Will a Dominant Standard for Object-Oriented System Development Emerge ?

Jos Van Hillegersberg and Kuldeep Kumar

Department of Decision- and Information Science, Faculteit Bedrijfskunde, Erasmus University
G1-40, PO Box 1738, 3000 DR Rotterdam, The Netherlands,
Email: {j.hillegersberg, k.kumar}@staf.fbk.eur.nl, Phone: 31-10-4082624

Abstract

The lack of a dominant Object-Oriented (OO) standard has been a hindrance to the OO paradigm's successful adoption on a large scale. Currently, within different versions of OO methodologies, various OO concepts can exist under different names and interpretations. While recently there have been some attempts to standardise OO systems development, until now, no standard has emerged as the dominant standard in practice. The objective of this paper is to assess benefits of a dominant standard, and to investigate strengths and weaknesses of the current approaches towards standardisation. Based on this investigation, we outline how these different standardisation approaches can contribute toward the emergence of a dominant standard. We conclude that a metamodelling approach, combined with argumentation and ontology when necessary, has the best chances to achieve the goal that, to quote Snyder [1], "those applying OO technology will one day speak the same language".

1. Introduction

1.1 The lack of a standard

Several authors have expressed their concern about the chaotic proliferation of object-oriented (OO) systems development concepts and have suggested that OO system development lacks well defined semantics. Constantine [2] observes that "although everybody talks about it, little consensus exists as to exactly what is meant by object orientation". This view is shared by Snyder [1] who notes that; "...the groups involved with object technology lack a shared understanding of the basic concepts and a common vocabulary for discussing them. Even within the language community, multiple terms are often used for the same concept, and the same term is sometimes used with different meanings". Discussing inheritance, Winkler [3] notes that this "...key-concept of OO programming is interpreted quite differently by different groups of the software community". Within the OO database community, Ling and Teo [4] also recognise the lack of standards as one of the main inadequacies in OO data models. Finally, even within the relatively familiar area of OO analysis and design, a well defined, complete model of OO is lacking [5].

An empirical analysis of recent directions in OO systems development research, from 1992-1994, show that theoretical foundations of OO development are still under development, and no consensus has been reached upon precise OO concepts [6]. The authors also found that most attempts to order and compare the variety of OO concepts typically have been based upon argumentation.

1.2 Potential benefits of a dominant standard

Several standard methods for OO systems development have been suggested in academia, organisations and industry consortia. Until now, none of these standards have emerged as a dominant standard. Adapting the general definition of a dominant standard given by Lee et al.[7], a set of standard methods for OO development can be called dominant if they form a:

"distinctive way of developing OO systems that has achieved and maintained the highest level of market acceptance for a significant amount of time".

The emergence of a dominant standard in a market is a key event in the evolution of an innovation. "It represents the end of the technical variation and selection cycle, and initiates an era of more incremental technological development" [7]. Lee et al. describe several type of impacts the emergence of a dominant standard can have on both the supply- and demand side. We will apply their ideas to assess, ex-ante, the possible effects of the emergence of a dominant set of methods for OO development:

First, OO product class confusion is reduced and producers will be enabled to explore greater scale economies through learning by doing effects. Once a dominant set of development methods emerges, the systems development industry will be able to reuse parts, developed in earlier projects. This reuse can be both at the design- and code level, since a dominant OO design method would exist.

Second, switching costs associated with choosing between competing sets of OO methods would be reduced for both producers and consumers. For producers, switching to new methods includes retraining personnel, redefining standards and buying new CASE-tools. Reusable parts that were developed using old methods have to be converted or will loose their value. For consumers, the largest risk is to end-up owning and using systems built with methods that are no longer supported by the industry and therefore not easy to maintain. Also, system integration problems can occur when integrating existing systems with those build according to a newer method. Consumers therefore often decide to rebuild existing systems using a new set of methods, that is believed to become dominant. This results in considerable loss of capital.

Third, a dominant set of OO methods has direct competitive effects. The emergence of a dominant set of methods to develop OO systems will influence the nature of competition in the industry. For example, price competition will increase in the consultancy- and CASE-tool market. As an example, in the current market a competitive factor of a CASE-tool can be the variety of design methods it supports. Once a dominant design method emerges, many different CASE-tool vendors will support this method, which will lead to high competition and lower prices.

Fourth, research and development in OO will focus more on process quality improvement than on product innovation. Currently, much research effort is dedicated to developing new OO methods. As a dominant set of methods emerges, research and development can focus more on measuring and improving OO systems development. OO quality metrics can be developed for the dominant methods. Empirical studies can be conducted to test and improve the dominant methods. The emergence of dominant methods for OO is also likely to increase the adoption of OO as a whole. As reported by Yourdon [8], several producers as well as consumers now stick to conventional development methods because of the confusing state of the OO market.

1.3 OO systems development

In order to investigate possible causes and solutions for the lack of an OO standard, a definition of OO systems development is needed. We have adopted the general definitions given by Welke [9] and Olle et al. [10] to specifically describe OO systems development (see Figure 1).

