Published in: McGregor, R. (Hg.): Proceedings of the Third Australian Conference on Information Systems. Wollongong/Australien 1992, pp. 675-690

A TOOL-SUPPORTED METHODOLOGY FOR DESIGNING MULTI-PERSPECTIVE ENTERPRISE MODELS

ABSTRACT

Development and organisational implementation of corporate information systems are still afflicted with severe economic problems. They are partially due to the fact that advanced software engineering methodologies are rarely applied. Moreover requirements analysis often does not produce satisfactory results. While user participation is widely agreed to be a prerequisite of successful system design it is often hard to accomplish. System analysts and the various users have different views of an organisation which makes it difficult to communicate efficiently. Object-oriented enterprise modelling can contribute to improve this situation. It encourages conceptual design using semantically rich domain level concepts. This is not only a prerequisite for implementing highly integrated information systems. Enterprise models are also suited to establish a common universe of discourse for both system developers and users by providing illustrative representations of organisations which are based on a solid software architecture. The paper introduces a conceptual framework for the design of multi-perspective enterprise models which does not only incorporate advanced object-oriented methodologies but also takes into account epistemological, organisational and economic aspects. A computer based integrated design environment is presented that is based on this framework. It supports the interactive development of object-oriented conceptual models. For this purpose it provides graphical representations of static as well as dynamic aspects of organisations and allows for analysing the effectiveness of office procedures.

INTRODUCTION

Computer based corporate information systems are of crucial importance for most business firms. There is a great amount of experience with implementing and using information technology. However, a satisfactory integration of information systems is still more likely to be the exception than the rule.There is a wide range of reasons which contribute to this phenomenon. During software development there is often not enough emphasis on a thorough and detailed requirements analysis. The participants in requirements analysis - like designers, users and managers - do not share the same view of the problem. In other words: when they talk about the enterprise or features of information systems they do not necessarily use the same language. Development suffers from friction between the different phases - information gets lost or is misinterpreted. People responsible for design and implementation are not sufficiently qualified. “To most people, including a surprising number who program computers, software engineering is a mystery” (Macro/Buxton [1987, p. vii]). Software development often starts - more or less - from scratch, reuse of application domain concepts rarely happens. Thereby costs for development and maintenance are very high, the results however are often poor - from a technological as well as from an economic point of view. The software architecture does not sufficiently reflect needs for maintenance and adaptability. Integration of the components of an information system is not satisfactory. Information systems are not always implemented within an organisation such that they support the business needs effectively. While most executives are aware of the economic importance of information systems they are usually not capable of evaluating their quality. IS-managers on the other side often lack a substantial understanding of enterprise goals (Lederer/Mendelow [1987]).

These problems tend to get even more complex. Planning and implementing of information systems suffers from the burdens of old technology ("the horrors of the past", Meyer [1990], p. 83). At the same time new technology is emerging (like CASE, graphical user interfaces, object-oriented programming etc.). While it is difficult enough for an IS-manager to keep informed it is almost impossible to evaluate the different options in a satisfactory way. On the other side managers who are responsible for planning and organising the business face numerous new challenges. With markets becoming more and more international competition is increased. Thereby it is required to evaluate different options to reorganise the business - like outsourcing, value added partnerships, electronic data interchange, concurrent engineering etc.

Since management of information systems as well as the issues of organisational design are highly complex matters on their own, it is not surprising that people tend to have an isolated scope of the problem. It is almost necessary in order to deal with complexity. The problem itself however requires to integrate the different scopes somehow. There has been awareness for this challenge for long. It is commonly agreed that it is necessary to improve the chances for participation. Real participation requires a solid understanding of the problem. Since people tend to perceive and conceptualise reality in numerous different ways representation of the domain of interest is a core issue. Within software engineering conceptual modelling aims at "the development of descriptions of a world/enterprise/slice of reality which correspond directly and naturally to our own conceptualizations" (Levesque/Myloupolos [1984, p. 11]). Object-oriented methods promise to be better suited to accomplish this goal than traditional data oriented methods: it seems to be more natural to describe objects as abstractions of real world entities than to reconstruct reality by artificially differentiating data and operations. Furthermore: enterprise wide object-oriented models promise to foster integration as well as software reuse on a high level (see below). Methodologies for object-oriented analysis and design however are mainly focusing software engineering needs. Research in system analysis and design that originates from social sciences on the other side puts more emphasis individual perception and social interaction. It resulted in methodologies which focus on the process of participation (Mumford [1983]), on the comprehensive representation of individual concepts (Checkland [1981]) as well as in theories on organisational change (for an overview see Keen [1981]). They lack however a software engineering perspective.

The project "Computer Integrated Enterprise" that was started at the German National Research Center for Computer Science (Frank/Klein [1992]) in 1990 was intended to integrate these different scopes. Particularly the project goals were to

- develop a methodology as well as a set of tools to support the design and maintenance of enterprise models.
- thereby provide a representation of organisations that is illustrative for business people and takes into account the requirements of object-oriented analysis and design at the same time.
- demonstrate the benefits of object-oriented analysis and design in a real-world application by comprehensively modelling a specific enterprise.

ENTERPRISE MODELS: PROSPECTS AND REQUIREMENTS

While the notion of enterprise models becomes more and more popular - within the research community (see for instance Pröfrock et al. [1989], or ESPRIT [1991]) as well as in the area of commercial software development (IBM [1990]) there is no detailed consensus on what an enterprise model should look like. Enterprise models are supposed to provide a suitable foundation for integrating information systems. In order to develop requirements for the design of enterprise models we will first analyse the characteristics of integration.

