15

Method Engineering: Who's the Customer?

L. Mathiassen, A. Munk-Madsen, P. A. Nielsen and J. Stage
Department of Computer Science, Aalborg University
Fredrik Bajers Vej 7, DK-9220 Aalborg Ø, Denmark
{larsm, pan, jans}@iesd.auc.dk

Metodica, Nyvej 19, DK-1851 Fredriksberg C, Denmark
metodica@post4.tele.dk

Abstract

This paper reports from a large Danish effort to engineer an object-oriented method for analysis and design of computer systems. Over a period of six years a method was developed based on new ideas on how to learn object-orientation supplemented with well-known ideas of how to work object-oriented in systems development.

The experience from this method engineering effort is interpreted as an iterative process involving elements of theory, method and case records. These elements played different roles when engineering the method. But, what is more important, they became key elements in structuring and presenting the method to practitioners and students of the field.

This particular method engineering effort has thus been governed by a paradigm for learning methods rather than a paradigm for working with methods. We discuss this paradigm by exploring three issues involved in method engineering: (1) the relation between learning the method and working with the method; (2) the role of principles, patterns, and guidelines in explaining the method; and, finally, (3) the relation between concepts for reflection and modelling and concrete representations used to create texts and diagrams.

We suggest that the primary customers of method engineering are those studying methods eager to learn a class of new systems development practices. Those actually working with the method should be thought of in a secondary role when structuring and presenting a new method – even though they are the ultimate judges of the method's practical strengths and weaknesses.

Keywords

Method engineering, systems development, object-orientation, learning, working.

S. Brinkkemper et al. (eds.), *Method Engineering*

1 INTRODUCTION

One may take different approaches to method research and engineering: a comparative approach with particular focus on the features of methods (e.g. Olle *et al.* 1982, 1983 and 1986, Nielsen 1990a and 1990b), a tool-oriented approach with a particular focus on notation and CASE (e.g. Steinholtz *et al.* 1990, Andersen *et al.* 1991), or a mixed approach (e.g. Tolvanen and Lyytinen 1993, Verrijn-Stuart and Olle 1994). We have taken an experience-based approach as outlined in the following.

It has been the authors' privilege to work with systems development and systems development methods for a number of years, teaching methods to computer science and engineering students at the university and to practitioners in companies in the computing industry, experimenting with methods in companies and laboratories, and researching into the qualities of methods both in processes of learning them and working with them. This is reflected in our previous work (Mathiassen 1981, Stage 1989, Nielsen 1990a and 1990b, Kensing and Munk-Madsen 1993).

We have taught both structured and object-oriented methods to computer science and engineering students for several years, and during the last six years we have taught object-oriented methods to a large number of practitioners. In these efforts we have often experienced a gap between the way a specific method is structured and presented and the requirements and needs we experience when trying to convey the underlying new systems development practices to practitioners and students in the field. It is this dissatisfaction with the pedagogical weaknesses of many systems development methods that initiated the research reported in this paper.

For more than six years we have been involved with our students, companies in the computing industry and practitioners of systems development in engineering a new Danish method for object-oriented analysis and design, called OOA&D. The method is documented in two books (Mathiassen *et al.* 1993 and 1995). Evolving versions of the method have been taught to students and practitioners in different pedagogical contexts (university courses, open courses for practitioners, and in-house courses for specific companies). From this stems the experience that has driven the method engineering process: our method is based on new ideas on how to explain and learn object-orientation supplemented with well-known ideas of how to work object-oriented in systems development (Jackson 1983, Coad and Yourdon 1991, Booch 1991, Rumbaugh *et al.* 1991, Jacobson *et al.* 1992).

During the development of OOA&D two fundamental questions to method engineering have been addressed:

- What are the key elements of the method?
- What are the basic principles for structuring and presenting the method?

