

Organization Design as a Groupware-supported Team Process

**GroupOrga — Participative and Distributed Organization Design for
Office Information and Workflow Management Systems**

Additional and Technical Documentation

by

Dipl.-Wirt. Inform. Marcus Ott

Paderborn 1998

Organization Design as a Groupware-supported Team Process

GroupOrga — Participative and Distributed Organization Design for
Office Information and Workflow Management Systems

Dokumentationsband zur **Dissertation** von **Dipl.-Wirt. Inform. Marcus Ott**

Paderborn 1998

CONTENTS

Chapter A - Examined Organization Models	1
A.1 An Analysis of Datamodels	1
A.2 Semantic Object Model (SOM) by Ferstl and Sinz	3
A.3 CIM-OSA Model by the Commission of the European Communities	4
A.4 LIBERO organization model by Seitz et al.....	7
A.5 ODB/OIS by Heilmann et al.	8
A.6 Organization Resource Model (ORM/OIS) by Rupietta.....	11
A.7 The Model of ProMInanD by Karbe et al.	13
A.8 The Model of BONAPART by Krallmann/UBIS	15
A.9 ITHACA Office Object Model by Ang and Conrath.....	17
A.10 Meta-Model Workflow by Derungs, Vogler, and Österle	19
A.11 The FUNSOFT Meta-Model by Gruhn et al.....	20
A.12 Business Model by Reim and Rathgeb	22
A.13 Architecture of Integrated Systems (ARIS) by Scheer.....	24
A.14 ROM by Esswein	26
A.15 The SAP Business Workflow Organizational Model	27
A.16 Office Model One (OM-1) by Ishii et al.	29
A.17 The OGM Meta-Model for Business Process Modeling by Rohloff	31
A.18 The IBM-FlowMark Organization Model	32
A.19 The Model of TOSCA by Prinz	34
Chapter B - Evaluation of Tools for Organization Design	39
B.1 List of Vendors of BPR Applications	39
B.2 Overview of the Results of a Product Evaluation	40
B.2.1 Tools for Drawing and Presentation Purposes	41
B.2.2 Tools for Design and Subsequent Analysis	42
B.2.3 Tools for Modeling and Optimization.....	44
B.2.4 Complex Organizational Tools with Integrated Functionality	49
B.3 Available Organizational Entities in the Examined Tools	50
B.4 Detailed Information about Selected Tools of the Survey	52
B.4.1 ARIS-Toolset by IDS Prof. Scheer GmbH.....	52
B.4.2 BONAPART by UBIS GmbH.....	58

B.4.3 Nautilus by integralISA GmbH.....	61
B.4.4 VISIO by VISIO GmbH.....	64
B.5 Results of the Benefit Analysis applied in the Market Survey	66
Chapter C - Introduction to the X.500 Standard Recommendations.....	75
C.1 The Structure of the X.500 Standards.....	75
C.2 The Directory Model	76
C.3 The Information Model.....	77
C.3.1 Directory User Information Model	77
C.3.1.1 Directory Entries	78
C.3.1.2 Attributes.....	78
C.3.1.3 Object Classes	78
C.3.1.4 Directory Information Tree	80
C.3.1.5 Names.....	81
C.3.1.6 Matching Rules	81
C.3.2 Directory Operational and Administrative Information Model	81
C.4 Distribution of the Directory	83
C.4.1 Directory Administrative Authority Model	83
C.4.2 DSA Information Model	85
C.4.3 Managing User Requests in the Distributed Directory	88
C.4.4 Replication of Directory Information.....	90
C.5 The Directory Protocols.....	91
C.5.1 The Directory Access Protocol - DAP.....	92
C.5.2 The Directory System Protocol - DSP	92
C.5.3 The Directory Operational Binding Protocol - DOP.....	92
C.5.4 The Directory Information Shadowing Protocol - DISP.....	93
C.6 The Directory Services	94
C.6.1 Bind Operations	94
C.6.2 Read Operations.....	95
C.6.3 Search Operations	96
C.6.4 Modify Operations	98
C.7 The Directory Schema	101
C.7.1 Object Class Definitions	102
C.7.2 Attribute Type Definitions	103
C.7.3 Matching Rule Definitions.....	103
C.7.4 DIT Structure Definitions	104
C.7.5 DIT Content Rule Definitions.....	104
C.7.6 The Directory System Schema.....	105
C.7.7 The Directory Schema Administration	106