Dimensions of Integration

Within the context of information systems the term integration is usually restricted to the different components of the system. The brief overview on the overall problem however indicates that other dimension of integration should be taken into account as well:

- integrating the different phases of the software life-cycle
- integrating the different roles and perspectives of those who analyse, design and use an information system
- integrating the information system with the organisation
- integrating the information system with those of business partners

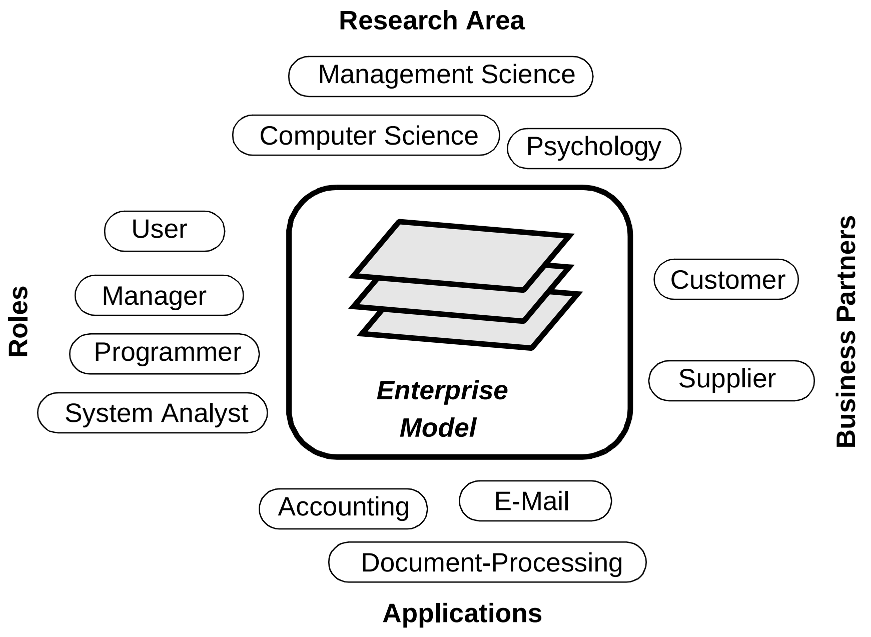


Fig. 1. Dimensions of Integration fostered by Enterprise Models

To find out what is required to accomplish integration and how different levels of integration can be distinguished we shall first look at the components of an information system. Integration implies communication. For components to be able to communicate there has to be a common semantic reference system. In other words: they need to have corresponding interpretations of the symbols they interchange as well as common unique names for these interpretations. Data types, functions of an operating system or relations within a database are examples for such reference systems. The more semantics is incorporated in the concepts that can be referred to the higher is the level of integration. The amount of semantics itself depends on the number of permitted interpretations. A data type like an integer can be interpreted in numerous different ways - depending on what real world entity it represents. A concept however that directly represents a real world entity reduces the set of possible interpretations. Is there any indication for the appropriate amount of semantics? It seems to be desirable to provide concepts that incorporate enough semantics not to bother any of the involved components with the need to reconstruct meaning for further processing. For instance: defining a concept “account” rather than only providing more general concepts that could be used to implement an account in a convenient way. If you then include a certain graphical representation of an account into a document the document processor should know the semantics of an account - which would improve the chances for powerful interpretations. Considering the need for flexibility and reusability however recommends to also provide more general concepts that allow for specialisation.

Common semantic reference systems are not only a prerequisite for technical integration. Integration of different human perceptions of reality also requires common reference systems. They are provided by terminology (not to speak of language in general), by culture (which Luhmann [1984, p. 224] defines as a reservoir of common themes that are possible subjects of communication), and of organisations as well, which is emphasized by Weick [1979, p. 3] who suggests that organising should be defined as “consensually validated grammar for reducing equivocality ...”. Different from formal systems a certain amount of ambiguity is not only tolerable but sometimes even helpful to cope with complexity.

Perspectives on the Enterprise

What are the implications of these thoughts for the design of enterprise models? First: for enterprise models to serve as promoters of integration they need to offer different levels of abstraction. There has to be a level that is appropriate for software development and integration of (software-) technical components. Furthermore it has to include representations that correspond with users´ conceptualisations of reality. Considering the numerous views/conceptualisations (see for instance ESPRIT [1991], Zachman [1987]) one can think of it is necessary to make a suitable selection. We decided on three main levels of abstraction:

- a *strategic view*
- an *operational/organisational view*
- an *information system view*

Designing and implementing a corporate information system requires to have a solid idea of how the operations of the firm are organized. Since an information system should be effective for a long time it is desirable to consider strategic options in time. The strategic view, which is not subject of this paper, is described by concepts like business goals, value chains (Porter [1985]), portfolio analysis, corporate culture etc. Within the organisational view we differentiate between three perspectives. The macro-view describes the main organisational units or functional areas of an enterprise, like marketing, accounting, controlling, personal etc. On a more detailed level concepts like roles, functions, objects, business rules, and scenes (for instance: sales negotiations) are described. Finally there is a dynamic level to represent office procedures in a way that is illustrative for managers. The informations system view is focused on what traditionally is called conceptual model (for a elaborated definition see Brodie [1984, p. 20]). Since the concepts that are used on the three levels are different it is crucial to translate or mediate between them, for instance by linking concepts that are somehow related.

Implications for an Environment to support the Design of Enterprise Models

An enterprise model that is to serve as a foundation for information systems requires a suitable software engineering approach. At the present time an object-oriented approach seems to be the best choice. The reasons that were most important for us:

- Objects allow for describing concepts on a higher semantic level than traditional data structures. Thereby they are better suited to directly correspond to real world entities.

- Encapsulation promotes a modular architecture and thereby flexibility of information systems.
- Inheritance allows for conveniently modelling generalisation and specialisation.

Considering the complexity of the overall design process it is important to provide a tool that supports a systematic approach. Such a tool should enforce a certain methodology for object-oriented analysis and design. It should prevent the model from becoming inconsistent by checking for ambiguity and contradictions. Participation requires a substantial understanding of how the system will look like. Therefore the tool should allow for fast prototyping. On the strategic as well as on the organisational level there may be concepts which cannot be formalised although they can be comprehensively described. In order to link them to related concepts it is desirable that the tool includes some kind of hypertext-features.

