Human work as context for development of object oriented modelling techniques

J. J. Kaasbøll and O. Smørdal
Department of informatics, University of Oslo

Published in Brinkkemper S (ed.) IFIP WG 8.1/8.2 working conference on principles of method construction and tool support (Method Engineering' 96) (Atlanta, USA), Chapman & Hall, pp. 111-25.

Abstract: Computer systems are increasingly being used for communication and coordination of work, while object-oriented modelling techniques aim at modelling the problem domain of the computer system. Current techniques have been developed with respect to easy implementation, while we argue that further development of the modelling techniques should also be based on knowledge about human work in organisations.

We outline a learning cycle of modelling technique and point to where such knowledge should be included.

We have carried out two alternative approaches to development of object oriented techniques based on these ideas, and we outline these development processes. One approach is based on semiotic concepts, the other is based on activity theory.

Keywords: Research Method, Method Engineering, Learning Cycle, Activity Theory, Semiotics, Evaluation

1 Introduction

Object oriented modelling techniques should be developed according to knowledge about human work within organisations. This paper argues why and points to ways to change current development practice.

The basic ingredients of object-oriented techniques for modelling are the mechanisms provided by object-oriented programming languages. In short, these mechanisms consist of encapsulated objects with properties and behaviour, and specialisation of classes by means of inheritance. It is often claimed that object-oriented modelling of the domain of an information system is easy, because object-orientation corresponds to our natural conception of the world. Considering that the core concepts of object-oriented techniques consist of implementation restrictions, we doubt the correctness of this claim.

Object oriented techniques are used within application areas that include human work within some organisation. Lately, the techniques have also been used to capture aspects beyond the domain of work, e.g., aspects relating to actors, communication, coordination of work, task flow, and work procedures. This is due to a shift of perspectives regarding the role of the computer in work settings; from a focus on the computer as means of control and administration of a problem domain, to a focus that also include the computer as a mediator in the work setting, e.g., as in CSCW applications. Carstensen et al (1995) point to inadequacies of object-oriented modelling in these respects. Others have reported problems related to modelling of different roles of actors (Richardson and Schwarz 1991; Coad 1992). These findings underpin our disbelief in the claim of the easiness of modelling. Research on the difficulties of learning object-oriented modelling (Vessey and Conger 1994) also indicate that the claim is incorrect.

We interpret these observations as symptoms of an underlying problem: that the development of object-oriented techniques for modelling has been too restrained by implementation considerations. The inadequacies that have been detected have been explained within the frame of the mechanisms of object-oriented programming languages, the theoretical contributions have been restricted to formal arguments within this frame and, consequently, the suggestions for improvements of the techniques have not extended these mechanisms. This paper aims at arguing that the way of developing techniques should open for a wider range of explanations, theories, and suggestions. In particular, we will show how we have included knowledge concerning actors in the process of developing object-oriented techniques for modelling.

1.1 Suggestions in the literature

A way to improve methods called “method engineering” has been defined as “the disciplined process of building, improving or modifying a method by means of specifying the method’s components and their relations” (Heym and Österle 1993; Rossi and Brinkkemper 1995). The concept is used to capture the development of a method and the adaptation of a method in a specific situation (Kumar and Welke 1992; Harmsen, et al. 1994), and method engineering is compared with the development and modification of an information system in an organisation.

Since the problems referred above concern object-oriented techniques for modelling in general, method engineering, which only deals with individual methods and compilation of methods from techniques, will fall short with respect to the generality of the problem. In addition, method engineering does not enrich the concepts and mechanisms for modelling, such that actors, roles, task flow, etc., are more easily modelled.

1.2 Seamlessness in modelling

An argument for object-oriented development is the seamlessness from analysis to design and implementation: the same concepts are used in all phases, such that no magic transition is needed. When arguing for richer concepts for modelling, we may put the seamlessness principle in danger.

To be precise in the further discussion, we first define areas that can be modelled during system development, based on similar concepts in Mathiassen et al (1993).

The problem domain of a computer system is what the computer system is about; the part of the world that the computer system is supposed to handle, control or monitor. Examples (with basic components): a flight booking system (flights, seats, reservations, customers), a banking system (customers, transactions, accounts, loans, interests).

The application domain of a computer system consist of the users, the organisational context, and the work in which the computer system is used, e.g., a travel agency, a bank. Elements of the application domain are employees, the coordination of work, communication, power structures, ad-hoc organised work, interruptions in work, etc.

The computer system including its application program, data/ object base, user interface module, and communication modules.

When analysing functionality requirements of a system, one could make a model of the application domain. Since it is assumed that the problem domain is more stable than the functional requirement, making an object-oriented model of the application domain is often not considered worthwhile.

