12

Soft Systems in Software Design*

Lars Mathiassen
Andreas Munk-Madsen
Peter A. Nielsen
Jan Stage

Introduction

This paper explores the possibility of applying soft systems thinking as a basis for designing application software and it outlines a new method for software design (Mathiassen *et al.* 1991). The method is called "Rapid Systems Modeling". It supports systems developers and users in going from a problematic organizational situation to the design of a new and modified computer application for that situation.

Rapid Systems Modeling combines a set of widely appreciated principles and methods into one coherent framework. The approach taken to design emphasizes learning as in Soft Systems Methodology (Checkland 1981). Rapid Systems Modeling combines this approach to learning with techniques and tools for modeling and experimenting with systems based on object-oriented thinking and the use of prototypes (Birtwistle *et al.*, 1973; Jackson 1983; Coad *et al.* 1990; Budde *et al.* 1984). The use of Rapid Systems Modeling is controlled through risk management (Boehm 1988, 1989) and a coherent design proposal is produced based on the idea of faking a rational design process (Parnas *et al.* 1986).

Until now, Soft Systems Methodology has been used in various organizational settings and disciplines (Checkland 1981; Checkland *et al.* 1990). Attempts have also been made to adapt soft systems ideas to information systems development, cf. (Wilson 1984; Wood-Harper *et al.* 1985; Avison *et al.*, 1990; Stowell *et al.* 1990). All of

* Published as: Soft Systems in Software Design. L. Mathiassen, A. Munk-Madsen, P. A. Nielsen & J. Stage. In: Systems Thinking in Europe, M. C. Jackson et. al. (Eds.), Plenum Press, 1991.

these efforts are concerned with organizational change and the modeling involved is based on rigorous use of *human* activity systems. This paper reports from ongoing research where we attempt to adapt and supplement soft systems ideas to make them useful in a specific *technical* domain, i.e. the design of computer applications as an integral part of organizational change.

Exploring soft systems ideas

We start by briefly reviewing soft systems ideas in relation to traditional software design methods. On that basis we discuss possibilities for adapting soft systems ideas to the design of computer applications.

Idea 1: Systems as intellectual constructs

The very idea of Soft Systems Methodology is that we may inquire into a problematic situation by means of the notion of system. Systems are intellectual constructs making explicit our subjective meanings attributed to reality and our visions about reality. Multiple perceptions are exploited to learn about and eventually improve a problematic situation.

In software engineering the term “system” is seldom defined, but computer applications are taken to *be* systems. The term “system” gets its semantics implicitly through a set of tools and techniques for specification of computer systems. Well-known examples are Structured Analysis/Structured Design (DeMarco 1979; Yourdon 1989), Jackson Systems Development (Jackson 1983), and Object-Oriented Analysis (Coad *et al.* 1990). The missing or weak distinction within software engineering between the world of phenomena and the world of perceptions has practical consequences. Traditionally, software engineers conceive a computer application as restricted to the automatic execution of the corresponding program on a computer disregarding the perceptions and actions of users. Concerns are separated and the computer system is thought of as something in itself. The designers' task is reduced to specification of a program meeting pre-defined and stable requirements (Floyd 1987).

The development of Rapid Systems Modeling is rooted in a tradition where systems consistently have been viewed as intellectual constructs dialectically related to the phenomena of computer appli-

cations. Thus, the exploitation of multiple viewpoints on the same computer application has been emphasized (Mathiassen 1981; Nygaard *et al.* 1987; Stage 1989; Nielsen 1990). In Rapid Systems Modeling we take the position that the idea of "systems as intellectual constructs" is applicable not only in learning about human activity but also in designing computer applications.

Idea 2: Learning through action and reflection

Soft Systems Methodology is based on the idea that effective learning takes place as an interaction between real world activities and thinking about the real world in terms of systems. The problematic situation is experienced and expressed, different systems are defined and modeled, and these models are then in turn confronted with the real situation. In this way, models are used to structure and orchestrate a debate amongst actors in the situation with the purpose of learning about the problematic situation.

Conventional software development methods strongly emphasize description, specification and modeling of the system to-be. There is a number of widely accepted techniques for evaluating specifications, e.g. structured walk-throughs and reviews, cf. (Freedman *et al.* 1982). But only a small number of techniques are provided to express situational characteristics in an informal and loosely structured way. Some of the rare examples are: event lists (Yourdon 1989) and lists of nouns and verbs (Jackson 1983). Generally, there is a growing appreciation of the idea of learning in software engineering, but there are still few frameworks that utilizes the relationship between action and reflection in a systematic way. Instead, there seem to be two competing strategies: the specification approach relying strongly on reflection before action, and the prototype approach relying mainly on experiments (actions) without emphasizing systematic reflection (Mathiassen *et al.* 1990).