It is often argued that an information system should be adapted to the organisation, not the other way around. While such a request seems reasonable at first sight (specially when you consider how restrictive today´s software sometimes is) it is not completely convincing. This is for two reasons. First: a business firm´s actual organisation does not have to be efficient. Adapting an information system to it means to put effort in reconstructing inefficient structures and procedures. Second: in order to exploit the potential of information systems it can be suitable to rearrange an organisation that had been efficient on a lower level of automation. Like Schank [1985, p. 23] states for a wider context: “The first users of cars and computers had to struggle to make these completely new machines operate within the limits of the systems that were designed for an earlier world. ... Computers are severely limited by the world views and ideas that have preceded them.” Similarly Savage [1990, p. xii]: “Could it be that we are putting fifth generation technology in second generation organizations?” Taking these thoughts into account recommends mutual adaptation of organisation and information technology. To reduce the complexity of this task it is desirable that a tool allows for simulating organisational alternatives.

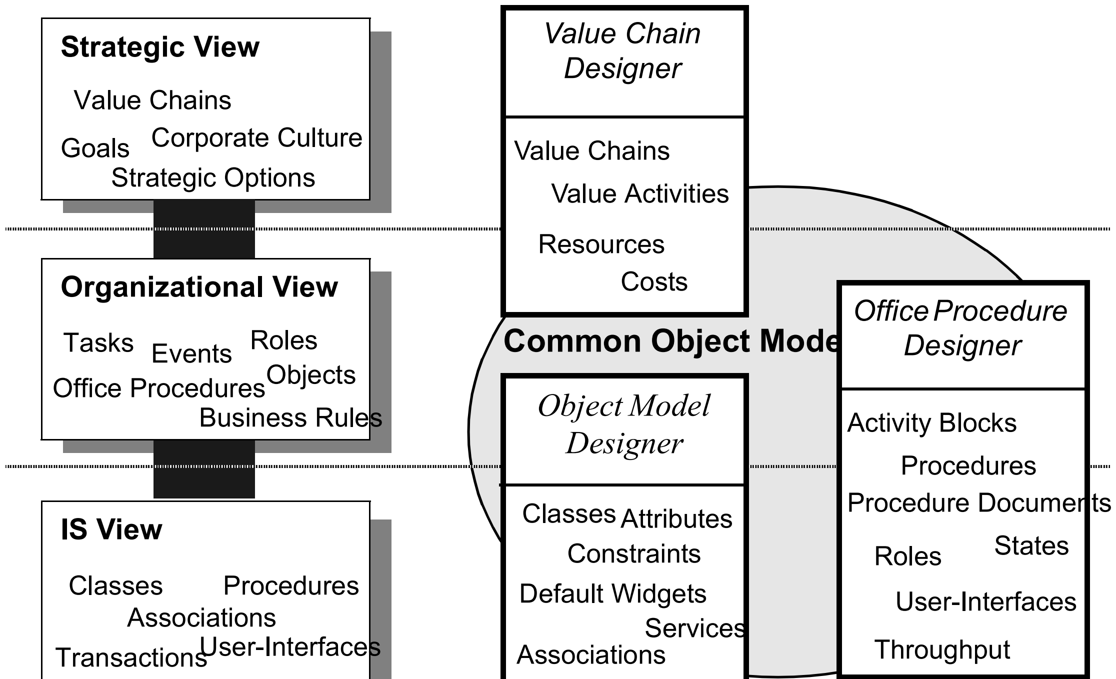


Figure 2. Tools of the design environment and their relation to different views of the enterprise

The environment we developed is a first attempt to fulfil the requirements listed above. Currently it consists of three tools which are enhanced by a hypertext-system. Each of the tools covers at least one of the three main levels of abstraction (see fig. 2). The *Value Chain Designer* (which is not subject of this paper) allows to design and analyse strategic options by applying Porter´s concept of value chains. Value activities, the core elements of value chains, are in part described by the resources and business processes they use - which are described on the organisational level. The primary scope of the *Object Model Designer* is the IS-view, while the *Office Procedure Designer* covers both dynamic aspects of the IS-view and of the organisational view. All the tools have been written in Smalltalk-80 within the Objectworks® environment.

OBJECT-ORIENTED REPRESENTATION OF ORGANISATIONS

During the last years a number of object-oriented analysis and design methodologies have been developed (Coad/Yourdon [1990, 1991], Booch [1990], Rumbaugh et al.[1991]). Our approach is inspired by Booch and Rumbaugh et al. They suggest three design levels for an object-oriented conceptual model: a static object model, a dynamic model, and a functional model. We regard an object model as the core of an enterprise model. While an object model can be sufficient to capture all the semantics you need for implementation it is definitely not sufficient to cover all important aspects of analysis and design. It hardly allows for comprehensively expressing temporal and functional semantics of an information system. Usually state transition diagrams (dynamic model) and data flow or message flow diagrams (functional model) are recommended to fill this gap. However, for our purpose these techniques have two shortcomings. They do not provide a representation that fits the average user´s perception of a business procedure. Since state transition diagrams describe the behaviour of objects of a certain class they can hardly be used to support the design of procedures from preexisting components. The Office Procedure Designer is intended to provide a more illustrative representation. It allows to define temporal as well as functional semantics. Furthermore our experience suggest that procedures are a preferred way for users to describe their view of the domain they work in. Thereby the description of a procedure can also serve as a heuristics to identify required objects/classes.

Although generic enterprise models that fulfil the requirements of a wide range of firms are an attractive research vision it is not possible to develop such models from scratch. You have to start with one enterprise of a particular domain. Applying the same approach to a set of similar enterprises allows for comparatively analysing invariance and differences. Then there is a chance to condense the specific model to a configurable generic model of a certain scope. The domain we started with is car insurance within an insurance company. The examples given below are taken from this domain. The following sections outline how to design object models and office procedures within the environment we have developed (for a more detailed description see Frank [1992]).