Many object-oriented methods suggest that one should model the problem domain, because this is what the computer system shall represent. The model is supposed to describe how the system developers and users conceive the problem domain. For now we regard this model as based on consensus among users and developers. An advantage of a model of the problem domain is that the model is independent of the technology for implementing the system. The model can be used as a part of a specification, such that computer systems conforming to the specification can be implemented on several platforms or with different languages.

A model of the future computer system will often be an extension of a model of the problem domain in order to include software modules and objects needed for implementation. Because the same concepts are used in all models and in the implementation, the model of the future computer system can be aligned with the model of the problem domain. This is referred to as the seamlessness of object-oriented system development.

However, iterations are carried out during development, and implemented systems are changed during long periods of further development. Experience shows that changes are often carried out directly on the code, without updating the models. To keep the seamlessness, it must be possible to keep the models in alignment with the code. If other concepts are introduced in the model of the problem domain, more effort may be required to keep the models updated.

We want to include in our models issues of work organisation, and this suggests an extension of the problem domain to include aspects of the application domain. The domain definitions given above represent useful distinctions. Hence we want to introduce an another concept, the model domain, which denotes the area of concern when modelling. As we identify in the next section, the most usual model domain for object-oriented modelling techniques is the problem domain, but we also identifies some approaches that have the future computer system as the model domain.

In our work we define the modelling domain to be the problem domain plus the aspects of the application domain that is mediated by the computer system. We discuss the domain for object oriented modelling techniques in Section 2.3.

1.3 Overview of the paper

The paper is organised as follows: Section 2.2 presents a learning cycle for development of object-oriented techniques, based on the interplay between modelling in practice and theoretical contributions. We identify contributions from some current object-oriented techniques in respect to 1) their notions and concepts, 2) their embedded theory, and 3) the reported technique development.

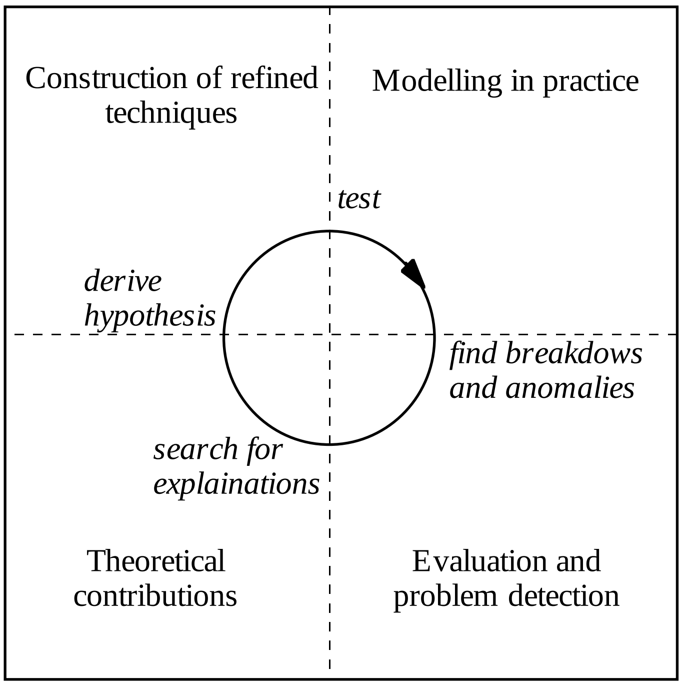
We conclude that most techniques have had a technology driven development. In Section 2.3 an extension of the domain of object-oriented techniques is suggested, also including issues of work organisation, roles, and communication between the users. Section 2.4 and 2.5 presents two development cycles that address this extended domain, one using semiotic concepts, the other using activity theory.

2 The development of techniques

In order to discuss different approaches to development of techniques for modelling, we outline a learning cycle for identifying the stages and components of the development.

Gaining new scientific knowledge can be regarded as a continuous cycle of formulation of hypotheses, evaluation in practice, explanation of results, contributions to theories, reformulation of hypotheses, etc. The learning cycle of development of techniques consists of four phases and transformations from one phase to the next, see Figure 1.

Figure 1. A model of technique development.


Construction of refined techniques
Modelling in practice
test
derive hypothesis
find breakdows and anomalies
search for explainations
Theoretical contributions
Evaluation and problem detection

Modelling in practice is the area we learn from, and also the area we want to improve.

Evaluation and problem detection is triggered by experiences when modelling is not straightforward. The main concern in this phase is to identify problems that stem from the use of this technique in a practical system development context. The problems may be identified due to 1) breakdown in the use of a technique, e.g., some property of the application domain could not be captured in the model, or the appearance of inconsistencies in the model, or 2) anomalies in the model or in the use of a technique, e.g. the resulting model seems strange compared to the application domain.

Theoretical contributions. When explaining problems in a scientific way and considering ways to avoid them, one has to consult other scientific results and theories. One may try to explain the problems within the frame of the research or search for other theories.