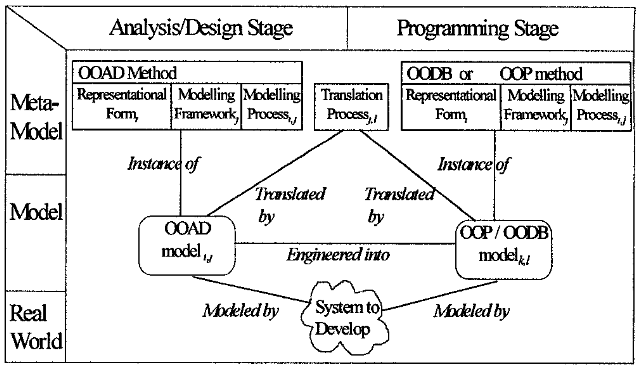

Figure 1: OO Systems Development

A rich variety of analysis and design- (OOAD), programming- (OOP) and database- (OODB) methods[1] currently exists. Although mostly discussed separately, these methods share the property of containing a number of representational forms, modelling frameworks and modelling processes. *Representational forms* define notations to create models of the system under development. These can be graphical (such as boxes, lines and arrows used in object interaction diagrams), non-graphical (e.g. matrices, tables or text) or symbolic (e.g. an OO computer program). Although usually implicitly presented, an OO method also offers a *modelling framework*, which provides constructs to help the developer to perceive some aspect of the system under development. For example, a modelling framework of an OOAD-method can define the semantics of the object and class constructs. Using a modelling framework and a representational form the developer creates a *model*. This model can be used to communicate views of the system or can be executed by a computer. To aid in this modelling activity, a method describes some *modelling processes*. These processes describe activities the developers have to perform to create the model. A separate set of *translation processes* is needed to aid developers in engineering OOAD models into executable models or the reverse. These are commonly not included in existing methods, and therefore separately defined in Figure 1. The solid lines in Figure 1 indicate relationships among the different elements of the definition (the names of the relationships should be read in upward direction) .

As an example, using the Booch [11] OOAD method, a developer can create a class model of a system. The representational form used defines the representation to be used (cloud-shaped classes, arrows between these clouds indicate inheritance etc.). The modelling framework is informally defined in the method and contains descriptions

[1] Often, a number of complementary methods is integrated into a "methodology".

of the semantics of classes, objects, inheritance etc. The Booch method also includes informal modelling processes which guide the recognition of classes and class structures. Although not explicitly present in the method, a translation process could be described to guide the translation of a class model into a C++ (executable) model.

1.4 Approaches towards standardisation

As outlined in section 1.2, achieving a dominant standard for OO systems development can have several benefits. However, the modelling frameworks of current methods are usually implicitly defined and not based on any common standard. This lack of a common framework hinders the rise of a dominant standard.

In this research recent standardisation efforts are investigated based on the approach followed to create the standard. Three main approaches can be distinguished: argumentation, ontology and metamodelling. The foundations of these three approaches are described and examples are given of their use (section 2-4). Next, advantages and drawbacks of the different approaches are evaluated (section 5). The discussion addresses a possible strategy towards a dominant standard (section 6), using the integrative definition of OO given in this introduction. This section also presents future research.

2. The argumentation approach

2.1 Argumentation and OO standardisation

In the argumentation approach to OO standardisation the opinions and experiences of authors, organisations or consortiums play a decisive role in selection and description of a set of standard OO concepts. In some cases, the authors may have first organised existing surveys of OO concepts and terminology. Usually, these surveys take one of an OOAD, an OOP, or OODB point of view. Using some classification scheme, the variety of concepts encountered are ordered and described. Sometimes, based upon this set of OO concepts, a standard is explicitly proposed. In other cases, concepts are surveyed and explained, without presenting an explicit standard, thus contributing somewhat indirectly towards a standard.

2.2 Examples of the argumentation approach

Several authors have compiled extensive surveys of existing OO terminology. Well known examples of such surveys are the Henderson-Sellers book of OO knowledge [12], Firesmith's dictionary of object terminology [13], Nierstrasz's survey of OO concepts [14] and Wegner's overview of OOP concepts [15]. In these surveys, while the semantics of OO concepts are addressed and can serve as a basis for standardisation, no explicit attempt is made towards standardisation.

An argumentation approach towards standardisation of the concept of inheritance can be found in Winkler [3]. In this study, the author recognises a dichotomy between the concept oriented view and the programming oriented view of inheritance. Often, during implementation, dramatic changes are necessary to inheritance hierarchies developed in OOAD. Winkler advocates to adapt the program oriented view of inheritance and thus standardise the associated terminology.

Van de Weg and Engmann [16] construct a "unifying" modelling framework for OOAD that is "theoretically sound and complete". The framework orders OO concepts based on their role in modelling object statics, object dynamics and object structure. The framework is populated by selecting different concepts from the areas of databases and programming (see Table 1). The authors illustrate how the elements of the framework can be formally described using a grammar and include a graphical representational form and informal descriptions of modelling processes.

