On the Role of Activity Diagrams in UML

B. Paech

Institut für Informatik, Technische Universität München
Arcisstr.21, D-80290 München
++49/89-28928186
++49/89-28928183
paech@informatik.tu-muenchen.de

Abstract
Activity Diagrams can be used to describe internal processing as well as action-object flow. Since they do not focus on events and object interaction, it is not clear, how to combine them with the typical object-oriented diagrams like class and statechart diagrams. In this paper we propose to use activity diagrams as a bridge between use case diagrams and class diagrams. This gives three benefits: a smooth transition from business processes to use cases, an abstract specification of complex object interactions and a succinct description of system functions affecting several objects. This use of activity diagrams is embedded in an overall software development process characterized by a focus on user tasks during analysis and incremental class diagram development.

Keywords
Analysis and Design, Use Cases, Activity Diagram

1 Introduction

UML defines a set of graphical diagrams providing multiple perspectives of the system under analysis or development. As recognized in the UML Summary, version 1.1 [UML1.1] , activity diagrams play a special role among these diagrams. They do not correspond to the typical object-oriented techniques from the predecessor methods Booch [Boo94] , OMT [RBP+91] , and OOSE [Jac92] . In particular, they incorporate concepts from data flow diagrams used in structured methods. Usually it is claimed that structured and object-oriented concepts do not fit well together, since object-orientation focusses on event flow and object interaction, while structured method focus on data and control flow between processes. Thus, it is not clear how and when to use activity diagrams in an object-oriented software development process.

In this paper we propose to use activity diagrams as a bridge between use case diagrams and class diagrams. First, activity diagrams are used to describe the flow between the system and the actors within the use cases. We call these descriptions *work processes*. They make the user tasks and the disvion of work between user and system explicit. Work processes can be viewed as a refinement of business process descriptions. Second, activity diagrams are attached to the system to describe the data effects of system functions without fixing the object interaction necessary to achieve the effects. We call these descriptions *function processes*. They are particularly helpful for a succinct description of functions affecting several objects. Typically, there is a wide variety of possible designs for such functions, since the control can be attributed to several entity or interface objects, as well as to a separate control object [Jac92] . Thus, it is often difficult to recover the effects of such functions from the class diagram.

Along with the development of work and function processes the class diagram can be developed incrementally. On the level of work processes a class diagram without operations and navigability indication is used. This preliminary class diagram serves as an illustration of the terminology making explicit the major dependencies between different entities. On the level of function processes navigability is added. This is necessary in order to determine which data is affected by a system function. Otherwise, it is e.g. not clear whether the update of a binary association affects one or two references. Finally, operations are added to the object model corresponding to the activities of the function processes.

The rationale behind this use of activity diagrams is a software development process focusing on user tasks in the early phases and a smooth transition from task oriented diagrams to class diagrams. There is evidence that class diagrams are not suitable for requirements engineering, since they are not intuitive for the users (cf. [Moy94]). While use cases are easy to understand for the users, they lack methodical guidance for their intergration with class diagram

development [WPJH98]. In our view, work and function processes are an adequate enhancement of use cases, since they are also intuitive for users, but more accurate, and since they give methodical guidance by separating work issues form data effect considerations.

The paper is structured as follows: in section 2 we introduce work and function processes by way of an example and discuss necessary extensions to activity diagrams. We show how to develop the class diagram in parallel with work and function processes. In section 3 we sketch the user-oriented software development process in which work and function processes should be embedded. Section 4 contains the conclusions. Related work is discussed along the way.

2 Work and Function Processes

In this section we introduce work and function processes by way of the ubiquituous library example. In particular, we treat the book return use case whose textual description is shown in Figure 1. Use cases contain a lot of details about the interaction of the users and the system. This makes it difficult to determine, whether all important information has been captured. Also, it is a very big step to the class diagram in which the behaviour described in the use cases has to be distributed between the classes. Thus, we propose to use two intermediary activity diagrams, namely work and function processes, to separate the information of use cases. Along with these intermediary levels the class diagram can be developed incrementally. In the example, we start with the textual use case which is then captured in work and function processes. Of course, one could imagine also the other way round, where work and function processes are developed first, and then enriched to the full textual use case. It depends on the project and especially the user community, which description should be used. For visionary use cases, the textual description might be easier to understand. If complex user task performance is to be captured, work and function processes might be more adequate, because they separate several concerns into different descriptions.