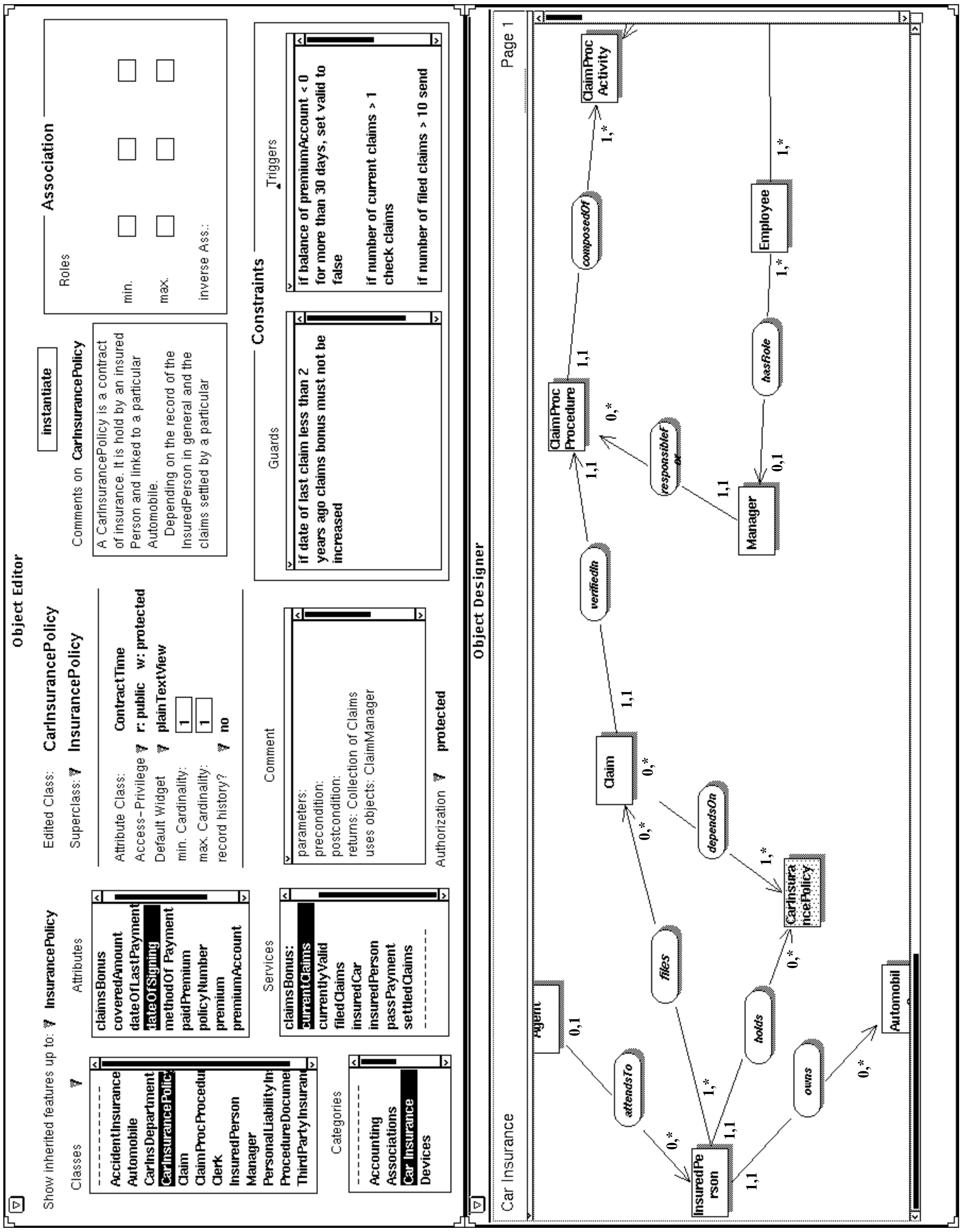
Conceptualisation of an Object Model

An object model consists of classes and relationships between them. While it is often argued that objects offer a natural way of describing reality it cannot be neglected that the notion of an object within a conceptual model has to be oriented towards a certain formal structure - no matter how people prefer to describe entities they perceive. An analysis/design methodology should provide analysts and users with a suitable and comprehensive concept of an object and guide the mapping of real world domains to object models. An object/class is modelled by describing attributes and services. Additionally we use the category constraints.

An attribute is regarded as an object that is encapsulated within the object. Among others it is described by *class*, *cardinality* and *default widget*. Specifying an attribute´s class is a prerequisite for typing. Cardinality has to be defined in min., max.-notation. For instance: a customer´s telephone number may have cardinality 0,*. In order to allow for generating prototypical user-interfaces it is possible to assign a default-widget to each attribute. One can also define a label that is to be presented with the widget. Additionally the size of the widget can be specified. This approach is a first attempt to deal with the complexity of user interaction. It cannot be completely satisfactory: the way a value of a certain class is presented to the user often is not unique but varies with the context of interaction. For instance: you can display a name using a scrollable text view, a listbox etc.

Services are characterized by their interface, where each attribute is defined by its class, and a natural language description of the function they fulfil. Furthermore a precondition and a postcondition can be specified. If the service returns an object, this object´s class can be specified. While attribute and service descriptions already include constraints (like attribute-classes, pre- and postconditions) there may be other object-constraints that cannot be assigned to just one attribute or service. This is the case for integrity rules which interrelate different attributes or services. We differentiate between two types of constraints: *guards* and *triggers*. A guard is a constraint that prevents the object from merging into a certain state. For instance: the resale-price assigned to a product should never be less than the purchase-price. A trigger on the other hand prevents an information system from becoming inconsistent by not reacting if some condition is fulfilled. For instance: If a customer who holds a car insurance policy has been driving without an accident for more than three years and has not assigned the highest claims bonus yet, his claim bonus has to be increased.

Objects within an information system are interrelated in various ways: objects may use services from other objects, they may be composed of other objects, their existence may depend on other objects etc. Taking such associations/relationships into account is crucial for maintaining the integrity of an IS. Therefore they are commonly regarded as an essential part of an object model. From a software engineering point of view it is desirable to limit the number of association types that are used. But in order to design illustrative as well as semantically rich domain level model we prefer associations which may include domain specific semantics and which are labelled with names that are known in the application domain. Having a wider range of different types of associations allows to define views on aspects of the object model. If somebody is interested in an organisational schema one could filter all classes which are associated via "is subordinated" or "is superior". A relationship may have features that cannot solely be assigned to any of the connected objects. For instance: information on the relationship *attendsTo* between an insurance agent and an insured person like "when was the relationship established?" or "where was it established?". For this reason we adopt the approach Rumbaugh et al. suggest: associations may be modelled as classes. The permissible cardinality range of an association has to be specified in min, max-notation. Each of the involved classes has to be assigned a tuple with the minimum number of instances that have to be part of the association and the maximum number that is permitted. One association class is thought to provide a substitute for multiple inheritance that even offers some advantages over the original. An object can import another objects´ features by establishing a "has role"-association (which is sometimes referred to as "dynamic" or "object-level" inheritance). The roles that are assigned to a class can be ordered to resolve possible naming conflicts. If you want to describe an employee who is a manager as well as a salesperson you do not define a class "managing salesperson" that inherits from manager and salesperson. Instead employee is assigned the roles manager and salesperson in a certain order.


