machines, or marketing information systems. This is the case for the corresponding terminologies as well as for abstractions used to represent system features. Additionally, there is an immense variety of people who use a modeling language - both as readers and writers. Since it can be expected that this variety is accompanied by a wide range of individual perceptions, conceptualizations, and preferences, it is almost impossible to design a language that fits the needs of all potential users. Secondly, there are *trade-offs*. A modeling language should be easy to use. Its notation should be intuitive, which implies that it corresponds to the conceptualizations its users prefer. At the same time it should support the design and implementation of software. At some point, that requires to introduce formal concepts which are suited to be mapped to implementation languages. Furthermore, there is the well known trade-off between quality, cost, and time. While it is difficult to tell how a modeling language relates to this trade-off, we cannot assume that quality, cost, and time are independent of the modeling languages used within a project. Thirdly there is *arbitrariness*. Like any artificial language, a modeling language should be designed to fit its purpose. However, it cannot be deduced logically from this purpose. This thought alone implies that arbitrary decisions can hardly be avoided. Furthermore, dealing with trade-offs will usually require compromises that will, to a certain degree, reflect individual preferences rather than objective reasons.

Comparing the current state of the art of object-oriented modeling against those challenges reveals a remarkable lack of knowledge. Regarding a modeling language as an instrument for software development would suggest to start with a thorough requirements analysis which would include a number of questions, for instance:

- What are the purposes the language is to be used for?
- Who are the prospective users of the modelling language?
- What are the concepts a language should provide in order to fit their cognitive styles?
- How does an intuitive notation look like?

There is no doubt that answering these questions requires some sort of empirical research. The authors of object-oriented modeling methods or languages sometimes emphasize the experience they have gathered with applying their approach in practice (for instance [FiHe96], p. 9, or [CeIb97], p. 1). Vendors and consultants have certainly received some kind of feedback from their customers. However, it is remarkable that up to now there has been no sophisticated study of how people perceive and deal with concepts and notations of object-oriented modeling languages (at least we do not know of any). There have been a few studies on the use of entity relationship (ER) modeling ([Hit95], [GoSt90]). They indicate that ER models are not intuitive at all for many people. Our experience, as well as the similarity between object models and ER models suggest that this is also the case for object-oriented modeling. Approaches like the UML or the OML, which offer a wide range of different modeling languages, can be expected to be even less intuitive for many prospective users. Furthermore some concepts featured by those languages, such as use cases, are difficult to understand. For this reason, applying them can easily result in bad design (for a detailed analysis of pitfalls see [Ber97]). In other words: Without substantial knowledge about the way how people perceive and apply modeling concepts, it is hard to tell whether those concepts contribute to software quality - or to "disaster" ([Ber97]).

Beside the lack of research on user perceptions and preferences, there is a wide range of domains and purposes that suggest the use of modeling languages. To give a few examples: business process (re-) design, workflow management, organizational design, enterprise modeling, design of document management systems, design of integrated circuits, design in the field of computer telephony integration (CTI). During the last years a lot of special purpose modeling approaches have emerged in those fields (for instance: [Fra97], [Fra94], [IsSt95], [Oul95], [Sch95], [Tay95]). While they usually require specialized concepts, many of them are closely interrelated with object-oriented modeling. For instance: Modeling a business process usually requires to refer to information specified within an object model. In order to foster integration, it is helpful to regard documents as objects which require special concepts. None of the modeling languages discussed above includes special concepts to cover one of those domains. Furthermore, one has to take into account that some of those fields are still in a virgin state, although they are of increasing relevance - like business process modeling or enterprise modeling. Against the background of standardization, it is important to notice that additional requirements for modeling languages can be expected from various evolving areas.

While reusability has been a research topic for long, there are two recent approaches that have gained remarkable attention: design patterns ([GaHe95]) and frameworks ([LeRo95]). They promise to deliver reusable artifacts which are rather flexible and/or convenient to use. Nevertheless, there is only little experience about how to integrate them in the process of software development. This is especially the case for design patterns since they are - by definition - less formal in nature than frameworks. It seems characteristic that, despite these problems, some of the approaches discussed in 4.1, and 4.2 already feature concepts to describe design patterns (for instance: [Rat97a], [FiHe96]) - however, without specifying them in an adequate way (for instance: [Rat97a] defines a design pattern as "a template collaboration", p. 66).