Table 1: Van de Weg et al. 's framework [16]

	Statics	*Dynamics*
Inter-object structure	**Relationships** classification: - association - generalisation - roles - aggregation - grouping	**Common Actions** classification: - synchronous - asynchronous - consecutive
Intra-object structure	**Attributes** abstraction: - states classification: - identifiers - properties - variables - states	**Actions** abstraction: - state transitions ordering: - life history

The Object Management Group (OMG), an industry consortium of over 500 companies, has put forward the Object Management Architecture (OMA) [17]. Part of OMA is a definition of core OO concepts. The definitions are given in structured English and have been established through voting by the members of the consortium.

3. The ontological approach

3.1 Ontology and OO standardisation

The proponents of the ontological approach claim that primitives used in representational forms can be studied, understood and integrated by providing an ontological level characterisation. It is claimed that using ontology, the semantics of OO can be precisely defined on a conceptual level [18].

Within philosophy, ontology is the branch of metaphysics that deals with the nature of being [19]. The ontological approach makes use of theories developed by linguists and philosophers to describe the real world. An ontology offers an explicit way to conceptualise the world. It provides a set of primitives to organise three different types of abstract knowledge; ontological primitives, structural knowledge and dynamic knowledge [20]. Within ontology, several theories exist differing in the aspects of the real world they aim to describe and in level of formality and detail. For example, to describe dynamic knowledge, an ontology based on events an actions can be used.

Since different types of ontologies exist, the application of ontology to standardise OO requires a choice of ontological primitives [20]. This choice is determined by several considerations, such as the required focus, level of formality and detail.

3.2 Examples of the ontological Approach

Bonfatti and Pazzi [20] apply ontology to create a foundation for the concepts of object state and identity. The authors use ontological principles to justify the necessity of unique object identity. Leibniz' ontology says that in nature no ambiguity exist among individuals, but only the full knowledge of all properties assures that they can be distinguished. In order to uniquely recognise objects in a system representation, where objects only have a limited set of properties, unique identity is necessary. This object identity is constant, e.g. independent of dynamic changes in the object state. Using the ontological definitions of object state and identity, Bonfatti and Pazzi also attempt to settle the debates on the assumption that; "everything can be viewed as an object" , including values (this is the case in Smalltalk based systems, see [21]). They propose to view values as fixed state objects, which have their state bound to their identity.

Taivalsaari [22] addresses similar conceptual issues using ontology. He applies the classical Aristotelian view of the world, where the real world is viewed fundamentally composed of objects and their properties. He argues that ontology recognises two different kind of abstractions; Eternal abstractions are typically known as values. They cannot be copied, instantiated or changed. Tangible abstractions have a natural beginning and ending. The tangible abstractions are objects. Objects can be created, destroyed and can be changed during their lifetime. The author list numerous other differences between values and objects, and concludes that based on ontology, it is not desirable to consider values and objects identical.

Artale et al. [23] use ontology to more precisely define the structural abstractions in the object model, focusing on part-of relationships among objects. The authors state that "the part-whole relation cannot simply be considered as an ordinary attribute: its specific ontological nature requires to be understood and integrated within data modelling formalisms and methodologies". It is concluded that OO representations use a simple view of the theory parts and wholes, failing to use the extensive theory developed in ontological studies.

Takagaki and Wand [24] use ontology to develop an formal object model based on Bunge's ontology. The authors emphasise the distinction between real world objects and models of these objects. Consequently, the real world objects have properties but the model objects have attributes, which are selected by the designer. The model recognises different types of object compositions. Dynamics are captured by describing events and change functions that cause these events to happen. Laws are introduced to limit the possible values of state variables. If an event causes a state variable to break a law, actions will be taken to correct this.

4. The metamodelling approach

4.1 Metamodelling and OO standardisation

Metamodelling has been applied for a variety of purposes. For example, metamodels have been used to develop flexible CASE-tools [9,25-27], to build automated support for translation between different system models [28-31] and to analyse-, and compare- [10,32] and more recently, build systems development methodologies [33,34].

A metamodel can be defined as "a conceptual model of a modelling technique" [35]. Two types of metamodels can be distinguished. Meta-data models which describe models of systems development representations, and meta-process models which describe systems development processes [27,35]. Since standardisation efforts have mainly focused on standardising static aspects of methods, we will focus on meta-data models.

Meta-data models capture the more static aspect of a method and are constructed using techniques similar to those used in constructing regular system models. Depending on the required level of detail the choice of the most appropriate modelling method can vary [27]. Often, variants of Entity Relationship (ER) modelling are used. Other applications might need a more elaborate metamodelling method such as OPRR [27], or a formal modelling method such as Z [36].

Several ways in which metamodels can be applied to aid in the standardisation of OO have been proposed. Existing OOAD and OOP modelling frameworks and processes can be described by creating their metamodels. Based on these resulting metamodels, *core-* and *extended* OO concepts can be distinguished. Core concepts are widely used and accepted, and result from an intersection of the individual metamodels. Extended concepts are recently developed, usually method specific and become visible by creating joins of the individual metamodels. Such metamodels, describing core- and extended OO concepts, can serve as a basis for a standardisation discussion. As is observed in [37]: "...many methodologists have agreed that metamodelling may provide a way forward to both underpinning the methodologies themselves by something a little more rigorous than the current informality and also to creating a potentially agreeable core meta-object model for all methodologies to adhere to".