Object Editor

Show inherited features up to: InsurancePolicy

Edited Class: CarInsurancePolicy
Superclass: InsurancePolicy

Classes
- - - - - - - - - -
AccidentInsurance
Automobile
CarInsDepartment
CarInsurancePolicy
Claim
ClaimProcProcedure
Clerk
InsuredPerson
Manager
PersonalLiabilityIn
ProcedureDocume
ThirdPartyInsuran

Categories
- - - - - - - - - -
Accounting
Associations
Car Insurance
Devices

Attributes
claimsBonus
coveredAmount
dateOfLastPayment
dateOfSigning
methodOf Payment
paidPremium
policyNumber
premium
premiumAccount

Services
claimsBonus:
currentClaims
currentlyValid
filedClaims
insuredCar
insuredPerson
passPayment
settledClaims
- - - - - - - - - -

Attribute Class: ContractTime
Access-Privilege: r: public w: protected
Default Widget: plainTextView
min. Cardinality: 1
max. Cardinality: 1
record history?: no

Comment
parameters:
precondition:
postcondition:
returns: Collection of Claims
uses objects: ClaimManager

Authorization: protected

instantiate

Comments on CarInsurancePolicy
A CarInsurancePolicy is a contract of insurance. It is hold by an insured Person and linked to a particular Automobile.
Depending on the record of the InsuredPerson in general and the claims settled by a particular

Association
Roles
min.
max.
inverse Ass.:

Constraints
Guards
if date of last claim less than 2 years ago claims bonus must not be increased

Triggers
if balance of premiumAccount < 0 for more than 30 days, set valid to false

if number of current claims > 1 check claims

if number of filed claims > 10 send

Object Designer
Car Insurance
Page 1

Figure 3. User-interface of the Object Model Designer

The Office Model Designer encourages the description of a real world domain using the conceptualisation introduced above. It allows for a graphical representation of an object model (see screenshot in fig. 3). Furthermore it provides dictionaries of already defined classes and checks for name conflicts. In order to facilitate searching for already defined classes as well as to support a systematic approach to find new classes, the classes are grouped into categories. The definition of categories should be oriented towards domain level concepts. Some of the categories we have chosen: accounting, car insurance, marketing, people, documents, devices, associations. A class may be assigned to more than one category.

Modelling Office Procedures

The object model describes the available classes, their services and - to a certain extent - what object states are permissible. It does not explain which objects are needed and how they are used in order to fulfil certain tasks. The concept of an office procedure is thought to provide a framework for describing these aspects.

We regard a procedure as an ordered graph of activity blocks (which I will refer to as activity as well), which can be represented as a semantically enriched Petri net. Each activity block (for a similar conceptualisation compare Lochovsky et al. [1988]) is an object associated with a certain role of an employee who is responsible for this particular task. An activity block can be modelled as a procedure itself. The information that is processed within a procedure is collected in an object of class "ProcedureDocument". In the case of concurrent processing special constraints have to be fulfilled (see below). Each activity block requires a certain state of the document as a precondition. Processing the document within an activity results in one or more new states of the document. Unlike a physical document it can be worked on at different locations at the same time - provided there are constraints which prevent inconsistent states.

A procedure's semantics can be divided into the following categories:

General constraints. For instance: A procedure must not contain deadlocks. There must not be endless loops. There should be no task that cannot be reached by any chance.

Constraints on activities. For instance: An activity requires a certain state of a certain document type. It must produce one of a set of possible document states.

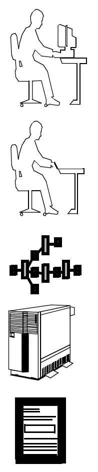
Constraints on documents. For instance: The variable parts of the document may be filled only with objects of a certain class. A part of the document that is processed within one activity may not be processed within another activity that works on the document concurrently.

Dispatching. For instance: After an activity block´s postcondition is fulfilled its successor has to be triggered, after an activity has been started, an employee who can take over the associated role has to be informed. It may be important to first check an employee's queue of activities before dispatching a new activity to him. Dispatching has to be done according to organisational rules, like: only one employee may be responsible for the whole procedure or for a collection of activities.

Exceptions. For instance: Within an activity block an inconsistent document state is detected that had been caused in a preceding activity. An employee becomes sick before completing the activity.

It is a crucial question for the design of a dynamic model to decide where to locate this knowledge. While general constraints should be checked already during the design process, all the other control knowledge can only be applied when the procedure is active. Each procedure is supervised by a procedure manager, which is an object that coordinates procedures of a certain domain. Whenever an event occurs that should trigger a procedure the procedure manager is notified. It then looks up its description of the particular type of procedure and instantiates the first activity block as well as the procedure document. Each activity block is responsible for transforming the document´s state to one of the states that are defined as postconditions. The procedure manager and the procedure document serve as “glue” to link the activity blocks. If an activity has terminated with one of its postconditional document states it notifies the procedure manager. The procedure manager looks up its list of available (human) operators and their queues of work to be done. Depending on its dispatch knowledge it will then instantiate an appropriate activity object and move it into the queue of the selected clerk.