Construction of refined techniques. When the appropriate theoretical considerations have been made, one may have to adjust the technique and possibly include new mechanisms, metaphors and notation. Hypotheses concerning the techniques and the approach to evaluate the hypotheses are worked out.

Modelling in practice. The cycle restarts with using the technique in modelling. Any kind of practice which contributes to learning about the technique and its place in system development is feasible.

Galliers (1992) separates research goals into theory building, theory testing, and theory extension. He argues that case study, survey, forecasting, simulation, argumentation, interpretation, and action research are possible research approaches for theory building. According to his categorisation, these research methods are appropriate in the phases of evaluation and theoretical contribution. The theories are tested in the phases construction of refined techniques and modelling in practice. Theorem proof, laboratory experiment and field experiment are suited for theory testing, according to Galliers. In our learning cycle, theorem proving may take place in the theoretical contribution and during construction of techniques.

Braa and Vidgen (1995) outline three types of knowledge interests in system development research: intervention, science, and interpretation. Intervention aims at change in the organisations where computer systems are used and developed, science aims at general knowledge that is useful for prediction, and interpretation aims at explaining and understanding information systems development in organisations from different viewpoints. Inspired by these three types of knowledge interests, we construct a taxonomy of three ways of developing techniques for modelling. We will use this taxonomy to discuss how the phases of the learning cycle are covered in the way techniques are developed.

The consultant approach. A consultant is involved in development of systems, and gathers experience of her/his ways of working, and expresses this experience in general terms as techniques and methods. In this approach, the evaluation is carried out in an unscientific manner, and theoretical explanations and contributions are not included. The main goal of this approach is improvement of system development practice.

Method engineering. Scientists measure use of methods in system development. After identifying problems, they calculate improved principles, formalise vague parts of the methods, and improve tools to support implementation. In this approach, all aspects of the cycle are included, but the theoretical considerations are limited to formal theories. The main goal of methods engineering is improved predictability of system development when it is carried out according to the method.

System development research. Scientists study system development and the role of methods in practical projects. Problematic areas are identified. Relevant theories are called upon to understand and explain the problems. Improved knowledge of system development constitutes the basis for possibly suggesting improved guidelines and techniques. The main goal of the research is improved knowledge of system development from different viewpoints, and the role of techniques therein.

Method engineering addresses methods and techniques in particular. System development research has a wider scope, and improvements of techniques is one of many possible outcomes. We nevertheless argue that development of methods and techniques should also be carried out in the perspective of system development research, because it opens for a richer variety of research methods and theories. When problems that lend themselves to formal methods are encountered, there is nothing that prevents an engineering approach to deal with these problems. However, if working within a method engineering perspective as outlined here, the perspective does not open for alternative interpretations or research methods.

Main differences between method engineering and system development research are found in the theoretical and the constructive phases of the learning cycle. They are summarised in Table 1.

Figure 2. Differences between Method engineering and System development research

	Method engineering	**System development research**
Viewpoint	Unified	Diverse
Explanation	Within the frame of object-orientation	Within any scientific frame
Theoretical contribution	Formal	Any kind
Suggestions for improvements of techniques	Constrained by straightforward implementation in object-oriented language	May require extensive implementation efforts or changes in the object-oriented languages

In the following, we will see that development of object-oriented techniques for modelling so far has been mainly carried out according to the methods engineering approach.

2.1 Explanations

In order to illustrate how problems in modelling often are explained, we consider modelling of actors.

When an actor can have roles that change over time, one encounters a problem in object-oriented modelling. The problem has been explained within the common concepts of object-orientation to be that the actor object has to change its class (e.g., Richardson and Schwarz 1991; Nerson 1992; Gottlob, et al. 1996). The suggestions for solutions have been minor extensions of the object concept along with guidelines for implementation.

Coad (1992) refers to another discipline when diagnosing problems in object-oriented modelling. Inspired by the concept 'pattern' in architecture, he explains that some of the problems in modelling appear because the basic object-oriented concepts are too fine-grained to capture some frequently occurring structures in domains. This explanation is grounded outside the area of object-orientation, and Coad therefore transcends the method engineering approach. The conclusion he draws is to suggest patterns of objects connected by well-known relations. This suggestion is well inside current object oriented concepts.

2.2 Theoretical contributions

Essink and Erhart (1991) have suggested a theoretical framework for conceptual modelling during analysis. Their framework departs from an ontology that is close to the core of object-orientation. The only extension is that they claim that "objects are bound by (natural) laws" (p.91), and this claim does not penetrate the formalistic assumption of object-orientation.