The course of events starts when a reader hands a book to the librarian to return it to the library. The librarian enters the book number. The system retrieves the title and author of the book, as well as the reader identity, for the librarian to acknowledge that the correct book is returned from the correct reader. In reaction to the acknowledgement of the librarian the system updates book and reader data and checks whether the book has been reserved. If so, an Email message is sent to the owner of the reservation. Finally, the success of the whole transaction is notified to the librarian.

Figure 1: Book Return Use Case

2.1 Work Process and Problem Domain Class Diagram

A work process is an activity diagram describing the division of work between user and the system as captured in the use case. Figure 1 shows the work process corresponding to the book return use case. As usual in activity diagrams, we use swimlanes to separate the activities of different actors. Control flow is shown by a solid arrow, object flow by a dashed arrow. As an extension of activity diagrams we allow - similar to sequence diagrams - messages as labels to the control flow arrows. It is not possible to use events, in particular the reception of a message - as typical for statechart diagrams -, instead of messages, since we label an arrow between two swimlanes *A* and *B* with the message sent from *A* to *B* and not with the message received by *A.*

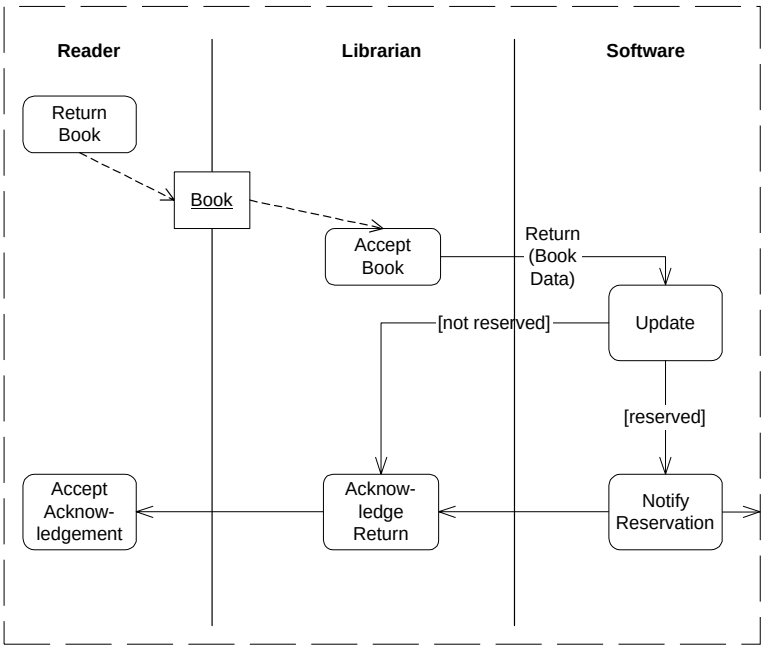

Figure 2: Work Process Book Return

The book return work process consists of the return book activity of the user from which the book flows to the librarian. The librarian accepts the book and commands the software system to execute the corresponding update. The system checks whether a reservation exists, and if so, notifies the next reader. The success of the transaction is signaled to the librarian which in turn acknowledges the book return to the reader.

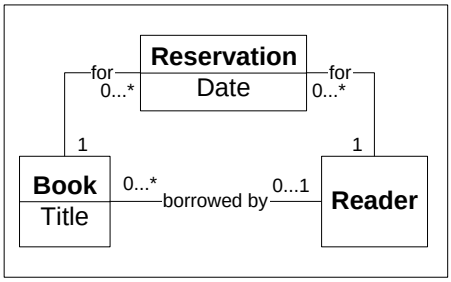

Figure 3: Problem Domain Class Diagram

Figure 3 shows the problem domain class diagram corresponding to the work process. Similar to OOSE we only use a preliminary class diagramm without operations and navigation indication. Only the major entities and their important relationships should be shown. For the book return use case, these are the book and the reader connected by the borrowing relationship, and the reservation connected to a unique book and reader.

Work processes and the problem domain class diagram do not show interaction details like the separate acknowledgement of the updates by the librarian or class details like operations. Only the major activities, decisions and entities are shown. At this level one can experiment with different possibilities of work division between the user and the system. The similarity of work process diagrams to business process descriptions, e.g. [Sch92], allows for an easy transition from strategic business processes to work place descriptions. This is in line with the adaption of the use case model to business processes in [JEJ94] . Based on a more detailed activity description, e.g. regarding the priority, frequency, degree of freedom, ressources and the like (cf. [BJ94]), the complete set of user activities can be analyzed to determine the adequacy of work design for human labour (e.g. [Uli94]).