Rapid Systems Modeling is based on the idea that software design requires learning and methods should thus support this by exploiting the relationship between action and reflection. One example of the application of this idea is the Spiral Model (Boehm 1988). Rapid Systems Modeling rejects the standpoint that prototypes and specifications represents two competing strategies. Instead, prototypes and specifications are seen as two complementary ways of expressing reflection in software design.

Idea 3: Systems as wholes

At the heart of soft systems thinking is the principle that whole entities exhibit emergent properties which are meaningful only when attributed to the whole, not to its parts. In this sense, Soft Systems Methodology utilizes holistic thinking. A conceptual distinction is made between what a system is (emergent properties) and what it does (constituent activities and relationships), and a practical distinction is made between defining the system and modeling its activities.

Methods for software development emphasize detailed and elaborate specification of systems. The methods distinguish between different levels of abstraction and different aspects, e.g. data flow and data definition. But overview and detail is provided without explicit conception of the system as a whole. A few methods suggest to define the purpose of the computer system, e.g. “statement of purpose” in Yourdon's modern version of Structured Analysis/Structured Design (Yourdon 1989). Despite this, traditional software engineering methods support development of reductionistic models.

Rapid Systems Modeling supports designers in defining emergent properties of the systems explicitly *in addition* to modeling their contents. This is in accordance with the ideas behind Soft Systems Methodology. This position is further discussed in the following two sections.

Idea 4: Defining systems

One of the main activities of Soft Systems Methodology is the definition of systems by formulation of root definitions. A root definition is a precise description of the emergent properties of a system. It is suggested that a root definition should contain the CATWOE elements explicitly: Customers, Actors, Transformation, Weltanschauung, Owner, and Environment. These six elements are closely connected to the idea of human activity systems.

In Rapid Systems Modeling, systems are to be defined in a similar way. The exact form of a definition is yet to be found, but certain differences seem obvious. Firstly, when understanding computer applications “transformation” is questionable as the key aspect of a system. The strong interactive nature of modern computer applications suggests metaphors like “actor”, “agent”, “medium”, and “tool” each implying somewhat different systems concepts. Sec-

ondly, the notion of Weltanschauung plays a crucial role in Rapid Systems Modeling, but it needs to be specifically oriented towards the assumptions underlying a particular computer system and its relation to wider human activity systems. Thirdly, the other CATWOE elements have to be reconsidered and possibly supplemented by other aspects relevant to the technical domain of computer applications, e.g. interface facilities and technological platform.

Idea 5: Modeling systems

In Soft Systems Methodology, a conceptual model contains the minimal set of related (human) activities needed to carry out the transformation described in the corresponding root definition. A system is thought of as being adaptive. A set of monitoring and controlling activities are therefore included in each model. The conceptual model must be defensible against the root definition and vice versa.

In Rapid Systems Modeling each system is going to be modeled. The models are evaluated and compared with the purpose of eventually arriving at a design proposal. The flavor of the models in Soft Systems Methodology is inherited, but the models have to be different from conceptual models to support reflection on the technical domain. As a consequence, the method supports the use of two types of models: object-oriented specifications and prototypes. The method also recommends to use different versions of models displaying different levels of detail. The purpose is to support organizational and technical learning and to facilitate choice among alternative systems.

Outline of Rapid Systems Modeling

The software design method, Rapid Systems Modeling, combines and adapts already established ideas and methods about learning, modeling and management. The ideas and methods are combined and projected into the domain of designing computer applications for specific organizational settings. We are in the midst of trying our ideas in practice and in education. This, in turn, will reshape the proposed method and hopefully make it more useful. In the following we present a first version of the method based on our experience

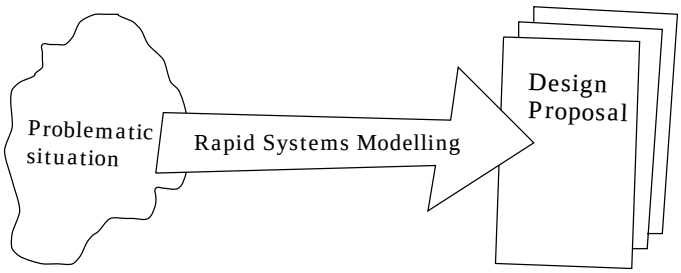

Figure 1. The overall transformation of Rapid Systems Modeling.

with each of the ideas and methods underlying Rapid Systems Modeling.

Overall transformation and basic activities

The area of concern is analysis and design of computer applications. We are interested in supporting systems developers and users in going from an unstructured organizational situation with an expressed need for improved application of computers to an agreed-upon proposal for a new or modified computer application. This overall transformation of Rapid Systems Modeling is illustrated in figure 1.