These questions are intrinsically related and answering them in our own method engineering effort required several iterations. Once the questions were answered the rest of the method fell in place more easily. In the engineering of OOA&D we made a fundamental decision, which helped us design the engineering process and answer the two questions:

- The underpinning paradigm of the method engineering process is that of learning and teaching object-oriented ideas to practitioners and students.

- The primary customers of the method engineering process are the practitioners and students that want to learn object-orientation.

This learning paradigm negates the conventional implicit assumption, that systems development methods should be structured and presented to reflect the way in which practitioners work when using the method.

The learning paradigm has had considerable implications for the engineering of our method and for the structure and documentation of the method. We suggest that this paradigm can help other method engineers overcome some of the difficulties involved in successfully engineering new methods. These difficulties include: hardship of teaching and learning the essentials of a method, difficulty in distinguishing the important differences between alternative methods, reluctance amongst practitioners and students to adhere to the method, and last but not least a slow adaptation of the method.

The structure of our discussion is as follows. The process through which we engineered OOA&D is described in Section 2. Our experience is interpreted as an iterative process involving elements of theory, method and case records as in (Checkland 1981). In Section 3, three major issues in designing OOA&D are then presented and discussed: learning and working with methods; the role of principles, patterns, and guidelines; and, finally, the relation between concepts for reflection and modelling and concrete representations used to create texts and diagrams. A summary of the learning paradigm and its implications for method engineering is given in Section 4.

2 THE ENGINEERING PROCESS

2.1 A Specific Case

Background: We have taught systems development methods to computer science and engineering students for a couple of decades. First, we taught a selection of state-of-the-art methods; then we used Jackson System Development (Jackson 1983) for a few years; later Modern Structured Analysis (Yourdon 1989) was used based on different course books; we have taught OOA and OOD (Coad and Yourdon 1991a, 1991b); and for the last three years, we have used different versions of our own method, OOA&D.

These methods are taught in a software engineering course introducing the students to software engineering in general and analysis and design of computer systems in particular. It is a one-term course with 20 sessions of lectures and related exercises of which the method part would cover more than half the sessions. The course runs in parallel to a programming course and a one-term project (half of the students' time) in which the students in groups of 6–7 would use the analysis and design method together with programming concepts and techniques to develop a computer system. Within the same term students are taught a method and required to use it for practical purposes. During the method engineering process this educational environment gave constant feed-back from the student projects.

In a different environment we have taught the same systems development methods to practitioners. Some of these activities have been general courses with participants from different companies and others have been in-house courses tailored to the needs of specific projects and

companies. In particular, starting in 1991, we have taught OOA and OOD based on Coad and Yourdon's method (1991a, 1991b). Typically the analysis part and the design part were taught in two separate 3-day courses with a series of lectures in combination with a mini-project in which the participants used the method on a small, but realistic case.

Experiences

We have different experiences teaching the various methods. Jackson System Development (Jackson 1983) was fairly easy to explain because of its emphasis on clear concepts, the elaborate examples, and the combination of well-defined activities and fundamental principles (e.g. model before function). The implementation part of the method was too elaborate with many technical details, the notion of entity was too simple, and some of the graphical representations of concepts were difficult for the students and practitioners to use.

Modern Structured Analysis (Yourdon 1989) is based on easy-to-understand concepts and intuitively appealing representations. The concepts and representations are, together with a few heuristics, the key elements of the method. There are no, or very few, fundamental principles. We found it easy to organize good lectures based on examples, but it was difficult for students and practitioners to combine the various models into coherent practical cases.

OOA and OOD (Coad and Yourdon 1991) attempt to overcome this difficulty by introducing object-orientation. The concepts are easy to understand, but the representations are less intuitive. Moreover, it was difficult for the students and practitioners to see how this method could cover all traditional aspects of systems analysis. The approach is heavily oriented towards data models, it is not obvious how services (or methods) are used during analysis, and many practitioners were missing the traditional function-oriented approach to requirements specification.

Approach

On the basis of these experiences we decided to develop a new method for object-oriented analysis and design in which we would combine the strengths and if possible avoid the weaknesses of these methods.