C.8 Authentication	107
C.8.1 Simple Authentication	107
C.8.2 Strong Authentication.....	108
C.9 Authorization - Access Control.....	110
C.9.1 The Basic Access Control Model	111
C.9.2 The Simplified Access Control Scheme.....	114
C.10 The Object Classes and Attribute Types Specified in the Recommendations	114
C.11 X.500 Summary	119
C.12 The Lightweight Directory Access Protocol (LDAP)	120
C.12.1 Origination of LDAP	120
C.12.2 Comparison between LDAP and X.500	121
C.12.3 The future of LDAP - Version 3.....	121
Chapter D - The Entity Relationship Model and Extensions	123
D.1 Entity Relationship Model	123
D.1.1 Description	123
D.1.2 Evaluation.....	125
D.2 Extended entity relationship models	125
D.2.1 Description	126
D.2.2 Evaluation.....	127
Chapter E - A Continuum of Organizational Design Users	129
E.1 The User Class "Intensive Changes"	130
E.2 The User Class "Regular Changes"	131
E.3 The User Class "Occasional Adaptations"	132
E.4 The User Class "Administration of One's Own Data".....	133
E.5 The User Class "Read Access Only"	134
E.6 General Considerations about the User Classes	135

LIST OF ILLUSTRATIONS

Figure A-1: The Organization View with the CIM-OSA model as described by Beekmann.....	5
Figure A-2: Conceptual data model of CIM-OSA.....	6
Figure A-3: LIBERO organization model	8
Figure A-4: Conceptual model of ODB/OIS	10
Figure A-5: Relations in the ODB/OIS conceptual model	11
Figure A-6: ORM conceptual model	13
Figure A-7: Conceptual model of ProMInanD	14
Figure A-8: Conceptual model of BONAPART	16
Figure A-9: ITHACA meta-model	17
Figure A-10: Conceptual model of the ITHACA office object model	18
Figure A-11: Conceptual model of Meta-model Workflow	19
Figure A-12: Conceptual data model of the WfMS Leu	21
Figure A-13: Conceptual model of Rathgeb and Reim's business model	23
Figure A-14: Conceptual model of ARIS as presented by Galler	25
Figure A-15: Conceptual model of ROM.....	27
Figure A-16: Conceptual model of SAP Business Workflow	28
Figure A-17: Conceptual model of OM-1	30
Figure A-18: Conceptual model by Rohloff.....	32
Figure A-19: Conceptual model of IBM-FlowMark	33
Figure A-20: Conceptual model of TOSCA.....	36
Figure B-1: Sample organization drawn with Chartist	41
Figure B-2: Sample organization drawn with Aufbau-Profi	43
Figure B-3: Sample organization modeled with ProAS\Doc.....	44
Figure B-4: Sample organization modeled with MetaDesign	45

Figure B-5: Sample organization modeled with INCOME Mobile	46
Figure B-6: Sample organization modeled with PRISMA-Tool.....	47
Figure B-7: Sample organization modeled with SDW-OM.....	48
Figure B-8: Sample organization modeled with AENEIS	50
Figure B-9: Views and levels of the ARIS method.....	53
Figure B-10: Sample organization modeled with a German ARIS-Toolset V 3.1	57
Figure B-11: Scenarios for structural organization in BONAPART	60
Figure B-12: Sample organization modeled with BONAPART.....	60
Figure B-13: Information about an instance of employee in Nautilus	63
Figure B-14: Relationship of entities in Nautilus	64
Figure B-15: Sample organization modeled in VISIO.....	66
Figure C-1: Directory structure	77
Figure C-2: Custom object class organizationalPosition	79
Figure C-3: Custom object class softwareAgent.....	79
Figure C-4: Custom object class substitutionRule	80
Figure C-5: Custom object class informationObjectReference	80
Figure C-6: Custom object class embeddedInformationObject	80
Figure C-7: Structure of entities in the DIT	80
Figure C-8: Sample DIT subtree specification.....	82
Figure C-9: Administrative areas	84
Figure C-10: DSA structure	85
Figure C-11: DSEs and the different directory models.....	87
Figure C-12: Chaining	88
Figure C-13: Referral	89
Figure C-14: Distributed name resolution	90
Figure C-15: Shadowing	91
Figure C-16: Overview of the directory protocols	92
Figure C-17: Overview of the directory schema.....	102
Figure C-18: Protected items	112
Figure C-19: The access control decision function.....	114