Our approach to this transformation is shown in figure 2. Rapid Systems Modeling consists of three strongly related activities. The method emphasizes learning about the problematic situation, technical possibilities in terms of computer systems, and the relationship to the organizational setting. The approach to learning is Soft Systems Methodology, but the specific techniques are adapted from software development. Learning is in our view a necessary and highly underrated activity in software design, see (Floyd 1987).

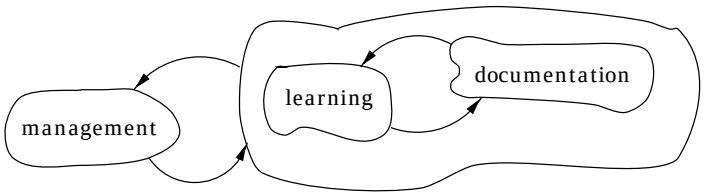

Figure 2. The main elements of Rapid Systems Modeling.

Still, other activities are also important: proper documentation is crucial and effective management of resources is required.

Modeling based on prototypes and object-orientation

Our approach to learning is illustrated in figure 3. In applying Rapid Systems Modeling, several concrete learning processes are initiated. The initiation of a learning process is based on management considerations. Learning processes can be performed in parallel each requiring different amounts of resources and applying different types of modeling techniques.

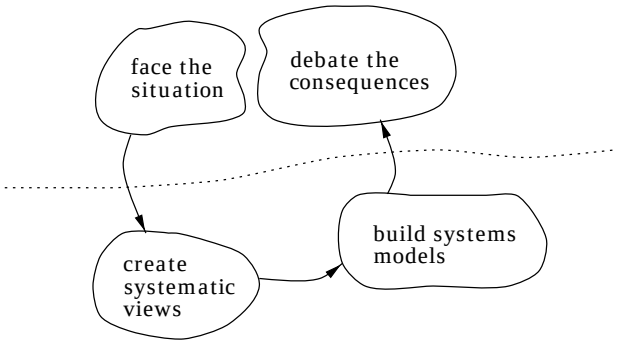

Figure 3. The learning activities of Rapid Systems Modeling.

Rapid Systems Modeling supports learning processes based on object-oriented specification. Object-oriented thinking supports designers in dealing with complexity by extracting in a condensed form the fundamental properties of a computer system. Techniques for defining systems and for modeling them as interacting sets of objects are provided. The specific outlook of well-formulated root definitions is still to be found. The object-oriented models are based on an integration of the abstract datatype approach of Object-Oriented Analysis (Coad *et al.* 1990) and the communicating sequential processes approach of Jackson Systems Development (Jackson 1983). Our approach to object-oriented modeling utilizes the encapsulation and abstraction mechanisms and the focus on data relationships as suggested by Coad and Yourdon, and the application of the idea of simulation of the real world as suggested by Jackson.

Rapid Systems Modeling supports learning processes based on prototyping. Prototyping supports learning about the practical effect

of specific design proposals. Rapid Systems Modeling provides techniques for defining systems and techniques for modeling these as computer-based prototypes. Also in this context, the specific outlook of well-formulated root definitions has yet to be found, though initial experiments have been performed (Bondgård *et al.* 1990). A significant aspect of this type of learning process is to ensure the systematic experimentation with use of prototypes in realistic settings.

Producing a consistent outcome

Proper documentation is crucial in software development. The outcome of Rapid Systems Modeling is a design proposal to be used as the basis for further development. The documentation is used as the basis for technical design and implementation, it is used to support division of labor between systems developers, and it plays a key role in assuring a satisfactory quality of the final computer application. The documentation activity of Rapid Systems Modeling evaluates and documents relevant insights gained through the learning activities. The approach taken is inspired by Parnas and Clements (Parnas *et al.* 1986) and principles for the resulting design document are being developed (Parnas 1972; Parnas *et al.* 1985, 1986; Stage 1989).

Managing risks

Learning processes are in general as well as in software development intrinsically open and experimental in nature, see for example (Floyd 1987). At the same time, software development is a resource demanding activity and effective management is required. Risk management, as proposed by Boehm (1988, 1989) offers an approach to management in software development that seems to handle this dilemma. Focus is on situational risks, i.e. uncertainties and complexities involved in deciding on relevant and useful actions. Techniques are provided for identification of risks (i.e. a need for learning) in the design situation, for assigning priorities to identified risks, and for practical planning of learning processes to resolve risks. A specific version of this approach is being developed that also involves monitoring and controlling the learning and documentation activities (Larsen *et al.* 1991).

Summary

The purpose of this paper has been to argue for the application of soft systems ideas in relation to software design. We have done this by discussing some of the fundamental aspects of soft systems thinking in relation to software design. The argument has been further substantiated by outlining how soft systems ideas can be supplemented and projected into this area of technical and organizational change.