The Office Procedure Designer is a tool to instruct analysis and design of office procedures according to the outlined architectural framework. For this purpose it provides the analyst/ designer with an interactive template for systematically describing a procedure´s tasks. It also includes a graphical editor that allows to model office procedures in an illustrative way using a set of graphical icons (see fig. 4). The icons represent either document states or tasks:



an activity that requires user-interaction

an activity that is not computer supported at all

an activity that itself is modelled as a procedure

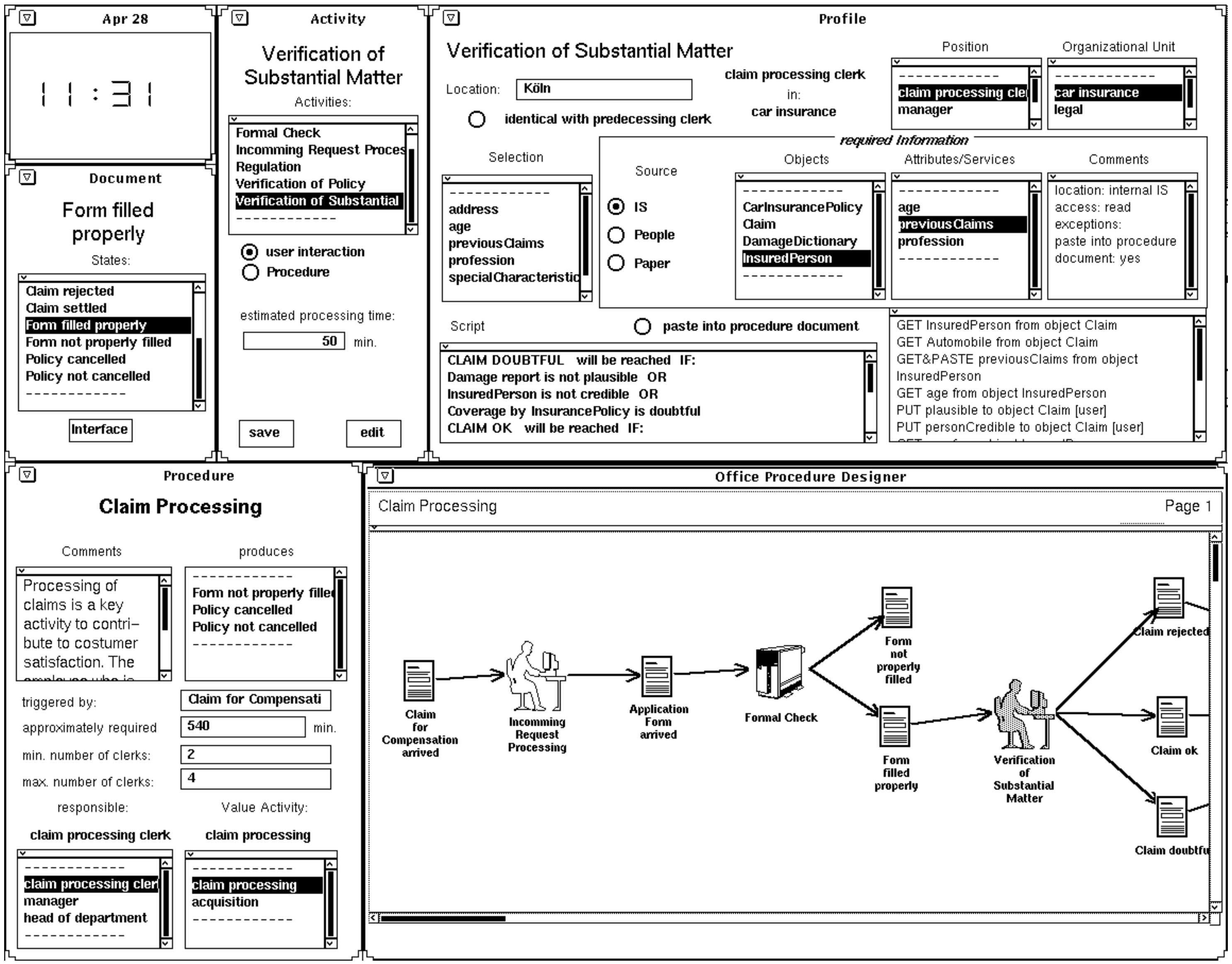
an activity that does not require user interaction

a procedure document´s state

Figure 4. Icons used for the graphical representation of office procedures

The first step of describing an office procedure as a net of activity blocks implicitly includes the definition of temporal semantics. Activity blocks can be ordered sequentially or concurrently which implies a notion of before, after and simultaneous. This allows the tool to perform certain consistency checks. For instance: detecting deadlocks, or an activity block that produces a document state that had already been produced before (the last example is only a strong indicator of inconsistent design).

Within the next step the activities are characterized by the structured, semi-formalized description that is encouraged by the interactive-template. Thereby three main aspects are differentiated: *organisational, informational*, and *control*. Organisational aspects are expressed by assigning a responsible employee (represented by an appropriate role, like “Manager”) and a department both to the whole procedure and to each activity block. Furthermore it is possible to define organisational constraints on the assignment of employees to activity blocks (like each activity has to be taken care of by only one person, or an activity block has to be supervised by the same person who supervised the preceding activity). Each activity block should also be assigned an estimated processing time. Gathering the information that is needed within an activity is crucial for capturing the essence of an activity. It is structured by offering three categories of information sources: *information system, people*, and *paper based documents*.


Apr 28

11:31

Document

Form filled properly

States:

Claim rejected
Claim settled
Form filled properly
Form not properly filled
Policy cancelled
Policy not cancelled

Activity

Verification of Substantial Matter

Activities:

Formal Check
Incomming Request Proces
Regulation
Verification of Policy
Verification of Substantial

[✓ user interaction] Procedure

estimated processing time: 50 min.