Figure C-20: DIT structure suggested in X.521	118
Figure C-21: LDAP Architecture	120
Figure D-1: Meta-model of the EER model	124

LIST OF TABLES

Table A-1: Types of conflict.....	1
Table A-2: Comparison options	2
Table B-1: Product vendors of BPR tools	40
Table B-2: Overview over organizational entities.....	52
Table B-3: Description methods at the levels of the organizational view in ARIS.....	53
Table B-4: Organizational entities of the ARIS organizational model.....	55
Table B-5: Relationship types in ARIS	56
Table B-6: Abstract organizational entities in the BONAPART meta-model	59
Table B-7: Organizational entities of the Nautilus organizational model.....	62
Table B-8: VISIO organizational chart master shapes	65
Table B-9: Criteria of the benefit analysis and valuation	73
Table C-1: Knowledge reference types.....	86
Table C-2: Operations of the DOP	93
Table C-3: Operations of the DISP.....	93
Table C-4: Read operation.....	95
Table C-5: Compare operation	96
Table C-6: Abandon operation	96
Table C-7: List operation.....	97
Table C-8: Search operation	98
Table C-9: addEntry Operation	99
Table C-10: removeEntry Operation	99
Table C-11: modifyEntry Operation.....	100
Table C-12: modifyDN Operation.....	101
Table C-13: Message	110

Table C-14: Permission categories	113
Table C-15: Selected object classes	116
Table C-16: Selected attribute types defined in X.520	118
Table E-1: GroupOrga scale of varying requirements by different user type classes	130

Chapter A

Examined Organization Models

Chapter 4 has given reference to some of the preliminary projects and researches that inspired the development of the GroupOrga Enterprise Information Management Model, the GEIMM. Various organizational models have been investigated and their advantages and drawbacks have been taken into consideration in order to develop a comprehensive organizational model. This chapter gives a short representation of most organizational or data models that were examined. Their major characteristics, entities and relations are shown and compared in this chapter.

A.1 An Analysis of Datamodels

In the course of various smaller, consecutive analyses of organizational models in the GroupOrga project, various forms of conflict have been considered (cp. [Hars 1994], pp. 178f.) in order to allow for a helpful comparison (see Table A-1).

Form of conflict	Description
Name conflict	Usage of synonyms and homonyms
Type conflict	Usage of different methodical concepts
Structural conflict	Usage of semantical contradictions

Table A-1: Types of conflict

Synonyms occur for words having the same or a similar meaning. They have to be distinguished from homonyms. Homonyms are one of two or more words that have the same sound and often the same spelling but differ in meaning. Type conflicts in the comparison of two data models occur, when the modeling of the same real-life situation has been done differently due to some freedom in the modeling concepts and because of varying methodical

approaches. Structural conflicts in compared models arise from similar structures in the model, which however exclude themselves because of semantical contradictions.

In Table A-2 some criteria are listed that allow for a comparison of the examined organizational models. This, however, assumes that the mentioned conflicts have been resolved and that the models are available at the same detailed level (cp. [Rosemann 1996], chapter 2.5.1).

Option	Description
Comparison of entity types	Number and form of entity types give information about the model's flexibility
Comparison of relationship types	Relationship types also give information about flexibility
Comparison of cardinality	Allows to draw conclusions on the model's restrictions
Comparison of attributes	For identification of freedom in modeling options

Table A-2: Comparison options

Both, the number and form of entity and relationship types give information about the flexibility that can be gained in an office or workflow management system if this particular organizational model is used as its base.

The more entity types are available in an organizational model, the more flexibility in the modeling process is given to the designers. In case of a similar amount of entity types for two models, the number of available relationship types is yet another indicator for flexibility.

Cardinality indicates how many instances of two entities can be in relation. Hence, a (0,1)-(1,1)-relation provides with lesser flexibility than a (0,n)-(0,m)-relation.

Attributes give entity and relationship types their individual characteristics. The number of attributes for the same entity type (e.g. person) may vary between different models which thus enlarges or reduces the freedom of modeling.

In sections A.2 to A.19 various meta-models will be introduced with a focus on their organization model, and their complex comparison served as starting point for the GEIMM. The result of this comparison has been described in chapter 4 in form of the GroupOrga enterprise model. In contrast to section 4.3.4 only concrete models (as opposed to office procedure systems, knowledge bases etc.) will be described here. Besides giving an overview of the respective model and its distinguishing characteristics, the main goal of the following sections is also to support the practitioner and the researcher in finding further readings about the data model. The section headings give the model's name or acronym first, then the name(s) of the major developer(s), if applicable. The sorting of the following sections does not indicate any preferences or major advantages of one data model over another. In other words, there is no deliberate order, ranking or systematization in how the respective data models are presented.