The paper reports from an ongoing research program of which the basic assumptions and ideas have been expressed at an overall level. Substantial questions and many details are yet to be investigated.

The research behind this paper has been partially financed by the Danish Natural Science Research Counsel, Program No. 11-8394.

References

Avison, D. & A. T. Wood-Harper (1990): *Multiview: An Exploration in Information Systems Development*. Oxford: Blackwell Scientific Publications.

Birtwistle, G. M., O. J. Dahl, B. Myrhhaug & K. Nygaard (1973): *Simula BEGIN*. Lund and New York: Studentlitteratur and Petrocelli/Charter.

Boehm, B. W. (1988): A Spiral Model of Software Development and Enhancement. *Computer*, May.

Boehm, B. W. (1989): *Software Risk Management*. Washington, D. C.: IEEE Computer Society Press.

Bondgård, P., E. Degn & K. Vraagaard (1990): Prototyping: Understanding and Change. Master's thesis, Institute for Electronic Systems, Aalborg University. (In Danish)

Budde, R., K. Kuhlenkamp, L. Mathiassen & H. Züllighoven (Eds.): *Approaches to Prototyping*. Berlin: Springer-Verlag.

Checkland, P. B. & J. Scholes (1990): *Soft Systems Methodology in Action*. Chichester: Wiley.

Checkland, P. B. (1981): *Systems Thinking, Systems Practice*. Chichester: John Wiley and Sons.

Coad, P. & E. Yourdon (1990): *Object-Oriented Analysis*. Englewood Cliffs, New Jersey: Yourdon Press and Prentice-Hall.

DeMarco, T. (1979): *Structured Analysis and System Specification*. Englewood Cliffs, New Jersey: Yourdon Press and Prentice-Hall.

Floyd, C. (1987): Outline of a Paradigm Change in Software Engineering. (191–210) in G. Bjerknes *et al.* (Eds.): *Computers and Democracy.* Avebury: Aldershot.

Freedman, D. P. & G. M. Weinberg (1982): *Handbook of Walkthroughs, Inspections, and Technical Reviews.* Boston: Little, Brown and Company.

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

Larsen, T., C. Millum, H. Solberg & F. Tolstrup (1991): A Risk-based Model for Designing Information Systems. Master's thesis, Institute for Electronic Systems, Aalborg University. (In Danish)

Mathiassen, L. & J. Stage (1990): Complexity and Uncertainty in Software Design. (482–489) in *Proceedings of the IEEE International Conference on Computer Systems and Software Engineering.* Washington DC: IEEE Computer Society Press.

Mathiassen, L., A. Munk-Madsen, P. A. Nielsen & J. Stage (1991): Rapid Systems Modelling: The Soul of a New Method. In *Proceedings of the Fourteenth Information Systems Research Seminar in Scandinavia.*, Institute for Electronic Systems, Aalborg University, February.

Mathiassen, L. (1981): *Systems Development and Systems Development Method.* Ph.D. thesis, Oslo University. (In Danish)

Nielsen, P. A. (1990): *Learning and Using Methodologies in Information Systems Analysis and Design.* Ph.D. thesis, Department of Systems and Information Management, Lancaster University, July.

Nygaard, K. & P. Sørgaard (1987): The Perspective Concept in Informatics. (371–393) in G. Bjerknes *et al.* (Eds.): *Computers and Democracy.* Avebury: Aldershot.

Parnas, D. L. & P. C. Clements (1986): A Rational Design Process: How and Why to Fake It. *IEEE Transactions on Software Engineering,* Vol. 12, No. 2 (251–257).

Parnas, D. L., P. C. Clements & D. M. Weiss (1985): The Modular Structure of Complex Systems. *IEEE Transactions on Software Engineering,* Vol. 11, No. 3 (259–266),

Parnas, D. L. (1972): On the Criteria to be Used in Decomposing Systems into Modules. *Comm. ACM.* Vol. 15, No. 12 (1053–1058),

Stage, J. (1989): *Between Tradition and Transcendence. Analysis and Design in Systems Development.* Ph.D. thesis, Institute for Electronic Systems, Aalborg University.

Stowell, F. A., P. Holland, P. Muller & R. Prior (1990): Applications of SSM in Information Systems Design: Some Reflections. *Journal of Applied Systems Analysis.* No. 17 (63–70).

Wilson, B. (1984): *Systems: Concepts, Methodologies, and Applications*. Chichester: Wiley.

Wood-Harper, A. T., L. Antill & D. Avison (1985): *Information Systems Definition: The Multiview Approach*. Oxford: Blackwell Scientific Publications.

Yourdon, E. (1989): *Modern Structured Analysis*. Englewood Cliffs, New Jersey: Prentice-Hall.