4.2 Examples of the metamodelling approach

Hong et al. [32] use metamodelling to compare six OOAD methods. In this research, metamodels are created of the OOAD methods and merged into a "super-method" metamodel. The individual metamodels are now compared against the super-method. Although the goal of this research is not directly standardisation, the results objectively identify the main differences that have to be resolved to reach a standard method.

A recent attempt to organise concepts from a OOAD point of view and create a standard OOAD method is undertaken by Booch and Rumbaugh jointly with Jacobson [38]. In their "Unified Method" an attempt is made to "represent the unification of the Booch and OMT methods as well as the best ideas from a number of other methodologists. By unifying these two leading OOAD methods, the unified method provides the basis for a *de-facto* standard in the domain of OOAD founded on a wide base of user experience". Booch and Rumbaugh define the Unified Method by constructing a meta-data model, to provide a "reasonably formal" description of the modelling concepts, semantics and the corresponding notation. The Unified Method metamodel has facilitated a rich discussion among OO users on the Internet concerning standard OOAD concepts and especially, the method's notation.

Snyder [1] describes the standardisation efforts of a Hewlett-Packard task force that developed a set of core concepts in object technology. The approach followed can be described as a metamodelling approach using informal modelling techniques. The concepts were elicited from a group of OOPL's, OODB's and OO operating systems. Snyder proposes a set of terms as representative for the core OO concepts, which are further refined into an abstract object model (see Table 2).

Table 2: Examples of Snyder's standard naming [1].

Proposed Term	*Related Terms*
object	instance, class instance, surrogate, entity
object reference	handle, object identifier, object name
request	message, method invocation, function invocation
interface hierarchy	inheritance, specification hierarchy, type hierarchy, class hierarchy, subtyping, conformance
...	...

The American National Standards Institute (ANSI) X3H7 committee concentrates on standardising OO models [39]. In order to identify essential differences and commonalties, the committee selected to include OOPL's, OODB and OOAD methods as well as specific standards such as OMG Core Object Model. This information was then structured by building an Object Model Features Matrix. In this approach the metamodel *is* the object features matrix, which can be viewed as a model of the specific methods. Rows in the matrix denote specific OO methods. Columns denote specified features of the OO method's modelling framework. Some examples of columns are; state, lifetime, identity, operations, types and classes, dynamic extensibility etc. The cells in the matrix contain descriptions of the presence of the feature. These descriptions have not yet been validated by the original developers of the methods.

To achieve a standard CASE data interchange format, the CDIF/OOAD working group also has followed a metamodelling approach [40]. First, metamodels were created of a number of leading OOAD methods, including Martin-Odell, OMT, Shlaer-Mellor etc. Only the OOAD concepts were included and programming language specific constructs were left out. Second, the common concepts of the individual models were integrated into a common metamodel. To create the metamodel, extended ER-modelling has been used. The metamodel attempts to capture the semantics of the different OOAD modelling constructs. Since the main goal of CDIF's metamodel is to exchange data, the metamodel does not attempt to enforce standard naming conventions.

Recently, the Centre for Object Technology Applications and Research (COTAR) of the University of Sydney has initiated the COMMA (Common Object-Oriented Methodology Metamodel Architecture) project [41]. The goal of this project is to build metamodels of a large number of popular OOAD methods, which will be validated by the authors of the original methods. Next, the metamodels will be integrated into a core metamodel, and form the basis of a de-facto industry standard. The COMMA initial focus will be on standardisation of syntax and semantics, but will possibly also address notation, modelling techniques and high level processes. The ACOMP (Agreement on Core Object Methodology Principles) initiative aims to use the COMMA-metamodels to invite several leading OO methodologists to agree upon basic concepts which will be grouped into "coherent and flexible object development frameworks" [37].

5. Assessment of the standardisation approaches

5.1 Properties of standardisation approaches

None of the attempts towards a standard described have, as yet, produced a dominant standard in the OO field.

Table 3: Properties of reviewed approaches

Approach	*Formality*	*Scope*	*View:A*[2]	*P*	*D*
Argumentation					
OMA [17]	inf.	OOAD		+	+
Van deWeg et al. [16]	form.	OOAD	+	+	
Winkler [3]	inf.	inheritance	+	+	
Ontology					
Taivalsaari [22]	inf.	object		+	
Artale et al. [23]	form.	part-whole	+	+	+
Bonfatti et al. [20]	form.	object state	+	+	
Takagaki et al. [24]	form.	OOAD	+		
Metamodelling					
Snyder [1]	inf.	OO		+	+
ANSI-X3H7 [39]	semi-	OO	+	+	+
Hong et al. [32]	semi-	OOAD	+		
Booch et al. [38]	semi-	OOAD	+		
COMMA [41]	semi-	OOAD	+		
CDIF [40]	semi-	OOAD	+		

Apart from the approach followed, a standardisation effort can also vary in the viewpoint chosen (OOAD, OOP or OODB) and in the level of formality and scope of the resulting standard (see Table 3). In this section we survey strength and weaknesses of the different approaches and assess these differences.