2.2 Function Process and Navigation Class Diagram

A function process is an activity diagram describing the data effects captured in the use case. More specifically, they detail the software activities identified in the work processes. Figure 4 shows the function process for book return. Function processes do not contain swimlanes, since they describe only activities of one actor, namely the software system. Each activity describes one kind of data change. We do not associate the activities with the objects at this level, since the flow between the objects might be changed through the addition of control objects anyway. The behaviour of functions affecting several objects can be described more succinctly by concentrating on the activities. Again we allow labels for the control flow arrows. In this case they represent data dependencies between the activities, not necessarily messages. In the Book Return function process, Reader and Reservation are data flowing from the update book activitiy to the other two activities.

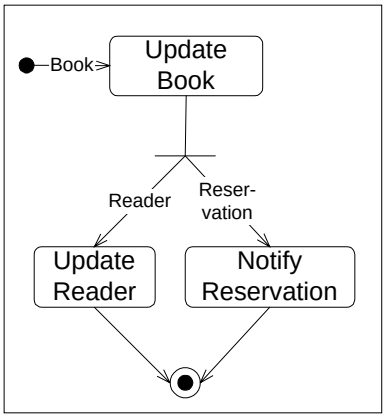

Figure 4: Function Process Book Return

To understand the function processes, one has to know the *navigation class diagram* which details the problem domain class diagram with navigation indication. Figure 5 shows the navigation class diagram for book return. In comparison with Figure 3, the borrowing relationship has been resolved into two references, while the relationships between reservation and book and reader, respectively, have been resolved into one-way-references from book to its reservations and from reservation to the reader who owns the reservation.

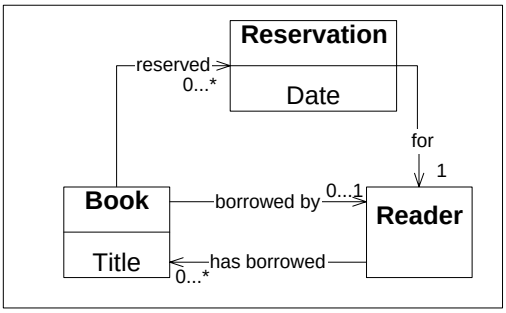

Figure 5: Navigation Diagram for Book Return

Because of the two-way-reference between book and reader, the book return function process details the update activity of the work process into book and reader update. In principle, book update, reader update and reservation notification could be executed in parallel. However, as shown in the navigation class diagramm of Figure 5, the reference to the borrowing reader and the reservations can only be gained from the book object. Therefore, update book should be executed first.

Figure 6 shows two possible collaboration diagrams describing the data effects of book return through object interaction. There are essentially two possibilities: either the reader object triggers the book and the reservation

object or there is another control object sequencing book update, reader update and notification. In general. there are much more possibilities.

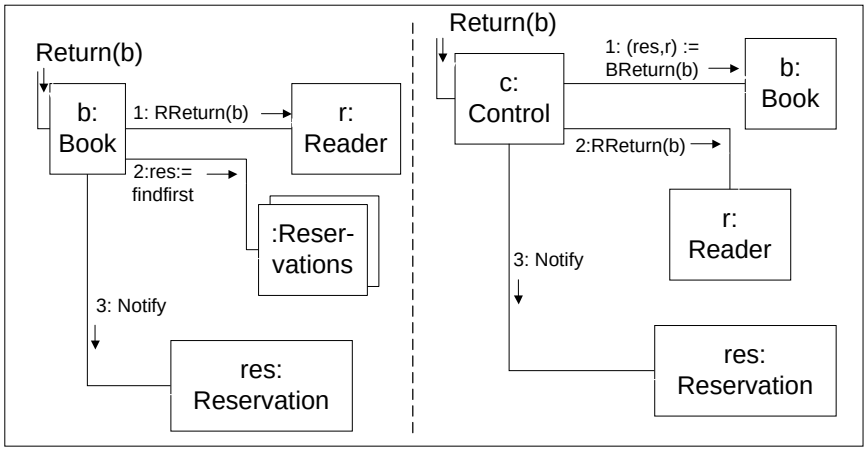

Figure 6: Two Possible Collaboration Diagrams for Book Return

This example demonstrates, that function processes are more abstract than collaboration diagrams. While function processes only show the major data effects and their sequencing, in collaboration diagrams the data effects are mixed with object interaction details. In our view, collaboration diagrams are more adequate for design than for requirements capture and analysis.