Our approach to this method engineering effort consisted of a number of elements. First, the process was designed to develop a series of versions of the method, to document each version as a series of transparencies together with a still more elaborate text, to use each version in different pedagogical environments, and to systematically collect feed-back from each of these teaching experiences. This process helped us develop an answer to one of the fundamental method engineering questions (What are the basic principles for structuring and presenting the method?)

Second, to answer the other fundamental method engineering question (What are the key elements of the method?) our approach included detailed studies of published methods on object-oriented analysis and design. Selected methods were studied in detail, their elements were critically analysed, similarities and differences between methods were identified, and based on this analysis, we chose principles, concepts and representations that we found potentially useful in our method.

Third, we wanted to make sure, that the resulting method emerged as a coherent whole, not as a mere collection of selected elements from other methods. To that end we designed our own conceptual framework explicating a specific perspective (or theory, cf. Section 2.2) on the processes and key products involved in object-oriented analysis and design: a model of a computer

system, a model of the context of a computer system, the fundamental concepts of objects and classes, the key activities involved in analysis and design, and, finally, an outline of the resulting documentation.

Finally, we stressed the development of realistic examples (or case records, cf. Section 2.2) of both the processes of analysis and design (i.e. chronological accounts and experiences) and of the resulting products (i.e. models, specifications, and documentation). These examples were typically developed through small experiments in which some authors were active while others observed and took notes.

The primary test-bed for the method engineering process was the different educational activities with students and practitioners, involving lectures, small exercises and large-scale student projects. Each time a version was tested, new insights were generated, and these insights were then accumulated as the key input to the development of the next version of the method.

Documenting each version of the method on transparencies allowed for relatively frequent changes and modifications. The complementary text describing elements of the method in greater detail were modified more seldom. In the end, the method was documented as two complete text books.

The method is presently being used as the standard text on object-oriented analysis and design in a number of Scandinavian universities and colleges. At the same time, the first large-scale industrial experiments have taken place.

2.2 A General Model

Stressing in method engineering, as we do, the context of learning the method at the expense of the context of working with the method, we run the risk of ending up with a wrongly balanced picture of the relation between methods and practices. To frame our discussion with a balanced view we use a general model by Anderton and Checkland (see Checkland 1981, p. 7–8). They argue convincingly that any development of a subject (in our case an object-oriented method including case records and underlying theories) has to be circular and has to contain the elements depicted in Figure 1.

In developing our method, the related area of reality (containing concerns, issues, problems, and aspirations) is object-oriented analysis and design practices. Together with other sources (e.g. experience with other types of analysis and design methods, general theories about design) these practices give rise to ideas about object-oriented analysis and design from which may be formulated substantive theories (about object-oriented models and systems) and methodological theories (about development of object-oriented models and systems). Such theories present problems (e.g. on how to understand the behaviour of objects) which may be analysed using models and related techniques.

The theories yield methodologies (in the context of this paper: systems development methods) which use the developed models and techniques. The methodologies are then used in action in the related area of reality. These applications of the methodology are documented in case records that support criticism of the theories.

This general model of development of a subject, or in this context, method engineering, emphasises a number of important points:

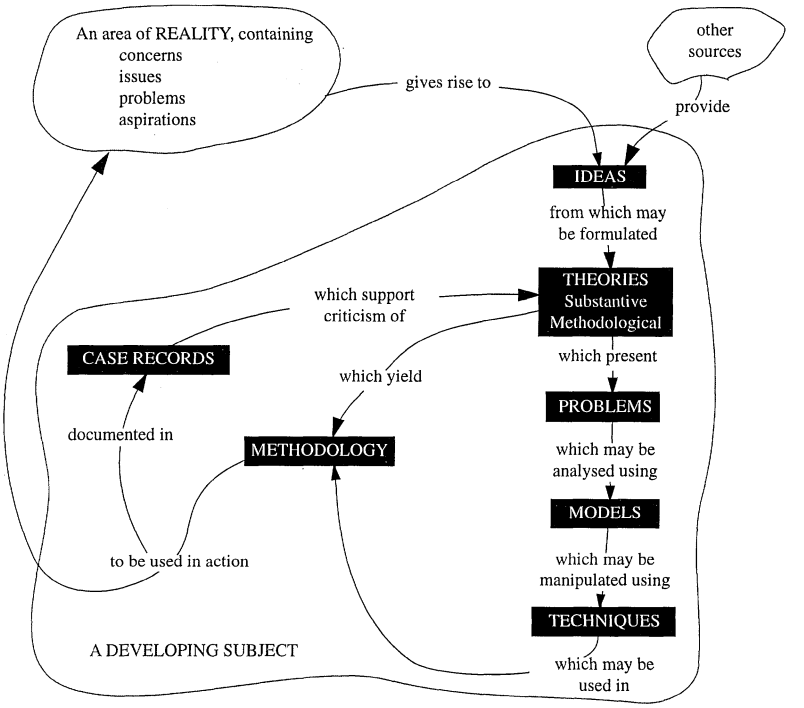

Figure 1 Anderton and Checkland's model of a developing subject (Checkland 1981, p. 8).

1. The ultimate source of inspiration and the ultimate test-bed for a systems development method is the related area of reality, i.e. the practices related to the method (including individual learning, organisational adaptation, and practical use in projects).
2. Systems development methods are based on (implicit or explicit) theories about the products and processes involved in working within the related area of reality (in our case object-oriented systems development).
3. Case records (documented practices of working with the method) play an important role in evaluating the underlying theories and the related models and techniques.

These three characteristics of the iterative process are all expressions of fundamental relations between methods and practices. In the development of our method they all played important roles, cf. Section 2.1. The last two points correspond quite directly to the approach taken in developing our method. In the following, we will elaborate on the first point by concentrating on the importance of learning a method (which is also an integral part of the concerns in the related area of reality) as opposed to mainly focusing on working with the method.

3 DESIGN ISSUES

Three major issues, all related to the learning paradigm, have played a significant role in the design of the method. This section will discuss these and thus explicate the difference and the relationship between taking a learning and a working paradigm.

3.1 Learning and Working

Soft Systems Methodology (SSM) is a method* employing systems concepts and ideas to change organisations. It is a very general method that has nothing in particular to do with the development of computer-based information systems. SSM was engineered or rather it evolved from the action research by Peter Checkland and his colleagues at the Systems Department at Lancaster University from 1969 to the present. The evolution can be described as in Figure 2.

This cyclic process based on working with the method in projects where students and researchers engage in real problems in organisations has been very successful. SSM has by now been used in hundreds of such projects, but it has also taken 25 years to bring SSM to its present form. Engineering† a method in this way makes the eliciting of experience from the practice of working with the method the core activity. The engineering approach is thus characterised by:

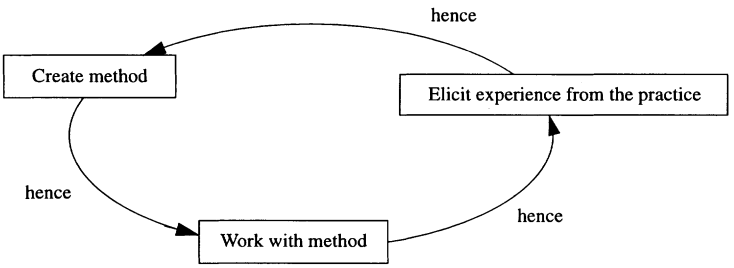

Figure 2 The evolution of a method through action research (Checkland 1981, p. 254)

- Experience is elicited from *working* with the method.
- There is a slow turn-around in the cyclic process as it is the real-world setting that forms the problems and therefore also the time-span.
- The usefulness of the method is judged on its strength as a working device in processes of development and change.
- It is difficult to evaluate 'working with a method' in real-world settings.