5.2 Assessment of the argumentation approach

The key criticism of the argumentation approach is the ad-hoc process used to develop the standard. Choices made by the designers are often not documented. In the field of OOAD, the creation of several methods appears to based upon an arbitrary selection of OO primitives and notations. Several methodologies do not emphasise on completeness or semantics of the introduced representational forms and modelling processes. The underlying modelling framework usually receives little attention and is only informally described.

[2] These abbrevations indicate an Analysis and design, Programming and Database viewpoint respectively.

5.3 Assessment of the ontological approach

The most obvious remark concerning the ontological approach is that, like in OO modelling, several ontological frameworks exist within ontology. The choice of constructs to use from an ontology is arbitrary [18]. For example, Takagaki and Wand [24] choose to apply Bunge's ontology, but acknowledge that "another set of ontological assumptions could have led to a different information system model". This could lead to the use of ontology to prove conflicting statements. As an example, while Taivalsaari [22] uses ontology to argue that things have a natural 'beginning and 'ending', Takagaki and Wand [24], note that "ontologically things do *not* disappear. Rather they persist until they change into other things". It seems that first an agreement is needed as to which ontology to use to standardise OO concepts.

Next, a complete ontology does not necessary lead to a complete OO method. Ontology by itself does not cover all aspects of conceptual modelling [18]. For example, ontology is concerned with modelling frameworks, not so much with modelling processes. The process aspect of a method therefore does not benefit directly from ontology.

Finally, there is the issue of practical application of ontology to systems development. Ontological constructs may be very well applicable in OOAD, but not so much in OOP. In some cases, ontological constructs, if implemented in their pure form in an object model, lead to inefficient implementation [22]. As an example, Bonfatti and Pazzi [20] choose not to follow Leibniz' ontology by adding *constant object identity* to their model for "economic and natural" reasons.

5.4 Assessment of the metamodelling approach

Perhaps the most interesting criticism on the metamodelling approach is that metamodelling is itself a modelling activity which can only be successful if an unambiguous and well defined metamodelling method exists. Even more than traditional modelling, metamodelling is a complicated and creative activity. If the used metamodelling method is incomplete, the constructed metamodel might miss important aspects of the modelled domain, that is, the OO method. Currently, extended ER-modelling seems to be a popular technique for creating metamodels. However, even in extended ER modelling, no clear defined modelling processes are known on eliciting entities from methods. Moreover, extended ER-modelling does not include any process modelling, and thus does not capture the process aspect of methods. Booch and Rumbaugh [38] find this process aspect less important: "At some future time, we will describe a unified process, although this is far less urgent, since a common notation is amenable to many different development processes, depending upon the nature of the system being developed and the business factors and culture that shape the development organisation itself".

The metamodelling approach followed is often not traceable and usually semi-formal. Since no clear modelling process exist, the metamodels are created iteratively and should be subsequently validated. Several standardisation efforts are indeed created by subsequent revisions and sometimes by public review. Both the ANSI and the Unified Method metamodel were created in this way. A different process to ensure the quality of the metamodel was adopted in the COMMA project, where metamodels are circulated to the original method developers for their ratification or correction.

Another critical step in the metamodelling approach is to integrate the different metamodels. The same metamodel notation and a comparable level of detail should be used to allow integration. Since several similar methods modelled, the metamodels very likely have many common components. Problems that occur when integrating such different metamodels are similar to those found in database view integration [42]. Problems of synonymy and homonymy have to be resolved to find true semantic equivalencies [43]. In current metamodel approaches, this research step is often not documented. For example, Booch and Rumbaugh [38] present the Unified Method metamodel in much detail, but the process of constructing this model from the OMT and Booch method is not documented and probably informal. The intraceability of this "unification process" makes it unrepeatable and difficult to criticise.

Finally, selecting which individual methods to metamodel to create a good 'core' metamodel is also non-trivial. For this reason, the OMG chooses not to follow the metamodelling approach. OMG feels an intersection of metamodels might be too minimal to be of use, while the unification of metamodels might have too many concepts to be comprehensible. Different strategies are chosen to attack this issue. In the COMMA project a very large number of methods are metamodelled. The ANSI X3H7 committee chooses to model a wide variety of methods and OOPL's, not necessary the most popular, to be sure to capture all interesting OO concepts.

5.5 Assessment of the level of formality

Formally defining the primitives of representational forms and modelling processes has several advantages. It facilitates unambiguous communication among developers and communications with the customer. Using first order logic customer requirements can be stated in precise terms and can be checked on validity and measured on quality. Also, formally defined representations are a sound basis for developing formal processes that perform translations between analysis, design and implementation models. For these reasons, ANSI's X3H7 committee also sees formal specification of object concepts and behaviour as its ultimate goal [39].

Argumentation and ontological approaches frequently use formal notations. The level of formality used in metamodelling approaches usually is semi-formal, which makes the resulting "core" metamodels not directly suitable for definition of standards. It may be needed to add a more formal definition of the semantics of the 'core' metamodel later.

Taivalsaari [22], however questions the suitability of mathematical formalisms to standardise OO systems development. He argues that: "in fact, there is a logical reason why no commonly accepted theory of OOP exists. Theories and formalisms should be based on a sound mathematical basis. Mathematical formalisms are essentially value based. Concepts that are central to OOP such as state, updating and sharing do not naturally fit very well to such formalisms".