2.3 Analysis Class Diagram

Based on the function processes and the navigation class diagram, the analysis class diagram can be completed. In particular, control objects are added and operations associated with the classes. Each activity of the function processes leads to an operation at the class corresponding to the affected data. If no control object is added, these operations also contain the sequencing between the operations of the different classes. Otherwise, this is localized in the control object.

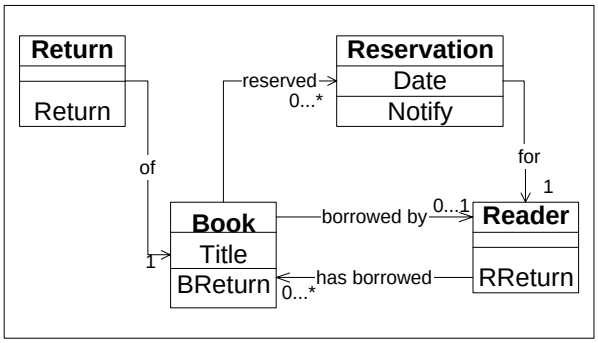

Figure 7: Analysis Class Diagram for Book Retun

Figure 7 shows the analysis class diagram for the book return use case. In the example, we have added the control object with the central return function. Book and reader have auxiliary functions for updating their references according to the return.

Use cases also contain information about interface objects which is not captured in the work and function processes and the corresponding class diagrams. In [Pae98a] we describe how to derive dialogue models from use cases which serve as input for the identification of interface classes and their operations. Here we only concentrate on the use of activity diagrams. The overall development process which work and functions processes are part of is described in the next section.

2.4 Evaluation of Activitiy Diagrams

Above we have shown by way of example the use of activity diagrams to model work and function processes. These two uses are quite different. Work processes use swimlanes and are therefore somewhat similar to sequence diagrams. Function processes describe behaviour of one actor. As discussed in [BHH+97] , it is not possible to give a common formal semantics to both uses. By allowing data labels in function processes we have given them yet another flavour - more similar to effect/correspondence diagrams of structured methods like SSADM [DCC92]. Thus, while we only needed slight extensions to the UML notation for activity diagrams, a formal semantics of work processes and function processes requires further effort. However, this effort will be worthwhile, since we will show in the next section that both play an important role in the early phases of a task- and object-oriented software development process.

3 A Task- and Object-oriented Software Development Process

UML only provides a notation, not a process. Of course, the development processes of the predecessor methods OOSE, OMT, and Booch can be adapted to the new notation. We are interested in a user-oriented process which allows for close interaction with the user in fixing the system functions supporting the user tasks and the user interface details. Class Diagrams - as proposed by most object-oriented methods for requirements capture -, are not suitable, since they do not support the notion of user task. As exemplified by the rich literature on work psychology (e.g. [Uli94], [Dia89]), task is the central notion to describe work places. This is the main reason for the success of use cases which bring into object-orientation the concept of user task. However, as discussed in the last section, in our view textual use cases mix different levels of task description. While they give the user and the developers a good intuition about the system support for user tasks (this is called *stakeholder requirements* in [SS97]) , they are not adequate as a system specification which has to make explicit the specification of the individual system functions. This detailed system specification is important for the contract between developers and customers, and project management issues like time and budget planning. Therefore, we propose in the following an adaption of the OOSE-process by work and function processes, where the function processes are part of the system specification. This process has not been applied in the development of an industrial software system, but the products of the process discussed in the following have proven very useful in the redocumentation of a legacy system.

Figure 8 depicts the major products of our task- and object-oriented software development process [Pae98b] . We only deal with business and requirements engineering, as well as the analysis stage. Similar to [JEJ94] , we use the same kind of models for business and requirements engineering. The *business use case model* gives an overview of the services of the company to the customer. *Business Processes* represented as activity diagrams detail the business use cases. At this stage, one can already start the development of *the problem domain class diagram.* The business engineering is completed by identifying the actors involved in the software system use (called *user roles*) and by fitting the scope of the software system into the overall *IT-strategy* of the company.

To identify the *stakeholder requirements*, on one hand the prospective user community is classified according to their competences regarding IT-usage. On the other hand, business processes are detailed into *work processes.* This means that the user tasks identified in the business processes are divided between the software system and the users. As discussed in section 2, the activities of the work processes should be described in sufficient detail for an evaluation of the work places for the user.