In order to specify/standardize modeling languages you need metamodels or metalanguages. Among other things, they should allow for convenient and safe extensions of the corresponding object level languages. At the same time, a metalanguage perspective, as emphasized in [IBM97] or [Pla97], allows to abstract widely from some problems occurring on the level of a particular modeling language - like user perceptions or preferences. For this reason it may appear that an approach like the one suggested by [IBM97] provides a satisfactory solution at least for the metalanguage level. However, this is not the case. Within computer science there is a wide range of alternatives. Beside general, well known approaches like predicate calculus, or algebraic specification, there are many special approaches that define representations for metalanguages to be used for CASE tools or Meta CASE tools respectively (for instance: [EbWi96], [KeSm96]). While there are requirements for metalanguages, like completeness, simplicity, and correctness (see [SüEb97], pp. 2), it is still difficult to compare them in an objective sense: Similar to modeling languages to be used on an object level, metalanguages are artificial. Therefore they are necessarily arbitrary to some degree. This is an important aspect for another reason as well: Even with metalanguages you cannot completely neglect cognitive styles of prospective users: An extensible metalanguage/metamodel is also used by people who have their own ideas about language concepts and notation - although the group of people who work on a meta level is much smaller than the group of those who use a modeling language on an object level.

The current state of the art is not only lacking knowledge which would be required for the specification of modeling languages. In addition, it is not evident which scientific discipline is responsible for filling those gaps. The design of artificial languages as well as the analysis of their use is an essential topic of linguistics. The interaction between language and cognition is subject of cognitive psychology. The specification of languages to be used for software development is part of computer science. Finally, the definition of concepts suited to describe and analyze certain real world domains, like business firms, is subject of disciplines like management science or organization studies. Considering both, the complexity of the open research questions and the fact that there is no single discipline to cover those questions in a satisfactory way, indicates that it will take considerable time before we are able to speak of a mature state of the art in modeling languages.

5. Concluding Remarks

Our overview of the current state of the art has shown that object-oriented modeling is a still an evolving field with relevant problems and challenges still to be overcome. In addition, the field of object-oriented modeling lacks a common focus: There is work on metamodels with more or less formal rigor, while other approaches focus on the application of modeling languages. There is also variety in the backgrounds of people working on various aspects of object-oriented modeling; to name a few: programming languages, database design, artificial intelligence - with motives ranging from purely academic to chiefly commercial.

From an academic point of view software development in general, conceptual modeling in particular has not reached a level of maturity that would recommend to freeze a certain paradigm by standardizing corresponding modeling languages. Although there has been considerable progress during the last years, we still agree with the authors of an open letter to the OMG ([MeSh93]) who, in 1993, advised emphatically against a standardization of object-oriented modeling methods (which includes the standardization of modeling languages)[1]:

"Standardization of this rapidly developing technology will be out of date almost immediately.

1. In the meantime some of them, like Booch, Rumbaugh, or Wirfs-Brock, have apparently changed their mind, since they participated in the preparation of proposals to the OMG.

Not only is standardization futile, but, to the extent that it succeeds, positively dangerous. Standardization will discourage the innovation required to advance and mature the methods."

However, things look different from an economic point of view: There is no doubt that standards are of crucial importance for establishing integrated information systems. Furthermore they foster reusability, help to protect investments, and decrease transaction cost. Hence, in the end it comes down to the trade-off between the benefits of standardized representations and the pitfalls of investing into premature concepts. This decision is beyond the capabilities of IS research. It finally happens in market-like settings: If enough players see a chance for an equilibrium between the incentives and the cost of standardization, a standard may be established.

References

[And96] Anderson, M.J.: Draft Workflow Standard - Interoperability. Abstract Specification WFMC-TC-1012, 3-June, 1996

[Ber97] Berard, E.V.: Be Careful With "Use Cases". 1997. Obtained via http://www.toa.com/pub/html/use_case.html

[Boo94] Booch, G.: Object-oriented Design with Applications. 2nd ed., RedwoodCity, Ca.: Benjamin Cummings 1994

[CDI96] CDIF: Harmonization of CDIF with other Standards Bodies, 96-07-26, 1996. Obtained via http://www.cdif.org/intro.html

[CeIb97] Cerrato, J.; Ibrahim, H.: The Ptech Method for Object-Oriented Development Version 1.0, 1997. Obtained via http://www.omg.org/library/schedule/AD_RFP1.html