Furthermore, a complete formal definition of the variety of OO concepts will be very complex, and difficult to build. Also, since empirical studies have shown that developers often have difficulties understanding mathematical formal notations [44], such a standard is likely to cause interpretation problems for most developers.

5.6 The chosen viewpoint and scope

Eventually, only an OO standard that takes OOAD, OOP as well as OODB viewpoints into account is capable to integrate the entire OO field. However, defining standards having a wide scope consequently leads to a low level of detail. Such standards can impossibly address complex object structuring mechanisms, such as composition or inheritance, in much detail.

For example, Takagaki and Wand [24] emphasise on object compositions and dynamics, but do not present other mechanisms such as inheritance or object identity.

6. Discussion

6.1 A standard modelling framework

In the last decade, OO research in the methodology-, programming- and database research communities has focused on developing various representational forms. Although representation differences can be eliminated, differences in the associated modelling framework create more severe problems. Usually, constructs of modelling frameworks such as class, object or inheritance are informally described. Furthermore, during OO systems development, methods are used that use different modelling frameworks that are inconsistent in their use and interpretation of OO concepts. Clearly, a standard modelling framework will have several important benefits. Unfortunately, the standardisation approaches reviewed in this paper have not yet resulted in a dominant standard. In this discussion, we argue a standardisation approach must possess certain properties in order to be successful.

A standardisation approach should be aimed at the creation of a standard OO modelling framework, which can be used in all development stages. The role of this modelling framework is illustrated by Figure 2. The view of OO system development presented in this figure is similar to Figure 1, but an extra layer is added containing the standard modelling framework. The modelling frameworks of individual methods are a specific view of this standard modelling framework. Which view is chosen, depends on the specific goal of the method. For example, a design method focusing on system dynamics during design, would use a view on the standard framework that includes a description of the object- and message constructs.

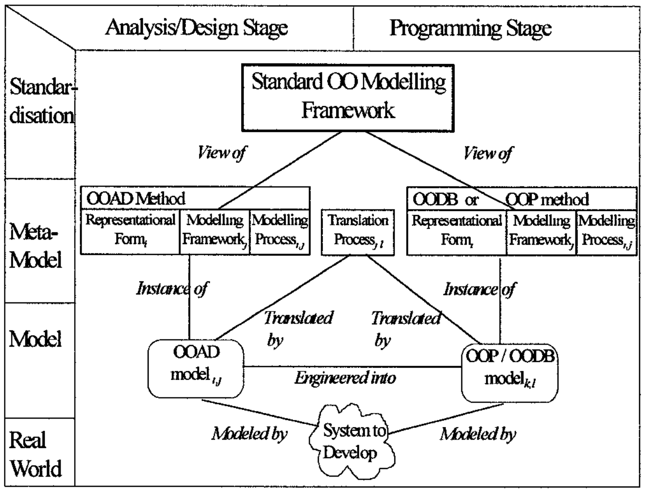

Figure 2: OO Systems Development Standardisation

Ideally, representational forms available are also standardised, but this is not necessary since the underlying standard modelling framework ensures easy translation among different notations. Newly introduced "extended OO constructs" should, if proven beneficial to OO systems development, be added to the standard modelling framework, enabling future methods to make use of these concepts.

6.2 Properties of a successful approach

From the overview of recent standardisation approaches given earlier, typical differences among the approaches are summarised in Table 4. Standards resulting from argumentation approaches are usually informal, may contain an arbitrary selection of concepts and often concentrate on either OOAD or OOP. The ontological approach draws from a solid base of well defined concepts, usually to concentrate on single OOAD concept in much detail. Metamodelling usually results in semi-formal standards with a wide scope. Currently, the use of metamodelling is limited to OOAD.

Table 4: Typical Properties of different approaches.

Approach	*Formality*	*Scope*	*View:*A^2	*P*	*D*
Argumentation	inf.	wide	+	+	
Ontology	inf. or form.	narrow	+		
Metamodelling	semi-	wide	+		
Complete	at least semi-	wide	+	+	+

Table 4 also lists the desired properties of a "complete" standardisation approach. The "complete" approach has to incorporate concepts of OOAD-, OODB- and OOP methods. The differences in interpretation of concepts within analysis, design- and programming are one of the important problems a standard modelling framework has to address. Furthermore, the standardisation process should make use of the huge body of available knowledge on OO development and must not attempt to introduce a new standard from scratch. The appropriate level of formality to be used is not easily decided. However, a complete formal definition of the variety of OO concepts will be very complex, difficult to understand and build. Therefore, the "complete" standardisation approach will produce a "semi-formal" definition of the OO concepts. The "complete" approach should be based upon a documented and repeatable standardisation process. Preferably, the products of the process must be freely accessible and be subject to validation by others.

6.3 Future research

Although argumentation and ontological approaches can be useful to introduce and define OO concepts, we argue a metamodelling approach towards a dominant standard for

OO systems development holds the most promise. Several strengths of the metamodelling approach contribute towards this belief:

First, concepts invented in countless methodologies and languages are captured and integrated rather than reinvented. Second, metamodelling seems to be able to capture and combine concepts from OOAD, OOP and OODB. Although applications of metamodelling have been mainly limited to OOAD, some attempts to apply metamodelling to describe OOP concepts have been undertaken [45]. Third, the semi-formal notations used in metamodelling have proven to facilitate discussion among OO system developers. These notations are understandable and more rigorous than "natural language" but yet less complex than mathematical formalisms.