Profile

Verification of Substantial Matter

Location: Köln

identical with predecessing clerk

claim processing clerk in: car insurance

Position: claim processing cle, manager

Organizational Unit: car insurance, legal

required Information

Selection: address, age, previousClaims, profession, specialCharacteristic

Source: [✓ IS] People Paper

Objects: CarInsurancePolicy, Claim, DamageDictionary, InsuredPerson

Attributes/Services: age, previousClaims, profession

Comments:
location: internal IS
access: read
exceptions:
paste into procedure
document: yes

Script

paste into procedure document

```
CLAIM DOUBTFUL   will be reached  IF:
Damage report is not plausible  OR
InsuredPerson is not credible  OR
Coverage by InsurancePolicy is doubtful
CLAIM OK   will be reached  IF:
```

```
GET InsuredPerson from object Claim
GET Automobile from object Claim
GET&PASTE previousClaims from object
InsuredPerson
GET age from object InsuredPerson
PUT plausible to object Claim [user]
PUT personCredible to object Claim [user]
```

Procedure

Claim Processing

Comments:
Processing of claims is a key activity to contri–bute to costumer satisfaction. The

produces: Form not properly fille, Policy cancelled, Policy not cancelled

triggered by:	Claim for Compensati
approximately required	540 min.
min. number of clerks:	2
max. number of clerks:	4
responsible:	claim processing clerk
Value Activity:	claim processing

responsible list: claim processing cler, manager, head of department

Value Activity list: claim processing, acquisition

Office Procedure Designer

Claim Processing — Page 1

Figure 5. User-interface of the Office Procedure Designer

In order to instruct the description of the control flow within an activity block a template is presented that is generated depending on the document states that may result from the activity. It encourages a declarative description, which may be more or less formal. To support system analyst and domain expert in filling the template a report that includes a description of all the required information is presented in another text view (see fig. 5).

SIMULATION

After having preliminarily completed requirements analysis and design the available descriptions can be used to analyse the effectiveness of the procedure´s organisation. For this purpose a communication diagram can be generated. It shows the different roles participating in the procedure as well as the media they use to communicate. For further evaluation this diagram has to be interpreted by a domain expert. A more substantial indicator for the need to reorganize the procedure is a report of detected media frictions (like they occur when paper-based information has to be transferred to the IS). Other indicators for further evaluation are the total time the involved employees have to work on the procedure as well as the costs that can be calculated from the different costs that have been assigned to the different information sources.

Another question is more interesting but also more complicated to analyse since its scope is not restricted to the described type of procedure: what is the optimum number of employees needed to guarantee a satisfactory throughput? Or in other words: how can organisational slack be reduced to an optimum? For this kind of analysis the conceptual level is not sufficient. Instead it is the case for simulation. The current version of the Office Procedure Designer provides only limited simulation capabilities. Bottlenecks only occur in case more than one person works on a procedure (assumed that totally automated activities do not take considerable time). For this case it is possible to assign a number of people to each activity block that requires user interaction. Simulation then reveals bottlenecks and total throughput-numbers for different constellations. This however will only be sufficient in rare cases. Employees occupied within one procedure may also have to fulfil other tasks. It has also to be taken into account that employees have vacation days, that they may become sick (may be depending on the work load they face), that effectiveness of human work depends on a variety of aspects. Furthermore quality of work cannot be neglected, its relation to other variables however is hard to find out. Last but not least it does not make much sense to optimise the organisation of a single type of office procedure. Since procedures may be interrelated you need to widen the scope (Porter´s value chain concept is one approach to get an enterprise wide view). Optimising the organisation of work has been a dream for long. We do not think that enterprise modelling along with simulating organisational alternatives will make this dream come true. It can help however to reduce complexity by providing an illustrative representation of important aspects and by detecting certain types of organisational misconception.

CONCLUDING REMARKS

Our experience with modelling an office domain within an insurance company indicates that the proposed representations offer illustrative abstractions of an enterprise. This is specially the case for the representation of office procedures. The graphical notation was intuitively understood by both system analysts and domain experts. Thereby it is a valuable medium for starting knowledge acquisition or object modelling respectively. Users seem to prefer procedures as guidance in conceptualising the domain they work in. Therefore asking for a detailed description of office procedures does not only serve the purpose of adding dynamic or temporal semantics to the model it also provides a heuristics to shape the static object model.

There is still a lot of research to be done. In order to refine the domain model we have built so far it is necessary to apply the approach to other domains, preferable car insurance departments within other insurance companies. Our work has been primarily concentrated on object models and office procedures. Other levels of abstraction proposed in the conceptual framework (within the organisational and the strategic view) recommend a less formal representation. Knowledge that is described in textbook-style could be added using the already available hypertext features. Although office procedures are an illustrative metaphor it is not sufficient to describe all kinds of work in the office. Certain tasks (like analysing the records of customers) can be regarded as short procedures (consisting only of one activity block). Ill-structured cooperative work however requires other concepts as well as another graphical representation. It would be interesting to complement the Office Procedure Designer by a tool that allows for illustratively modelling CSCW-applications (for an example on the instance level see Ellis [1987]).

References

Booch, G.[1990], *Object-oriented design with applications*. Benjamin/Cummings, Redwood City

Brodie, M.L. [1984], “On the Development of Data Models.”, in: *On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases and Programming*. Ed. by Brodie, M.L.; Mylopoulos, J.; Schmidt, J., Springer, Berlin, Heidelberg etc., pp. 19-47

Checkland, P. [1981], *Systems Thinking, Systems Practice*. Wiley&Sons, Chichester

Coad, P. and Yourdon, E. [1990], *Object-oriented analysis*. Prentice Hall, Englewood-Cliffs

Coad, P. and Yourdon, E [1991], *Object-Oriented Design*. Prentice Hall, Englewood-Cliffs

Ellis, C.A. [1987], “NICK: Intelligent Computer Supported Cooperative Work”, in: *Proceedings of the IFIP WG 8.4 Workshop on Office Knowledge: Representation, Management and Utilization*. Ed. by Lochovsky, F., Toronto, pp. 95-102

ESPRIT Consortium AMICE [1991], *CIM-OSA AD 1.0 Architecture Description*. Brussels

Frank, U. [1992], “Designing Procedures within an Object-Oriented Enterprise Model”, in: *Dynamic Modelling of Information Systems*. Ed. by Sol, H.G.; Van Hee, K.M., North-Holland, Amsterdam, New York (to appear soon)

Frank, U. and Klein, S. [1992], *Unternehmensmodelle als Basis und Bestandteil integrierter betrieblicher Informationssysteme*. GMD research paper, No. 629, Sankt Augustin

IBM [1990], *IBM Enterprise Business Process Reference Model.*

Keen, P.G.W.: [1981], “Information Systems and Organizational Change”, *Communications of the ACM*, Vol. 24, No. 1, pp. 328-337

Lederer, A.L. and Mendelow, A.L., “Information Resource Planning: Overcomming Difficulties in Identifying Top Management’s Objectives”, *MIS Quarterly*, No. 11, Vol. 3, Sept., pp. 388-399

Levesque, H.J. and Mylopoulos, J. [1984], “An Overview of Knowledge Representation”, in: *On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases and Programming*. Ed. by Brodie, M.L.; Mylopoulos, J.; Schmidt, J., Springer, Berlin, Heidelberg etc., pp. 3-17

Lochovsky, F.H et al. [1988], “OTM: Specifying office tasks”, in: *Conference on Office Information Systems*. Ed by Allen, R.B., Palo Alto, pp. 46-54

Luhmann, N. [1984], *Soziale Systeme. Grundriß einer allgemeinen Theorie*. Suhrkamp, Frankfurt/M.