From their framework, they generate four kinds of abstraction relations: specialisation, containment (aggregation with parts depending on the whole), assembly (aggregation with independent parts), and grouping (set inclusion). These four relation types are specialised according to whether they are permanent or temporary, e.g., "roletype" is a temporary specialisation that may meet the need for modelling roles, which is an important aspect of actor concepts. The suggestions of Essink and Erhart have neither been used in recent methods for modelling (Henderson-Sellers and Edwards 1994; Reenskaug, et al. 1996) nor been quoted in the solution presented in (Gottlob, et al. 1996), even if their roletype relation is similar to the solution that is elaborated by Gottlob et al. One reason may be that it may be hard to decide when and how to use the different types of relations suggested, based on the brief discussion in the conference paper.

van de Weg and Engmann (1992) suggest another framework where they distinguish between interobject and intraobject structures, and static and dynamic properties. They also suggest a "role-of" relationship. Their framework does not support their suggestion of this relationship, instead they refer to an earlier suggestion by Pernici (1990), while ignoring Essink and Erhart (1991). van de Weg and Engmann's role-of relation is also ignored in recent methods (Henderson-Sellers and Edwards 1994; Reenskaug, et al. 1996), even though roles are considered in these methods and other research is cited.

The ignorance of these research suggestions shows that they have not succeeded in adding new issues to the core of object-orientation. In addition, the suggestions have been limited to formal theories.

We have not been able to collect much information about development of modelling techniques from the literature, and we have not carried out a survey on our own. Nevertheless, the available information from the method designers support the observation that the theoretical contributions have not entered the methods properly.

Rumbaugh tells how he collects knowledge for updating his Object Modelling Technique (OMT):

> Any method must grow or die, so I have used three drivers in guiding the evolutions of OMT: user experience and feedback, good ideas from other authors, and new insights of my own. (Rumbaugh 1995, p. 21)

While his reference to user experience indicates a consultant approach, he also gets good ideas from others. The material he outlines includes research discussions, e.g., concerning constraints, so he is carrying out method engineering. His considerations do not go beyond the formal and implementation issues, however. E.g., a discussion about objects that are part of several aggregates does not go beyond defining relations.

Other authors of modelling methods may draw upon a richer background of literature. However, since they to a limited extent refer to the research of formal or implementation character, it seems unlikely that they have brought wider focused theories into their considerations.

The OOram method (Rumbaugh 1995, p. 21) is one exception, in which Weber's bureaucratic theory is used as a template for how to provide structure to a system. This structure concerns design of the relations between objects and roles in the computer system, and it is not indicated that Weber's theory can be effective in modelling of the problem.

2.3 Suggestions for change of techniques

Based on a literature survey of object-orientation, Bjornestad (1994) summarises the core of object-orientation to consist of the following:

- encapsulated objects with properties and behaviour,
- classes of objects, and
- inheritance of general properties and behaviour to specialised classes.

Monarchi and Puhr (1992) have surveyed object-oriented methods, and the methods seem to conform to the general core of object-orientation, with one exception: communication between objects is found in a majority of the methods in the survey. Somewhat surprisingly is aggregation only found in 5 of 19 methods, and 7 of the methods include constraints on structure, e.g., cardinalities.

Recently, the methods have adapted a larger number of concepts for modelling (e.g., Embley, et al. 1992; Martin and Odell 1992; Henderson-Sellers and Edwards 1994), and aggregation and constraints are included. However, no standard definition of aggregation has emerged (Motschnig-Pitrik 1994), so even this minor extension of object-orientation has not yet succeeded, nearly twenty years after it was suggested in data modelling (Smith and Smith 1977).

It is commonly assumed that the object-oriented model should represent the domain to be modelled. A step towards a more radical extension is found in the methods by Wirfs-Brock et al (1990), Jacobson et al (1992) and Reenskaug et al (1996). These methods suggest that the interaction between the user and the computer system should be the starting point for selection of objects rather than first achieving a model of the problem domain. Designing a system according to a desired human-computer interaction opens for modelling domains from different user viewpoints. However, the methods go for a unified model that is supposed to serve all interests, without separating between different viewpoints in the model.

2.4 Current trend: Method engineering

Conclusively, we have seen some explanations of modelling problems that extend object-oriented theories. However, neither these nor other theoretical contributions have extended the basis for deriving concepts for modelling. Consequently, the suggestions for improvements of techniques have been constrained by the implementation considerations. In addition, neither the theoretical considerations nor the techniques captures multiple perspectives on domains in the models. Taken together with Rumbaugh's story, this indicates that the way second generation object-oriented methods are developed conforms to method engineering rather than the system development research approach.

3 The domain of techniques

As we mentioned in the introduction, we have noticed a shift in the perspective in respect to the roles the computer systems may play in human work within organisations. Earlier, a common view of the computer was that it was used for handling or controlling the problem domain, hence the models did not address elements in the application domain explicitly. Lately there has been an increasing attention in both system development practice and in the research community toward using the computer as a medium in the work organisation, thus enabling the use of computers as means of coordinating work and communication in and about work (Simone and Schmidt 1993; Carstensen, et al. 1995).