The *system specification* collects the work processes into the use case model. It also details the software activities of the work processes into function processes in combination with a navigation class diagram. Of course, also a textual system function description and non-functional requirements description as e.g. standardized in [IEEE93], has to be added. Based on the navigation class diagram, one can derive data views (cf. [Zie97]). These views are preliminary interfaces classes, but without operations. Similarly, from the function processes dialogue models (cf. [Den92]), can be devised. These dialogue models describe the possible user inputs to control system function execution depending on the data views presented to the user. To tailor the usage options to the specific needs of the users, *scenario techniques* as described in [Car95] should be used.

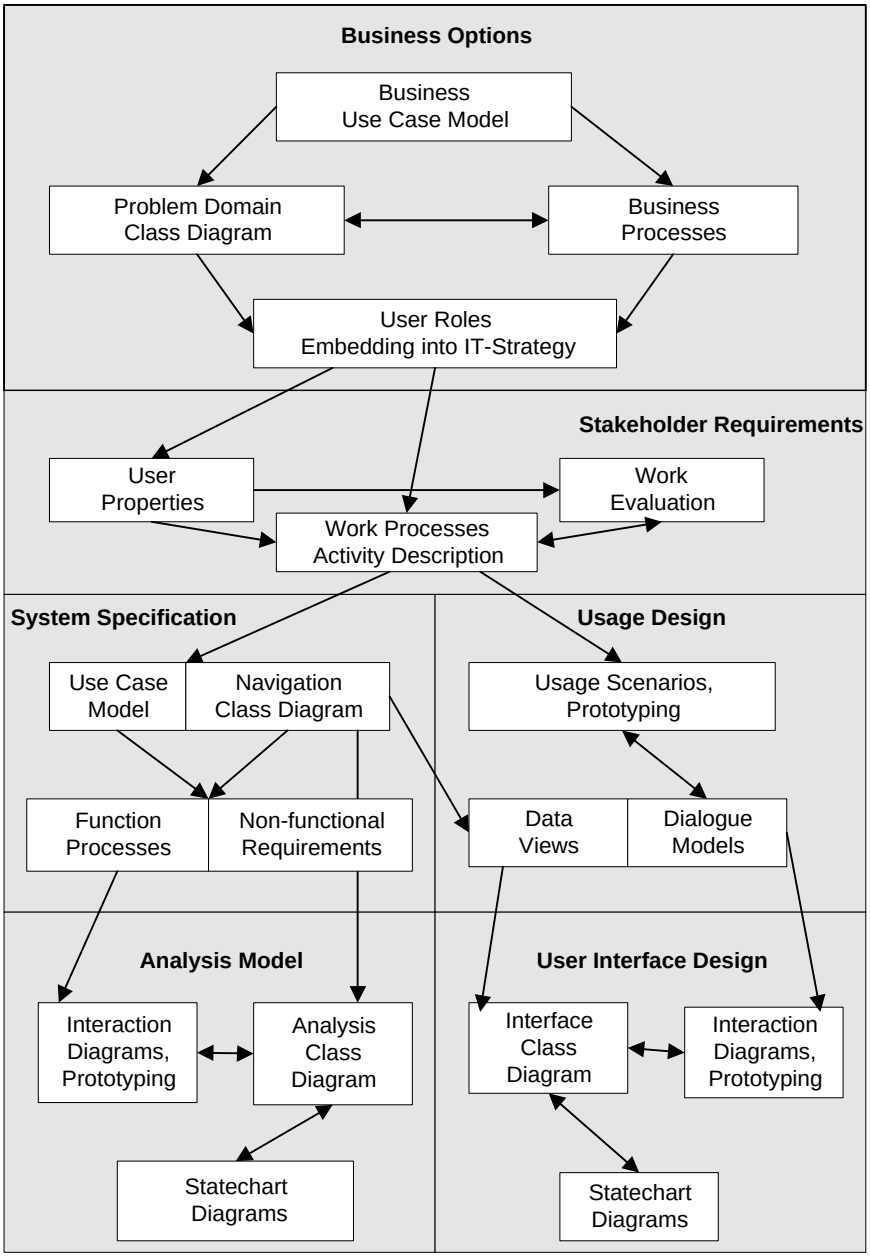

Figure 8: Task- and Object-Oriented Software Developement

For the *analysis model*, the navigation class diagram is complemented with operations and control objects as described in section 2. To support the addition of the control objects, the realization of the function processes through object interaction can be examined with the help of sequence and collaboration diagrams. Statecharts diagram are used to describe the integration of several function processes within the different classes.