Macro, A. and Buxton, J. [1987], *The Craft of Software Engineering*. Addison-Wesley, Reading, Mass.

Meyer, B. [1990], “Lessons from the Design of the Eiffel Libraries”, *Communications of the ACM*, Vol. 33, No. 9, pp. 68-89

Mumford E. [1983], *Designing Participatively. Manchester Business School*, Manchester

Porter, M.E. [1985], *Competitive Advantage*. MacMillan, London, New York

Pröfrock, A.-K. et al. [1989], “ITHACA: An Integrated Toolkit for Highly Advanced Computer Applications”, in: *Object Oriented Development*. Ed. by Tsichritzis, D., Genf, pp. 321-344

Rumbaugh et.al. [1991], *Object-oriented modelling and design*. Prentice Hall, Englewood-Cliffs

Savage, C.M. [1990], *Fifth generation management - integrating enterprises through human networking*. Digital Press

Schank, R.C., *The Cognitive Computer. On Language, Learning and Artificial Intelligence*. Addison-Wesley, Reading/Mass.

Weick, K.E. [1979], *The Social Psychology of Organizations*. Addison-Wesley, Reading, Mass.

Zachman, J.A. [1987], “A framework for information systems architecture”, *IBM Systems Journal*, Vol. 26, No. 3, pp. 277-293

References

Booch, G.[1990], *Object-oriented design with applications*. Benjamin/Cummings, Redwood City

Brodie, M.L. [1984], “On the Development of Data Models.”, in: *On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases and Programming*. Ed. by Brodie, M.L.; Mylopoulos, J.; Schmidt, J., Springer, Berlin, Heidelberg etc., pp. 19-47

Checkland, P. [1981], *Systems Thinking, Systems Practice*. Wiley&Sons, Chichester

Coad, P. and Yourdon, E. [1990]**,** *Object-oriented analysis*. Prentice Hall, Englewood-Cliffs

Coad, P. and Yourdon, E. [1991], *Object-Oriented Design*. Prentice Hall, Englewood-Cliffs

Ellis, C.A. [1987], “NICK: Intelligent Computer Supported Cooperative Work”, in: *Proceedings of the IFIP WG 8.4 Workshop on Office Knowledge: Representation, Management and Utilization*. Ed. by Lochovsky, F., Toronto, pp. 95-102

ESPRIT Consortium AMICE [1991], *CIM-OSA AD 1.0 Architecture Description*. Brussels

Frank, U. [1992], “Designing Procedures within an Object-Oriented Enterprise Model”, in: *Dynamic Modelling of Information Systems*. Ed. by Sol, H.G.; Van Hee, K.M., North-Holland, Amsterdam, New York (to appear soon)

Frank, U. and Klein, S. [1992], *Unternehmensmodelle als Basis und Bestandteil integrierter betrieblicher Informationssysteme*. GMD research paper, No. 629, Sankt Augustin

IBM [1990], *IBM Enterprise Business Process Reference Model.*

Keen, P.G.W.: [1981], “Information Systems and Organizational Change”, *Communications of the ACM*, Vol. 24, No. 1, pp. 328-337

Lederer, A.L. and Mendelow, A.L., “Information Resource Planning: Overcomming Difficulties in Identifying Top Management’s Objectives”, *MIS Quarterly*, No. 11, Vol. 3, Sept., pp. 388-399

Levesque, H.J. and Mylopoulos, J. [1984], “An Overview of Knowledge Representation”, in: *On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases and Programming*. Ed. by Brodie, M.L.; Mylopoulos, J.; Schmidt, J., Springer, Berlin, Heidelberg etc., pp. 3-17

Lochovsky, F.H et al. [1988], “OTM: Specifying office tasks”, in: *Conference on Office Information System*s. Ed by Allen, R.B., Palo Alto, pp. 46-54

Luhmann, N. [1984], *Soziale Systeme. Grundriß einer allgemeinen Theorie*. Suhrkamp, Frankfurt/M.

Macro, A. and Buxton, J. [1987], *The Craft of Software Engineering*. Addison-Wesley, Reading, Mass.

Meyer, B. [1990], “Lessons from the Design of the Eiffel Libraries”, *Communications of the*

ACM, Vol. 33, No. 9, pp. 68-89

Mumford E. [1983], *Designing Participatively. Manchester Business School*, Manchester

Porter, M.E. [1985], *Competitive Advantage*. MacMillan, London, New York

Pröfrock, A.-K. et al. [1989], “ITHACA: An Integrated Toolkit for Highly Advanced Computer Applications”, in: *Object Oriented Development*. Ed. by Tsichritzis, D., Genf, pp. 321-344

Rumbaugh et.al. [1991], *Object-oriented modelling and design*. Prentice Hall, Englewood-Cliffs

Savage, C.M. [1990], *Fifth generation management - integrating enterprises through human networking*. Digital Press

Schank, R.C., *The Cognitive Computer. On Language, Learning and Artificial Intelligence*. Addison-Wesley, Reading/Mass.

Weick, K.E. [1979], *The Social Psychology of Organizations*. Addison-Wesley, Reading, Mass.

Zachman, J.A. [1987], “A framework for information systems architecture”, *IBM Systems Journa*l, Vol. 26, No. 3, pp. 277-293