We have in the previous section identified that the modelling domain of object oriented modelling techniques usually is the problem domain, and in some approaches the future computer system. Apart from the use-case technique (Jacobson, et al. 1992), we have not identified any object oriented modelling technique that explicitly address the application domain. However, Carstensen et al (1995) have applied an object oriented analysis technique to capture aspects of the application domain, and have reported problems with modelling interactions between actors involved in coordinating their activities. Other have reported problems related to modelling of different roles of actors in respect to the computer system (Richardson and Schwarz 1991; Coad 1992).

We want to apply oo modelling techniques in a human work context, where the computer has a role in interhuman communication, coordination of work and cooperation. With the problems mentioned above in mind we claim:

- that the notation and concepts for modelling either fails to or makes it difficult to model issues of work context, and
- that the theoretical foundations for oo techniques do not give clues as to what properties of the work context that should be modelled, and how this should be done.

According to our learning cycle, the problems are both on a theoretical and on a modelling technique level. We argue that the perspective on human work is fundamental to the selection and development of theoretical foundations for modelling.

3.1 Human work and modelling domains

We have reached a position to not model human work itself, but rather model the roles the computer have in the work. There are both political and theoretical arguments for this position. One the political side, we fear a de-skill of workers if the computer systems control the execution of work, in terms of what activities should be done, and in what sequence. On the theoretical side, we regard work to be to complex to model. Several schools of theories of work exist, and we use some of their findings to support this claim:

- Strauss (1988) reports that tasks and lines of work do not automatically arrange themselves in proper sequences or with proper scheduling, hence further work must be done in order to get the work done. This work, denoted articulation work, is driven by the situation at hand, and is often a result of contingencies. We regard it difficult to be explicit about articulation work itself, as would be necessary when trying to model it.
- Suchman (1987) reports that work is not strictly governed by plans, rather it is driven by the possibilities and limitations of the situation at hand. We regard a model of e.g. a work task or a routine a plan in this respect, hence problems related to ad-hoc arrangements in a practical setting are expected.
- Several persons are often involved in work, because several areas of competence are needed. This requires that they integrate and coordinate their individual activities in order to get the work done. When modelling the different actors of the application domain, and their roles should be made explicit in the model, to ease this coordination.

To summarise, we have identified a need for representing the different actors in the application domain, and a need to model the role of the computer in a work context. The issue of actors and roles is covered by a theoretical approach based on semiotic concepts, presented in the next section. The role of the computer is covered by an approach based on activity theory, presented in Section 2.5.

4 A learning cycle bringing in a semiotic relation

Problems of object oriented modelling of actors with roles (Kvisli 1993; Ressem 1995) and entities that have both form and content (Fog 1992) have been reported. Others have explained the problems of roles within the object-oriented perspective (e.g., Richardson and Schwarz 1991; Nerson 1992; Gottlob, et al. 1996). In order to explain both types of problems, we have searched for new ways to interpret the phenomena to be modelled. One approach has been to study the referential aspects of information systems.

Information systems are referential systems, because the data in information systems is perceived by their users to refer to things and events that are separated from the information system. Roles also exists in information systems e.g., persons playing the roles of users, and computer hardware playing roles of data processing units. We have therefore regarded the referential aspects of information systems as a domain for modelling, and we have seen how theories relevant to this area can contribute to the learning cycle of development of modelling techniques.

The data of information systems are expressions that refer to extensions, e.g., the object 'Diana Smith' in the airline reservation system refers to a specific passenger. In order to explain how the hardware of computers can play the role of data processing, one identifies layers of implementation, e.g., saying that an object is implemented in ASCII code, which is implemented in binary code, which is implemented in electronic circuits. During system design, one often has to construct programs at different layers. A similar separation into layers is also found in semiotics (Andersen 1990), where the form of expressions is realised in substance. E.g., letters are realised in black curves on white background. In order to explain roles in a broader framework than object-orientation, we have therefore adopted this semiotic form-substance relation.

In order to deal with other issues as well, e.g., entities that have both form and content, we have defined a more general relation. Since the substance has to exist for the form to exist, the relation is defined to capture this property, and it is called the «lifetime dependency» relation (Kaasbøll and Motschnig-Pitrik 1996). We have demonstrated that this relation is more general than previous solutions to role modelling, including those of Essink and Erhart (1991) and van de Weg and Engmann (1992), because it allows an object to be a role of several objects. This is not possible in previous approaches.

We have also started evaluating the lifetime dependency relation (Kaasbøll 1996). The hypotheses were that the relation occurred in most domain models, and that the models became less complex when using the relation. In the initial evaluation that we have carried out, we departed from object-oriented models of domains, and remodelled them using the lifetime dependency relation. This test showed a higher frequency of the relation than expected. We also achieved some reduction of complexity in the models through a decrease in the number of relations (ibid.).