Although metamodelling is chosen as the "closest" to the "complete" approach, ontology and argumentation can be used in addition to metamodelling: Using theory from ontology, particular constructs, such as composition and identity can be investigated further and expanded. Since none of these approaches are deterministic, argumentation will decide the final shape of the framework.

The authors are currently involved in a research project which follows this approach and is aimed at developing a modelling framework based on a number of leading OOP and OOAD methods.

7. References

[1] A. Snyder, The essence of objects: Concepts and Terms, *IEEE Software*, vol. no. January, 1993, pp. 31-42.

[2] L. L. Constantine, Object-oriented and function oriented software structure. A revised form of 'objects, functions and extensibility', *Computer Language*, vol. 7 (Jan), 1990, pp. 34-56.

[3] J. F. H. Winkler, Objectivism: "Class" considered harmful, *Comm. ACM*, vol. 35, no. 8, 1992, pp. 128-130.

[4] T. W. Ling and P. K. Teo, Toward resolving inadequacies in object-oriented data models, *Information & Software Technology*, vol. 35, no. 5, 1993, pp. 267-276.

[5] D. E. Monarchi and G. I. Puhr, A research typology for Object-Oriented Analysis and Design, *Comm. ACM*, vol. 35, no. 9, 1992, pp. 35-47.

[6] J. van Hillegersberg, K. Kumar, and R. J. Welke. Directions in Object-Oriented Systems Development Research. In: *Proc. of the 4th ECIS: European Conference on Information Systems*, eds. J. Dias Coelho, T. Jelassi, W. König, H. Krcmar, R. O'Callagham, and M. Sääksjärvi. Lisbon: 1996, pp. 633-646.

[7] J. R. Lee, D. E. O'Neal, M. W. Pruett, and H. Thomas, Planning for dominance: a strategic perspective on the emergence of a dominant design, *R&D Management*, vol. 25, no. 1, 1995, pp. 3-15.

[8] E. Yourdon. *Object-Oriented Systems Design. An Integrated Approach*, Englewood Cliffs, NJ: Prentice Hall, 1994.

[9] R. J. Welke. IS/DSS: A DBMS support system for information systems development. In: *Database Management: Theory and Applications*, eds. C. W. Holsapple and A. B. Whinston. Dordrecht: D. Reidel Pub. Company, 1983, pp. 195-250.

[10] T. W. Olle, J. Hagelstein, I. G. Macdonald, C. Rolland, H. G. Sol, F. J. M. v. Assche, and A. A. Verrijn-Stuart. *Information Systems Methodologies: A framework for understanding*, Workingham,England: Addison Wesley, 1991.

[11] G. Booch. *Object-oriented analysis and design with applications*, Redwood city, California: Benjamin / Cummings, 1993.

[12] B. Henderson-Sellers. The Object-Oriented Paradigm. In: *Book two of object-oriented knowledge*, Englewoods Cliffs, NJ: Prentice Hall, 1994, pp. 41-101.

[13] D. G. Firesmith and E. M. Eykholt. *The Dictionary of Object Technology*, New York: SIGS Books, 1995.

[14] O. Nierstrasz. A survey of Object-Oriented Concepts. In: *Object-Oriented concepts, Databases and Applications*, eds. W. Kim and F. Lochovsky. Addison-Wesley, 1989, pp. 3-21.

[15] P. Wegner, Concepts and paradigms of object-oriented programming, *OOPS Messenger*, vol. 1, no. 1, 1990, pp. 7-87.

[16] R. L. W. van de Weg and L. Engmann. A framework and method for object-oriented information systems analysis and design. In: *Proceedings of the IFIP TC8/WG 8.1 Working conference on Information system concepts: Improving the understanding*, eds. E. D. Falkenberg, C. Rolland, and E. N. El-Sayed. North-Holland: Elsevier Science Pub. 1992, pp. 123-146.

[17] R. M. Soley and C. M. Stone. *Object Management Architecture Guide*, New York: Wiley, 1995.

[18] Y. Wand, D. E. Monarchi, J. Parsons, and C. C. Woo, Theoretical foundations for conceptual modelling in information systems development, *Decision Support Systems*, vol. 15, 1995, pp. 285-304.

[19] J. e. Sinclair and P. e. Hanks. *Collins Cobuild -English Language Dictionary*, London: HarperCollins Pub. 1991.

[20] F. Bonfatti and L. Pazzi, Ontological foundations for state and identity within the object-oriented paradigm, *International Journal of Human-Computer Studies*, vol. 43, no. 5/6, 1995, pp. 891-906.

[21] A. Goldberg and D. Robson. *Smalltalk-80 - The language and its implementation*, Reading,MA: Addison-Wesley, 1983.

[22] A. Taivalsaari, On the notion of Object, *Journal of Systems Software*, vol. 21, 1993, pp. 3-16.