*Checkland actually makes a point of calling SSM a methodology (Checkland 1981, p. 162); but to avoid unnecessary confusion in this paper we call it a 'method' though he undoubtedly is right in his phrasing of the construct as something which is in between a philosophy and a technique.

†Engineering here to be understood in its broadest sense.

This model of method engineering is expanded in Figure 3 to embrace the method learning cycle. Figure 3 highlights the cyclic process where the learning of a method forms the practice from which experience is elicited such that the method can be re-created to become a better device for learning the method.

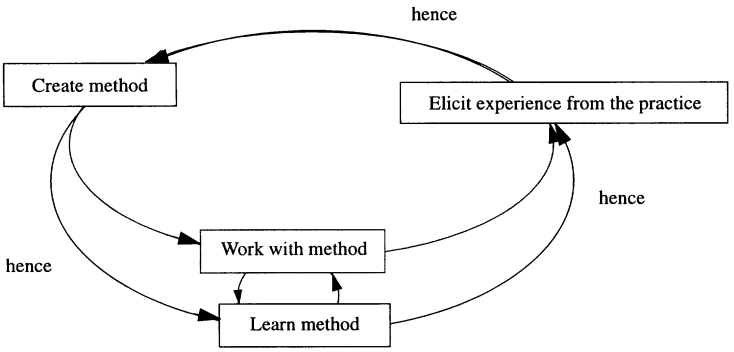

Figure 3 The learning cycle and its relationship with the working cycle

This expanded engineering approach is characterised by:

- Experience is elicited from *learning* the method.
- There is a fast turn-around in the cyclic process as it is the class-room setting that forms the problems and the time-span.
- The usefulness of the method is judged on its strengths as a learning device in processes of teaching and acquiring knowledge on the relationship between the method and practice.
- It is easier to evaluate 'learning a method' in a class-room setting.

To elaborate on this distinction between learning and working we take a 'method' to be an abstract account of possible ways in which activities in systems development may be performed. Viewed from the perspective of learning, a method is a framework for learning a class of systems development practices. Viewed from the perspective of working, a method is a set of guidelines and underlying theories for working within a class of systems development practices. Use of a method will in this sense encompass both the learning perspective and the working perspective. To acquire eloquence in a method one needs to be involved in both. Working with a method enables the learning of the method; and learning a method enables the work with the method. In both cases, as part of the practical use of a method, it is adapted to the specifics of the use situation and supplemented with other approaches.

In our engineering of OOA&D we have chosen a basic structure and presentation primarily suited for learning. This is in part due to the fact we have to define the structure in which the method is presented while working with the method may well follow various structures depending on the specifics of the situation. A book has a fixed structure, a series of lectures and exer-

cises has a fixed structure and even a single lecture has a fixed structure. A pertinent question for us as method engineers was therefore: In which sequence should the story of object-orientation be told? We chose a sequence which we followed in both the transparencies documenting the courses and in the text-book. For example, from our learning perspective one has to learn about objects and classes and how to take decisions on which classes to include in a model and which to exclude before anything else. Then one can learn about object and class structures and how to model such structures. Then one can learn about object dynamics and how to model that in classes. We could, of course, have chosen another sequence based on other pedagogical ideas.

In taking these fundamental decisions on sequence the method engineer has to apply pedagogical arguments rather than arguments of the type "this is how it works in practice". A presentation of a method is like a story with good explanations of the complexities of its subject (like object-orientation in analysis and design) and with explanations that are understandable by novices with respect to the subject (e.g. students and practitioners not mastering object-orientation). Different pedagogies may be chosen, but it is still a pedagogical decision.

In the presentation of OOA&D we have made a particular point of giving such a sequence that is good for learning our method; but we have also explained in more general terms and using examples how that sequence differs from actually working with OOA&D in practice.