The planned further evaluation includes modelling of systems that are going to be replaced. Also working out guidelines for implementation and suggestions for changes of programming tools and languages remain.

Although this learning cycle is not fully completed, it illustrates that problems of modelling can be explained in another context than object-orientation, and that such a widening of scope can contribute to new suggestions for changes in techniques. This may require more extensive implementation efforts than previous suggestions, which is the problematic side of trying to carry out system development research instead of method engineering.

5 A learning cycle bringing in activity theory

As we have mentioned, Carstensen et al (1995) have reported problems in modelling interaction between actors involved in coordinating their activities. We have not identified any theoretical foundation of object oriented modelling techniques that include issues of work coordination, communication or interaction among actors in the application domain. Therefore we have started the learning cycle with a search for a theory that address issues of work context and how computers are used in order to mediate communication, coordination and interaction. Activity theory was selected, and extended to explain the role of computers in human work. This work was done in a case study, in an organisation using Lotus Notes to communicate, share documents and coordinate work.

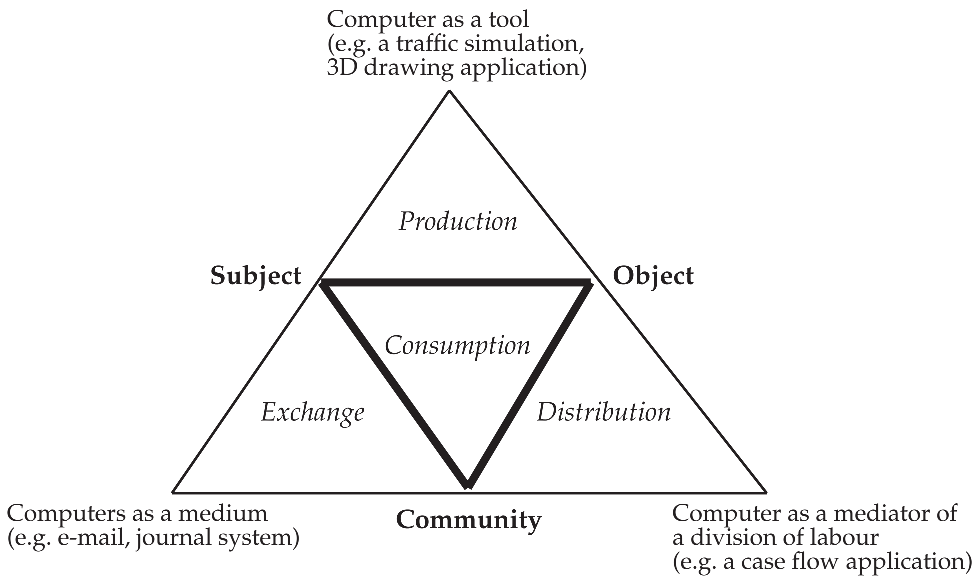
Activity theory address human work within a social context (Engeström 1987). The theory accounts for the individuals' relations to the object of work, and to the fellow workers. The relationships are not dual, but mediated through instruments, e.g. computers. We use the relationships as a basis for understanding the role of computers in an activity.

Production denotes the relationship between subject (a human) and object. The relationship is mediated through tools. The computer may be regarded a tool in this relation. See example in figure 2.

Distribution denotes the relationship between community (e.g. the workgroup or the employees in the organisation) and object. This relation is mediated through the division of labour. The computer may be regarded a mediator of this division of labour, in the sense that coordination of work may be done by means of the computer.

Exchange denotes the relationship between the a subject and the community. This relation is mediated through rules of social behaviour and communication. The computer may be regarded a communication channel in this relation. E-mail and conferencing software are examples of this role in the work context.

Figure 3. The aspects of a human activity, and the corresponding roles of a computer system.



According to Engeström, the three relations constitutes an organic whole. Issues of integration of the roles of the computer in a human activity is an issue (Fjuk, et al. 1995), and therefore the interdependencies of the various roles should be made explicit in the model.

The next step in this work is to do a systems development project, in order to derive hypothesis regarding how metaphors and notation in a object oriented modelling technique should be developed. This work will be carried out in a large Norwegian municipal organisation, (see Smørdal 1996). The learning cycle is restarted by testing the modelling technique in a practical setting.

6 Conclusion

New computer applications address issues of work context, in addition to representing the object of work. Object-oriented models of such systems therefore have to capture some aspects of work, e.g., actors and roles. Since human work is complex and governed by rules to a much lesser extent than computer processing is, the modelling techniques have to avoid making assumptions about regularities in work. Therefore object oriented modelling techniques should be developed according to knowledge about human work within organisations.

In order to point to how to bring such knowledge into the process of developing techniques, we have outlined a learning cycle consisting of practice, evaluation, theoretical contribution, and suggestion of improved techniques.