[23] A. Artale, E. Franconi, N. Guarino, and L. Pazzi, Part-Whole relations in object-centered systems: an overview, *Data & Knowledge Engineering*, vol. 16, no. 12, 1995.

[24] K. Takagaki and Y. Wand. An object-oriented information systems model based on ontology. In: *Proceedings of the IFIP TC8/WG 8.1 Working conference on the object-oriented approach in Information Systems*, eds. F. Van Assche, B. Moulin, and C. Rolland. North-Holland: Elsevier Science Pub. 1991, pp. 275-296.

[25] D. Teichroew, P. Macasovic, E. A. Hershey, and Y. Yamamototo. Application of the Entity-Relationship approach to information systems modelling. In: *The entity-relationship approach to systems analysis and design*, ed. P. P. Chen. North-Holland: 1980.

[26] K. Smolander, K. Lyytinen, T. Veli-Pekka, and P. Marttiin. Meta-Edit --- A flexible graphical environment for Methodology Modelling. In: *Advanced Information Systems Engineering, Third International Conference CAiSE'91: Proceedings*, eds. R. Andersen, J. A. Bubenko jr., and A. Solvberg. Berlin: Springer-Verlag, 1991, pp. 168-193.

[27] R. J. Welke. The CASE repository: More than another Database Application. In: *Challenges and strategies for research in systems development*, eds. W. W. Cotterman and J. A. Senn. Chichester: John Wiley & Sons, 1992, pp. 181-214.

[28] M. D. Fraser, K. Kumar, and V. K. Vaishnavi, Informal and Formal requirements specification languages; Bridging the Gap, *IEEE transactions on SE*, vol. 17, no. 5, 1991, pp. 454-466.

[29] P. Atzeni and R. Torlone, A metamodel approach for the management of multiple models and translation of schemes, *Information Systems*, vol. 18, no. 6, 1993, pp. 349-362.

[30] S. A. Demurjian and D. D. Hsiao, Towards a better understanding of data models through multilingual data systems, *IEEE transactions on SE*, vol. 14, no. 7, 1988, pp. 946-958.

[31] M. Morgestern. A Unifying approach for conceptual schema to support multiple data models. In: *Entity-Relationship approach to information modelling and analysis.* ed. P. P. Chen. North-Holland corp. 1983, pp. 279-297.

[32] S. Hong, G. v. Goor, and S. Brinkkemper. A formal approach to the comparison of Object-Oriented analysis and design methodologies. In: *Proc. of HICSS-26*, 1993, pp. 689-698.

[33] K. Kumar and R. J. Welke. Methodology Engineering: A proposal for situation specific methodology construction. In: *Challenges and strategies for reserach in systems development*, eds. W. W. Cotterman and J. A. Senn. Chichester: John Wiley & Sons, 1992, pp. 257-266.

[34] F. Harmsen, S. Brinkkemper, and H. Oei. A language and tool for the engineering of situational methods for information systems development. In: *Porc. of the ISD'94 conference*, eds. J. Zupanic and S. Wrycza. Bled: 1994.

[35] S. Brinkkemper, Formalisation of Information systems modelling pp. 1-209, 1990. Katholieke Universiteit Nijmegen, The Netherlands. PhD-Thesis. Thesis Publishers.

[36] M. Saeki and K. Wenyin, Specifying Software Specifications & Design Methods eds. G. Wijers, S. Brinkkemper, and T. Wasserman. eds. G. Goos and J. Hartmanis, pp. 353-366, 1994. Lecture Notes in Computer Science 811. Springer-Verlag. Berlin.

[37] ACOMP, Agreement on Core Object Methodology Principles (ACOMP), *L'OBJET*, vol. 1, no. 3, 1996, pp. 1-4.

[38] G. Booch and J. Rumbaugh, Unified Method : Metamodel Description Version 0.8, pp. 1-45, 1995. Rational Software Corporation. Available by Rational.

[39] X3H7 Technical Committee, ANSI X3H7 (Object Information Management) Technical Report ed. F. Manola. Interim Report, pp. 1-25, 1995.

[40] CDIF, Draft for the CDIF/OOAD Meta-model ed. J. Ernst. CDIF-JE-N24-V1, pp. 1-24, 1995.

[41] B. Henderson-Sellers, COMMA: An architecture for method interoperability, *Report on Object Analysis and Design*, vol. 1, no. 3, 1994, pp. 25-28.

[42] C. Batini, S. Ceri, and B. Navathe. *Conceptual Database design. An Entity-Relationship Approach*, Redwood City, CA: Benjamin/Cummings Pub. Comp. 1992.

[43] P. M. Steele and A. B. Zaslavsky. The role of meta models in federating system modelling techniques. In: *Lecture Notes in Computer Science 823*, Berlin: Springer Verlag, 1995, pp. 315-326.

[44] K. Finney, Correspondence: Mathematical Notation in Formal Specification: Too difficult for the masses ? *IEEE transactions on SE*, vol. 22, no. 2, 1996, pp. 158-159.

[45] J. van Hillegersberg and K. Kumar. Using Meta-modelling in understanding object-oriented programming. In: *ECOOP 1995 workshop on Education in OO*, eds. S. Holland and H. Sharp. Arhus, Denmark: 1995.