A method engineer should not forget the working perspective and neglect the experience elicited from the practice of working with the method. Any presentation and good explanation of a method will also have to encompass guidance on how to relate learning and working. The crux of the matter is that the basic structure of a method should primarily be suitable for learning and secondarily for working.

3.2 Principles, Patterns and Guidelines

A method contains descriptions of processes and products involved in the related class of systems development practices. These descriptions can be given in reasonably abstract forms. Guidelines are rather concrete expressions of ways of doing things. They typically include ways of representing concepts combined with techniques or procedures for how to apply these representations to develop models. Principles and patterns are more abstract descriptions. Principles are understood as abstract accounts of approaches to specific processes. Patterns are abstract descriptions of partial solutions to modelling the products.

In the engineering of OOA&D and most prominently in the presentation of OOA&D, guidelines are, of course, used to illustrate and explain. But principles and patterns are at the very centre. Each of the core chapters in the text-books and each of the core lectures in the courses explains an activity in the method. Figure 4 shows part of the front-page of Chapter 5 in the text-book on analysis. This chapter explains the activity 'Dynamics' and it puts strong emphasis on the purpose of the activity, the concepts to be explained in the chapter as well as being used during performing the activity, the principles giving abstract accounts of processes, and the product of the activity.

Similarly, patterns are frequently used to explain solutions to modelling problems or product development. The activities on architectural design are mainly explained in terms of different patterns (e.g. layered architecture and client-server architecture). Another chapter gives a pattern (template) for the analysis and design documentation and these are complemented with specific examples of full analysis and design documents. Basically, there are two ways of using patterns in a description of a method: as exemplars and masterpieces to be plagiarised, and as gen-

Dynamics

Purpose	• To describe the dynamics of the object system i terms of behaviour of objects.
Concepts	• Event lifecycle: A concrete sequence of events which an object during a particular time-span is involved in. • Behavioural pattern: An abstract pattern of events that stipulates the possible and desirable event lifecycles for all objects in a class. • Attribute: The name of a data property for a class or an event.
Principles	• Describe the behaviour of objects by a behavioural pattern for their class. • Consider particularly events which are common for several objects. • Deduce the attributes for a class from its behaioural pattern.
Results	• Behavioural pattern and attributes for each class.

Figure 4 Sample of a front-page from Chapter 5 in (Mathiassen *et al.* 1993)

erally applicable solutions that have been shown to solve partial problems. Principles and patterns supplement each other nicely as the principles provide abstract accounts to guide and understand processes and the patterns provide abstract accounts of (partial) products without any of the two over-emphasising particular ways of performance.

Principles and patterns are important means for learning a systems development method. Guidelines are useful, of course, as they provide possible concrete ways to apply principles and patterns. What matters, though, is often what is done, not precisely how it is done. Explanation in terms of principles and patterns focuses on the important and the essential rather than on the many and sometimes obscure details. Moreover, such explanations encourage the method engineer to explicate the ideas, theories, and problems underlying the method, cf. Figure 1. This insight will help practitioners and students trying to learn the method to understand and appreciate, not only what should be done, but also why it should be done. For practitioners working with the method this level of appreciation is useful to understand how the method can be combined with other methods and tailored to different situations, maintaining the essentials and reshaping the specifics.

3.3 Concepts and Representations

A systems development method supports the modelling of computer-based systems through concepts and means of representation. From a working perspective it is important to provide a full notation and all that comes with a notation, e.g. symbols, semantics, rules for applying the symbols, ways of manipulating texts or diagrams written with the notation. This embodies both concepts and means of representation, but with a leaning towards the means of representation.

From a learning perspective concepts are more important elements of a method than the related means of representation. Means of representation are not neglected, but they play a different role in learning the modelling techniques compared to the concepts. In particular, specific means of representation are needed to explain and illustrate a method. This is done for various reasons: examples of products are given in a representation, ways of modelling and modifying models are given in a representation, etc. In presenting a method it is useful to adhere to a few, preferably coherent representations.