We have defined method engineering to be a way to develop methods, where formal theories and implementation considerations are used in evaluation, theoretical contribution, and as the basis for suggesting improvements in techniques. The literature indicates that most development of techniques for modelling has been carried out according to a method engineering approach.

In order to develop the techniques such that they can model issues related to work properly, knowledge of work has to be included in the ways modelling problems are explained and new modelling mechanisms are suggested. Therefore, we have argued to widen the theoretical scope of development of techniques from the focus on formal and implementation considerations in method engineering to a system development research learning cycle that is open for any contribution to understanding the domain that is to be modelled. We have illustrated the system development research approach with two cases of our own research, and shown that new concepts for modelling may emerge.

To develop these concepts into practical techniques, guidelines for implementation have to be worked out. For this part of the research, an engineering approach is probably well suited.

Even if we argue for widening the scope during development of techniques, we are aware that widely focused research into the techniques may lead to new knowledge that provides no clues as to how to improve the techniques. Instead, the research may, e.g., point to needs for better training or project organisation. We have introduced our contributions with an eye to the possibility for also using the wider focus for constructive suggestions to solve modelling problems. This points to the fact that choosing research approach is only one condition for setting the direction for development of techniques. The background of the researchers and their knowledge may be more decisive. Being aware of where to place ones development effort in method engineering or system development research may help to see the limits and possibilities of the effort.

Acknowledgements

This article has benefited from discussions with Frieder Nake and Markku Nurminen.

Biography

Jens Kaasbøll i assistent professor at the Department of Informatics, University of Oslo. He has published in Information Systems Journal, Journal of Object-Oriented Programming and in international conferences. He has served as a co-editor of Scandinavian Journal of Information Systems. His research interests are in object-oriented modelling and in ways of providing computer support in organizations such that the systems fit user tasks and seem integrated from users' points of view.

Ole Smørdal is a research associate in the Department of Informatics, University of Oslo. Current research interests include theoretical foundations for object oriented modelling in relation to information systems.

References

Andersen PB (1990) *A Theory of Computer Semiotics*. Cambridge University Press.

Bjornestad S (1994) A research programme for object-orientation. *European Journal of Information Systems* **3** (1), pp. 13-27.

Braa K and Vidgen R (1995) Action Case: Exploring The Middle Kingdom in IS Research Methods. In *Proceedings of Computers in Context: Joining Forces in Design* (Aarhus, Denmark).

Carstensen PH, Krogh B and Sørensen C (1995) Object oriented Modelling of Coordination Mechanisms. In Dahlbom B, Kämmerer F, Ljungberg F, Stage J and Sørensen C (eds.) *Proceedings of The 18th Information Systems Research Seminar in Scandinavia (IRIS'18)* (Gjern, Denmark), Gothenburg Studies in Informatics, Report 7.

Coad P (1992) Object-Oriented Patterns. *Communications of the ACM* **35** (9), pp. 152-9.

Embley DW, Kurtz BD and Woodfield SC (1992) *Object-Oriented Systems Analysis: A Model-Driven Approach*. Prentice-Hall, NJ.

Engeström Y (1987) *Learning by Expanding. An Activity-theoretical approach to developmental research*. Orienta-Konsultit Oy, Helsinki.

Essink LJB and Erhart WJ (1991) Object Modeling and System Dynamics in the Conceptualization Stages of Information Systems Development. In van Assche M and Rolland (eds.) *Object Oriented Approaches to Information systems*. North-Holland, Amsterdam, pp. 89-116.

Fjuk A, Sandahl T and Smørdal O (1995) Toward Incorporating Computer Applications in Cooperative Work Arrangements. In Dahlbom B, Ljungberg F, Stage J and Sørensen C (eds.) *Proceedings of The 18th Information Systems Research Seminar in Scandinavia (IRIS'18)* (Gjern, Denmark), Gothenburg Studies in Informatics, Report 7, pp. 159-69.

Fog C (1992) *A comparison of flow- and object-oriented analysis of information processing: Iterations and intuition in interpretation of large quantities of information*. Master thesis in Norwegian. Department of Informatics, University of Oslo.

Galliers RD (1992) Choosing Appropriate Information Systems Research Approaches: A Revised Taxonomy. In Galliers RD (ed.) *Information Systems Research: issues, methods and practical guidelines*. Blackwell Scientific, Oxford.

Gottlob G, Schrefl M and Rock B (1996) Extending Object-Oriented Systems with Roles. *ACM Transactions on Information Systems* **14** (3).

Harmsen F, Brinkkemper S and Oei H (1994) A language and tool for the engineering of situational methods for information systems development. In Zupancic and Wrycza (eds.) *The Fourth International Conference Information Systems Development (ISD'94) Methods & Tools. Theory & Practice* (Kranj), Moderna Organizacija, pp. 206-14.