[Coa95] Coad, P.: Object Models: Strategies, Patterns, and Applications. Englewood Cliffs, NJ: Prentice Hall 1995

[Col89] Colbert, E.: The Object-Oriented Software Development Method: A Practical Approach to Object-Oriented Development. In: Proceedings of TRI-Ada ’89 - Ada Technology in Context: Application, Development, and Deployment. New York: ACM Press 1989, pp. 400-415

[Der95] Derr, K.W.: Applying OMT: A practical step-by-step guide to using the object modeling technique. New York: SIGS Books 1995

[Des94] Desfray, P.: Object Engineering - The Fourth Dimension. Reading, Mass.: Addison-Wesley 1994

[EbWi96] Ebert, J.; Winter, A.; Dahm, P.; Franzke, A.; Süttenbach, R.: Graph Based Modeling and Implementation with EER/GRAL. Thalheim, B. (Ed.): Proceedings of the 15th International Conference on Conceptual Modeling. Berlin et al.: Springer 1996, pp. 163-178

[EIA93] CDIF Framework for Modeling and Extensibility, EIA/IS-107, Electronic Industries Association, November 1993

[Fir92] Firesmith, D.: Object-oriented requirements analysis and logical design. Chichester 1992

[FiHe96] Firesmith, D.; Henderson-Sellers, B.; Graham, I.; Page-Jones, M.: OPEN Modeling Language (OML). Reference Manual. Version 1.0. 8 December 1996. Obtained via http://www.csse.swin.edu.au/OPEN/comn.html

[FrHa97] Frank, U.; Halter, S.: Enhancing Object-Oriented Software Development with Delegation. Arbeitsberichte des Instituts für Wirtschaftsinformatk, Nr. 2, Koblenz 1997

[FrPr97] Frank, U.; Prasse, M.: Ein Bezugsrahmen zur Beurteilung objektorientierter Modellierungssprachen - veranschaulicht am Beispiel von OML und UML. Arbeitsberichte des Instituts für Wirtschaftsinformatk, Nr. 6, Koblenz 1997

[Fra97] Frank, U.: Enhancing Object-Oriented Modeling with Concepts to Integrate Electronic Documents. In: Proceedings of the 30th HICSS, vol. VI, ed. by R. H. Sprague, Los Alamitos, Ca.: IEEE Computer Society Press 1997, pp. 127-136

[GaHe95] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Reading/Mass. et al.: Addison-Wesley 1995

[GoRu92] Goldberg, A.; Rubin, K.S.: Object Behaviour Analysis. In: Communications of the ACM. Vol. 35, No. 9, 1992, pp. 48-62

[GoSt90] Goldstein, R.C.; Storey, V.C.: Some Findings on the Intuitiveness of Entity Relationship Constructs. In: Lochovsky, F.H. (Ed.): Entity Relationship Approach to Database Design and Query. Amsterdam: Elsevier 1990

[Gra91] Graham, I.: Object-Oriented Methods. Wokingham et al.: Addison-Wesley 1991

[HeEd94] Henderson-Sellers, B.; Edwards, J.M.: Book Two of Object-Oriented Knowledge: The Working Object. Object-Oriented SoftwareEngineering: Methods and Management. Sidney et al.: Prentice Hall 1994

[Hen92] Henderson-Sellers, B.: A Book of Object-Oriented Knowledge: Object-Oriented Analysis, Design and Implementation. A new Approach to Software Engineering. Englewood Cliffs, NJ: Prentice Hall 1992

[Hit95] Hitchman, S.: Practitioner Perceptions on the Use of some Semantic Concepts in the Entity Relationship Model In: European Journal of Information Systems, Vol. 4, 1995, pp. 31-40

[IBM97] IBM; ObjecTime Limited: OMG OA&D RFP Response Version 1.0. 1997. Obtained via http://www.omg.org/library/schedule/AD_RFP1.html

[ISO90] ISO/IEC1990 IRDS Framework. ISO/IEC-Standard 10027. 1990

[IsSt95] Isakowitz, T., Stohr, E.A., Balasubramanian, P.: RMM: A Methodology for Structured Hypermedia Design. In: Communications of the ACM, Vol. 38, No. 8, 1995, pp. 34-44

[Jac94] Jacobson, I.; Ericsson, M.; Jacobson, A.: The Object Advantage. Business Process Reengineering with Object Technology. Wokingham et al.: Addison-Wesley 1994