In the engineering of OOA&D we put more emphasis on which concepts we chose to be part of the method than on the representation of the concepts in models. Our means of representation were taken from (Coad and Yourdon 1991a, Jackson 1983, Harel 1987, Jacobson *et al.* 1992). What we didn't find there we invented our own representations of. Having decided which concepts it should be possible to use in creating a model it was fairly easy to go through the available notations used by others and find suitable representations.

After having written the two text-books we were asked by a company within telecommunications and associated with major foreign companies all using Rumbaugh *et al.*'s method (1991) to give a course on object-oriented analysis. They wanted to stick to Rumbaugh *et al.*'s means of representation but wanted our method in the learning of object-oriented practices. Because the key elements of OOA&D are independent of the chosen means of representation it was fairly easy to change the means of representation in a whole 3-day course to those used by Rumbaugh *et al.* It took 7 hours to change the transparencies and none of the concepts in OOA&D had to be changed in the process.

All in all, concepts are more important elements of a method than the related means of representation.

4 SUMMARY

This paper has reported experiences from a method engineering effort that was governed by a paradigm for learning methods rather than a paradigm for working with methods. We have discussed this paradigm by exploring three issues involved in method engineering: (1) the relation between learning the method and working with the method; (2) the role of principles, patterns, and guidelines in explaining the method; and, finally, (3) the relation between concepts for reflection and modelling and concrete representations used to create texts and diagrams. The main points to consider for other method engineers are:

- When experience is elicited from learning the method there is a fast turn-around in the cyclic process of development and it becomes easier to evaluate new versions of the method.

- Principles and patterns are important means for learning a systems development method. Guidelines are useful, of course, as they provide possible concrete ways to apply principles and patterns. But what matters is often what is done and why it is done, not precisely how it is done.
- From a learning perspective concepts are more important elements of a method than the related means of representation. In addition, when the key elements of a method are independent of the chosen means of representation it was fairly easy to change the means of representation when adapting the method to new situations.

In summary, we suggest that the primary customers of method engineering are those studying methods eager to learn a class of new systems development practices. Those actually working with the method should only play a secondary role in structuring and presenting a new method – even though they, of course, are the ultimate judges of the method's practical strengths and weaknesses.

5 REFERENCES

Andersen, R., J. A. Bubenko Jr. and A. Sølvberg (1991). *Advanced Information Systems Engineering. Proceedings from CAiSE '91*. Springer-Verlag, Berlin.

Booch, G. (1991). *Object-Oriented Design with Applications*. Benjamin/Cummings, Redwood City, California.

Checkland, P. B. (1981). *Systems Thinking, Systems Practice*. Wiley, Chichester.

Coad, P. and E. Yourdon (1991a). *Object Oriented Analysis*. Prentice-Hall, New York. 2nd edition.

Coad, P. and E. Yourdon (1991b). *Object Oriented Design*. Prentice-Hall, New York.

Fichman, R. G. and C. F. Kemerer (1993). Adoption of Software Engineering Process Innovations: The Case of Object Orientation. *Sloan Management Review*, **34**, 2, 7-22.

Harel, D. (1987). Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, **8**, 231–274.

Jackson, M. (1983). *System Development*. Prentice-Hall, Englewood Cliffs, New Jersey.

Jacobson, I., M. Christerson, P. Jonsson and G. Övergaard (1992). *Object-Oriented Software Engineering*. Addison-Wesley, Wokingham.

Kensing, F. and A. Munk-Madsen (1993). Participatory Design: Structure in the Toolbox. *Comm. ACM*, **36**, 6, 78–85.

Mathiassen, L. (1981). *Systems Development and Systems Development Method*. In Danish. Dr.scient. Thesis, Oslo University.

Mathiassen, L., A. Munk-Madsen, P. A. Nielsen and J. Stage (1991). Soft Systems in Software Design, in *Systems Thinking in Europe* (eds. M. C. Jackson *et al.*), 311–317, Plenum Press, New York.