Henderson-Sellers B and Edwards J (1994) *BOOKTWO of Object-Oriented Knowledge: The Working Object. Object-Oriented Software Engineering: Methods and Management*. Prentice-Hall, Sydney.

Heym M and Österle H (1993) Computer-aided methodology engineering. *Information and Software Technology* **35** (6/7), pp. 345-54.

Jacobson I, Christerson M, Johnson P and Övergaard G (1992) *Object oriented Software Engeneering. A Use Case Driven Approach*. Addison-Wesley.

Kumar K and Welke RJ (1992) Methodology engineering: A proposal for situation-specific methodology construction. In Cotterman and Senn (eds.) *Challenges and Strategies for Research in Systems Development*. John Wiley and Sons, Chichester, pp. 257-69.

Kvisli J (1993) *Object-oriented analysis and design of adminstrative computer applications*. Master thesis in Norwegian. Department of Informatics, University of Oslo.

Kaasbøll JJ (1996) Between controlled irrelevance and unrepeatable complexity: Initial evaluation of the concepts of domain modelling techniques. In *Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, The 8th Conference on Advanced Information Systems Engineering. Software engineering challenges in modern information systems. (CAiSE*96)* .

Kaasbøll JJ and Motschnig-Pitrik R (1996) Lifetime dependency relationships and their application to modelling roles and relationship objects. In Sutcliffe AG, Assche Fv and Benyon D (eds.) *Domain Knowledge for Interactive System Design*. Chapman & Hall, London.

Martin J and Odell JJ (1992) *Object-Oriented Analysis and Design*. Prentice-Hall.

Mathiassen L, Munk-Madsen A, Nielsen PA and Stage J (1993) *Objektorienteret analyse (in Danish)*. Marko, Aalborg.

Monarchi DE and Puhr GI (1992) A Research Typology for Object-Oriented Analysis and Design. *Communications of the ACM* **35** (9), pp. 35-47.

Motschnig-Pitrik R (1994) Analyzing the notions of attribute, aggregate, part, and member in data/knowledge modelling. In Zupancic and Wrycza (eds.) *The Fourth International Conference Information Systems Development (ISD'94) Methods & Tools. Theory & Practice* (Kranj), Moderna Organizacija, pp. 31-42.

Nerson J-M (1992) Applying Object Oriented Analysis and Design. *Communications of the ACM* **35** (9), pp. 63-74.

Pernici B (1990) Objects with Roles. *SIGOIS Bulletin* **11** (2/3), pp. 205-15.

Reenskaug T, Wold P and Lehne OA (1996) *Working With Objects: the OOram Software Engineering Method*. Manning, Greenwich.

Ressem JE (1995) *Where do all the objects come from? A study of the approach of three object-oriented methods to the identification of objects*. Master thesis in Norwegian. Department of Informatics, University of Oslo.

Richardson J and Schwarz P (1991) Aspects: Extending objects to support multiple, independent roles. *SIGMOD Record* **20** (2), pp. 298-307.

Rossi M and Brinkkemper S (1995) Metrics in Method Engineering. In Iivari, Lyytinen and Rossi (eds.) *Advanced Information Systems Engineering (CAiSE '95)* , Springer, **LNCS 932**.

Rumbaugh J (1995) OMT: The object model. *Journal of object-oriented programming* **January** , pp. 21-7.

Simone C and Schmidt K (1993) *Computational Mechanisms of Interaction for CSCW.* COMIC, Esprit Basic Research Project 6225, Lancaster University.

Smith JM and Smith DCP (1977) Database Abstractions: Aggregation and Generalization. *ACM Transactions on Database Systems* **2** (2), pp. 105-33.

Smørdal O (1996) Soft Objects Analysis, A modelling approach for analysis of interdependent work practices. In Patel D and Sun Y (eds.) *3rd International Conference on Object-Oriented Information Systems (OOIS'96)* (London, UK), Springer-Verlag, pp. 195-208.

Strauss A (1988) The Articulation of Project Work: An Organizational Process. *The Sociological Quarterly* **29** (2), pp. 163-78.

Suchman LA (1987) *Plans and Situated Actions*. Cambridge University Press.

van de Weg RLW and Engmann R (1992) A framework and Method for Object-Oriented Information Systems Analysis and Design. In Falkenberg ED, Rolland C and El-Sayed EN (eds.) *Information System Concepts: Improving the Understanding ISCO 2, IFIP Transactions A-4*. North-Holland, pp. 123-46.

Vessey I and Conger SA (1994) Requirements Specification: Learning Object, Process, and Data Methodologies. *Communications of the ACM* **37** (5), pp. 102-13.

Wirfs-Brock R, Wilkerson B and Wiener L (1990) *Designing Object-Oriented Software*. Prentice-Hall, NJ.