[JaCh92] Jacobson, I.; Christerson, M; Jonsson, P; Overgaard, G.: Object-Oriented Engineering. A Use Case Driven Approach. Reading, Mass.: Addison-Wesley 1992

[KaCh92] Kain, J.B.; Christopherson, M. et al.: Object Analysis and Design. OMG Document 92-10-01.PDF, draft 7.0, 1992. Obtained via http://www.omg.org/library/public-doclist.html

[KeSm96] Kelly, S.; Smolander, K.: Evolution and issues in metaCASE. In: Information and Software Technology. Vol. 38 (Special Issue: Method engineering and meta-modelling), No. 4, 1996, pp. 261-266

[LeRo95] Lewis, T.; Rosenstein, L. et al. (Eds.): Object Oriented Application Frameworks. Greenwich, CT: Manning 1995

[MeSh93] Mellor, S.J.; Shlaer, S.; Booch, G.; Rumbaugh, J.; Salmons, J.; Babitsky, T.; Adams, S.; Wirfs-Brock, R.J.: Premature methods standardization considered harmful Open Letter to the Industry In: JOOP, Vol. 6, 1993, No. 4, pp. 8-9

[Mey97] Meyer, B.: Object-Oriented Software Construction. 2nd Ed., Englewood Cliffs: Prentice Hall 1997

[OMG96] Object Management Group: Object Analysis & Design RFP-1, ad/96-05-01, 1996. Obtained via http://www.omg.org/library/public-doclist.html

[Oul95] Ould, M.A.: Business Processes: Business Processes: Modelling and Analysis for Re-Engineering and Improvement. Chichester et al.: Wiley 1995

[Pla97] Platinum: Object Analysis and Design Facility Response to OMG/OA&D RFP-1. Obtained via http://www.omg.org/library/schedule/AD_RFP1.html

[Rat97a] Rational: UML Semantics. Version 1.0, 02-13-97, 1997. Obtained via http://www.rational.com

[Rat97b] Rational: UML Notation Guide. Version 1.0, 02-13-97, 1997. Obtained via http://www.rational.com

[Rat97c] Rational: UML Summary.0, 02-13-97, 1997. Obtained via http://www.rational.com

[Rat97d] Rational: UML Semantics. Version 1.1 alpha R6, 07-21-97, 1997. Obtained via http://www.rational.com

[Ree95] Reenskaug, T.: Working with Objects: The OORAM Software Engineering Method. Englewood Cliffs: Prentice Hall 1995

[Rum91] Rumbaugh, J. et al.: Object-oriented Modelling and Design. Englewood Cliffs, N.J.: Prentice Hall 1991

[Sch97] Schnur, B.: Objektorientierung in Versicherungsunternehmen. Die Branche gibt sich bislang noch zurückhaltend. In: Informatik Spektrum,Vol. 20, No. 1, 1997, pp. 52-53

[Sch95] Schwabe, D., Ross, G.: The Object-Oriented Hypermedia Design Model. In: Communications of the ACM, Vol. 38, No. 8, 1995, pp. 45-48

[Sof97] Softeam: Submission of the specification of Object Analysis & Design Facility OMG RFP response, 1997. Obtained via http://www.omg.org/library/schedule/AD_RFP1.html

[SüEb97] Süttenbach, R.; Ebert, J.: A Booch Metamodel. Fachberichte Informatik, 5/97, Universität Koblenz 1997

[Tas97] Taskon: The OOram Meta-Model - combining role models, interfaces, and classes to support system centric and program centric modeling. A proposal in response to OMG OA&D RFP-1, 1997. Obtained via http://www.omg.org/library/schedule/AD_RFP1.html

[Tay95] Taylor, D.A.: Business Engineering with Object Technology. New York et al.: Wiley 1995

[Wak93] Wakeman, L.: PCTE: The Standard for Open Repositories. Foundation for Software Engineering Environment. New York et al.: Prentice Hall 1993

[WFM96] WfMC (Workgroup 1): Interface 1: Process Definition Interchange WfMC TC-1016, Version 1.0 Beta, May 29, 1996. Obtained via http://www.aiai.ed.ac.uk/WfMC/ 1996

[Whi94] White, I.: Using the Booch Method - A Rational Approach. New York et al.: Benjamin Cummungs 1994

[WiWi90] Wirfs-Brock, R.J.; Wilkerson, B.; Wiener, L.: Designing Object-Oriented Software. Englewood Cliffs, NJ.: Prentice Hall 1990