Mathiassen, L., A. Munk-Madsen, P. A. Nielsen and J. Stage (1993). *Object-Oriented Analysis*. In Danish. Marko, Aalborg.

Mathiassen, L., A. Munk-Madsen, P. A. Nielsen and J. Stage (1995). *Object-Oriented Design*. In Danish. Marko, Aalborg.

Nielsen, P. A. (1990a). Approaches for Appreciating information systems methodologies: A soft systems survey. *Scandinavian Journal of Information Systems*, **2**.

Nielsen, P. A. (1990b).*Using and Learning IS Development Methodologies*. Ph.D. Thesis, Lancaster University.

Olle, T. W., H. G. Sol and A. A. Verrijn-Stuart, editors (1982). *Information Systems Design Methodologies: A Comparative Review*. North-Holland, Amsterdam.

Olle, T. W., H. G. Sol and C. J. Tully, editors (1983). *Information Systems Design Methodologies: A A Feature Analysis*. North-Holland, Amsterdam.

Olle, T. W., H. G. Sol and A. A. Verrijn-Stuart, editors (1986). *Information Systems Design Methodologies: Improving the Practice*. North-Holland, Amsterdam.

Rumbaugh, J., M. Blaha, W. Premerlani, S. Eddy and W. Lorensen (1991). *Object-Oriented Modelling and Design*. Prentice-Hall, Englewood Cliffs, New Jersey.

Stage, J. (1989). *Between Tradition and Transcendence: Analysis and design in systems development*. In Danish. Dr.scient. Thesis, Oslo University.

Steinholtz, A. Sølvberg and L. Bergman (1990). *Advanced Information Systems Engineering. Proceedings from CAiSE '90*. Springer-Verlag, Berlin.

Tolvanen, J.-P. and K. Lyytinen (1993). Flexible Method Adaptation in CASE: The metamodeling approach. *Scandinavian Journal of Information Systems*, **5**, 51–78.

Verrijn-Stuart, A. A. and T. W. Olle, editors (1994). *Methods and Associated Tools for the Information Systems Life Cycle*. North-Holland, Amsterdam.

Yourdon, E. (1989). *Modern Structured Analysis*. Prentice-Hall, New York.

6 BIOGRAPHY

Lars Mathiassen is currently professor in Information Systems at the Department of Computer Science, Institute for Electronic Systems, at Aalborg University. For the last twenty years he has done research in the intersection between software engineering and iInformation systems. His reaearch interests include obejt-oriented software engineering, risk-based project management, IT management and strategy, and the philosophy of computing. Lars Mathiassen has published several scientific papers on these subjects and he has co-authored a number of books on software engineering and information systems, including two books in Danish on object-oriented analysis and design.

Andreas Munk-Madsen is partner in Metodica, a Copenhagen-based company specialised in software methods. He has a broad experience in research, education, and consultancy. He has a PhD in computer science. He is co-author of several books on system development methods and is currently completing a book on strategic project management. His current interests include requirement management, project management, methods implementation, and object-oriented analysis and design.

Peter Axel Nielsen is currently associate professor in Information Systems at the Department of Computer Science, Institute for Electronic Systems, at Aalborg University. Over the past years he has been engaged in understanding the use of information systems development methodologies. His research interests include analysis and design techniques, object-orientation, domain modelling, the modelling process with a particular focus on recurrent and reusable

patterns as well as IT management. He is currently the editor of Scandinavian Journal of Information Systems. Peter Axel Nielsen is co-author of two books in Danish on object-oriented analysis and design.

Jan Stage is currently associate professor in Information Systems at the Department of Computer Science, Institute for Electronic Systems, at Aalborg University. His research interests include theoretical and methodological aspects of information systems development, especially development of new object-oriented methods and techniques for analysis and design. He is teaching graduate and undergraduate students in computer science and information systems and giving industry courses on analysis, design, and programming for software professionals. Jan Stage is co-author of two books in Danish on object-oriented analysis and design.