Similarly, the *user interface design* consists of the interface class diagram which is a completion of the data views identified in the usage design by operations. Again, interaction diagrams arnd prototypes are useful for the examination of different realizations of the global dialogue models through object interaction. Statechart diagrams again integrate the different dialogues within the interface classes.

Based on the analysis and interface class diagrams, the OOSE design stage, as well as other object-oriented design stages can be added to the development process.

In the process, we have not included the textual use case description of [Jac92] . The information of the textual use cases is separated into different models, namely, the work processes and activity descriptions, function processes, usage scenarios and dialogue models. As discussed in section 2, one could also use only the textual use cases. The separation into different models gives a guidance on how to develop these use case text incrementally and on how to check whether all important information has been captured by the use case desriptions.

4 Conclusions

In the paper we have shown how to make use of UML activity diagrams in a user-oriented software development process aiming at an object-oriented analysis and interface model. In particular, we enhanced the notion of use cases by work processes and function processes which separate work description from data effect description. This allows for an incremental development of the class diagram, as well as for an explicit system specification. The latter is missing in many object-oriented development methods. Work processes are inspired by business process and task descriptions. Function processes are inspired by structured methods. Both diagrams serve as a bridge between the task- and business-oriented requirements capture and the object-oriented analysis and interface model.

5 Literature

[BHH+97] R. Breu, U. Hinkel, C. Hofmann, C. Klein, B. Paech, B. Rumpe and V. Thurner, Towards a Formalization of the Unified Modling Language, in ECOOP, LNCS 1241, pg. 344-366, Springer, 1997

[BJ94] A. Beck and Ch. Janssen, TASK - Technik der Aufgaben- und Benutzerangemessenen Software-Konstruktion, Technical Report, IAT, 1994

[Boo94] G. Booch, Object-Oriented Analysis and Design with Applications, Redwood City, 1994

[Car95] J.M: Carroll, Scenario-Based Design, John Wiley & Sons, 1995

[Dav95] A. Davis, Object-oriented Requirements to Object-oriented Design: An Easy Transition? Journal of Systems Software, 30, pg. 151-159, 1995

[Den92] E. Denert, Software-Engineering, Springer, 1992

[DCC92] E. Downs, P. Clare and I. Coe, Structured Systems Analysis and Design Method: Application and Context, Prentice-Hall, 1992

[Dia89] D. Diaper, Task Analysis for Human-Computer Interaction, Ellis Horwood Limited, 1989

[IEEE93] IEEE-STd. 830-1993, Recommended Practice for Software Requirements Specification

[JEJ94] I. Jacobson and M. Ericsson and A. Jacobson, The Object Advantage: Business Process Reengineering with Object Technology, Addison-Wesley, 1994

[Jac92] I. Jacobson, Object-oriented Software Engineering, Addison-Wesley, 1992

[Moy94] T. Moynihan, Objects versus Functions in User-Validation of Requirements: Which Paradigm Works Best?, in OOIS'94, pg. 54-73

[Pae98a] B. Paech, The Four levels of Use Case Description, REFSQ'98, to appear, 1998

[Pae98b] B. Paech, Aufgabenorientierte Softwareentwicklung, Habilitationsschrift, eingereicht an der TU München, April 1998

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, 1991

[Sch92] A. Scheer, Architecture of Integrated Information Systems: Foundations of Enterprise Modelling, Springer, 1992

[SS97] I. Sommerville and P. Sawyer, Requirements Engineering - A Good Practice Guide, Wiley & Sons, 1997

[Uli94] E. Ulich, Arbeitspsychologie, Schaeffer-Poeschel Verlag, 1994

[UML1.1] G. Booch, J. Rumbaugh and I. Jacobson, The Unified Modeling Language for Object-Oriented Development, Version 1.1, 1997

[WPJH98] K. Weidenhaupt, K. Pohl, M. Jarke and P. Haumer, Scenario Usage in System Development, A Report on Current Practice, in ICRE'98, IEEE, 1998

[Zie97] J. Ziegler, Viewnet - Konzeptionelle Gestaltung und Modellierung von Navigationsstrukturen, in Software Ergonomie'97, pg. 343-350