A.2 Semantic Object Model (SOM) by Ferstl and Sinz

Description

According to the authors of the Semantic Object Model (SOM), a business process is a transaction or a sequence of transactions on business objects ([Ferstl/Sinz 1995], p. 214). This general idea explains the main concepts of SOM: each transaction in a business process is represented by a task and a business object is assigned to such a task. For describing these circumstances, SOM knows a *conceptual object model* which is considered mainly static and a dynamic *procedural object model*. The object system is that part of the static conceptual object model which covers the organizational situation with its relations to the environment. After having defined this object system, an *interaction model* and a *task system* are designed. The *interaction model* describes which general transactions have to be performed on the objects and the *task system* explains which tasks have to be carried out in which order for describing the flow of work.

The conceptual object model, which is of main interest here, combines two basic models: a semantic datamodel which is based on the extended entity relationship model, and an object model which mainly bases on the object-oriented programming language Smalltalk.

Theoretical model, prototype, or product

SOM is mainly a theoretical model for describing business processes. In connection with the corresponding modeling tool (see [Ferstl et al. 1994]), the SOM can be used to design and explain processes. It has not yet found its way into a product.

Primary modeling focus

SOM's primary focus is on business *processes*. Its elements allow to describe task objects which simultaneously represent the flow of work (business process) and the resources needed to perform the task.

Model entities

SOM bases on a general extended entity relationship model approach and has no entities predefined which specifically address an organization's structure.

Model relations

SOM bases on a general extended entity relationship model approach and has no relations predefined which specifically address relationships between organizational entities.

Tool support

A tool support for designing models with SOM is available with the tool SOM-CASE.

Authors and references

Otto K. Ferstl and Elmar J. Sinz

[Ferstl/Sinz 1990], [Ferstl et al. 1994], and [Ferstl/Sinz 1995]

Ferstl, Otto K.; Sinz, Elmar J.: Ein Vorgehensmodell zur Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM), in: Wirtschaftsinformatik, 6, 1991, pp. 477-491.

Ferstl, Otto K.: Der Modellierungsansatz des Semantischen Objektmodells (SOM), Bamberger Beiträge zur Wirtschaftsinformatik, Nr. 18, 1993.

Ferstl, Otto K.; Sinz, Elmar J.: Geschäftsprozeßmodellierung, WI-Schlagwort, in: Wirtschaftsinformatik, Vieweg, Wiesbaden, 6, 1993, pp. 589-592.

Ferstl, Otto K.; Sinz, Elmar J.: Multi-layered development of business process models and distributed business application systems, An object-oriented approach, Bamberger Beiträge zur Wirtschaftsinformatik, 1994.

A.3 CIM-OSA Model by the Commission of the European Communities

Description

The objective of CIM-OSA (Computer Integrated Manufacturing–Open Systems Architecture) is to ensure the integration of the enterprise at the level of the business functions. First, the business requirements of the enterprise are captured in the Enterprise Model and, starting from these requirements, the Implementation Model (which is the physical CIM system) is derived. CIM-OSA defines three levels of modeling: *enterprise*, *intermediate*, and *implementation* (cp. Figure A-1). The Enterprise Model, which is the most interesting for this research, describes in a business sense and terminology what needs to be done within the enterprise. According to four different viewpoints, each of the three modeling levels is described in terms of the *function view*, the *information view*, the *resource view*, and the *organization view*. Due to their importance here, the latter two are again the focus. The resource view describes and organizes the enterprise's resources and the organization view fixes its organizational structure. Most efforts have so far been spent on the function view of the Enterprise Model in CIM-OSA. It describes what has to be done using business process building blocks. The information view structures the information inputs and outputs of the enterprise activities.

Building an Enterprise Model in CIM-OSA includes defining the required resources as inputs of the enterprise activities. This set of inputs is to be structured to provide a consistent view of all resources needed. The resource view is the basis for the further organization of the resources in terms of physical location, and for identifying responsibilities.

Theoretical model, prototype, or product

CIM-OSA provides a framework and concepts for developing a CIM system, prototype implementations are described in literature.

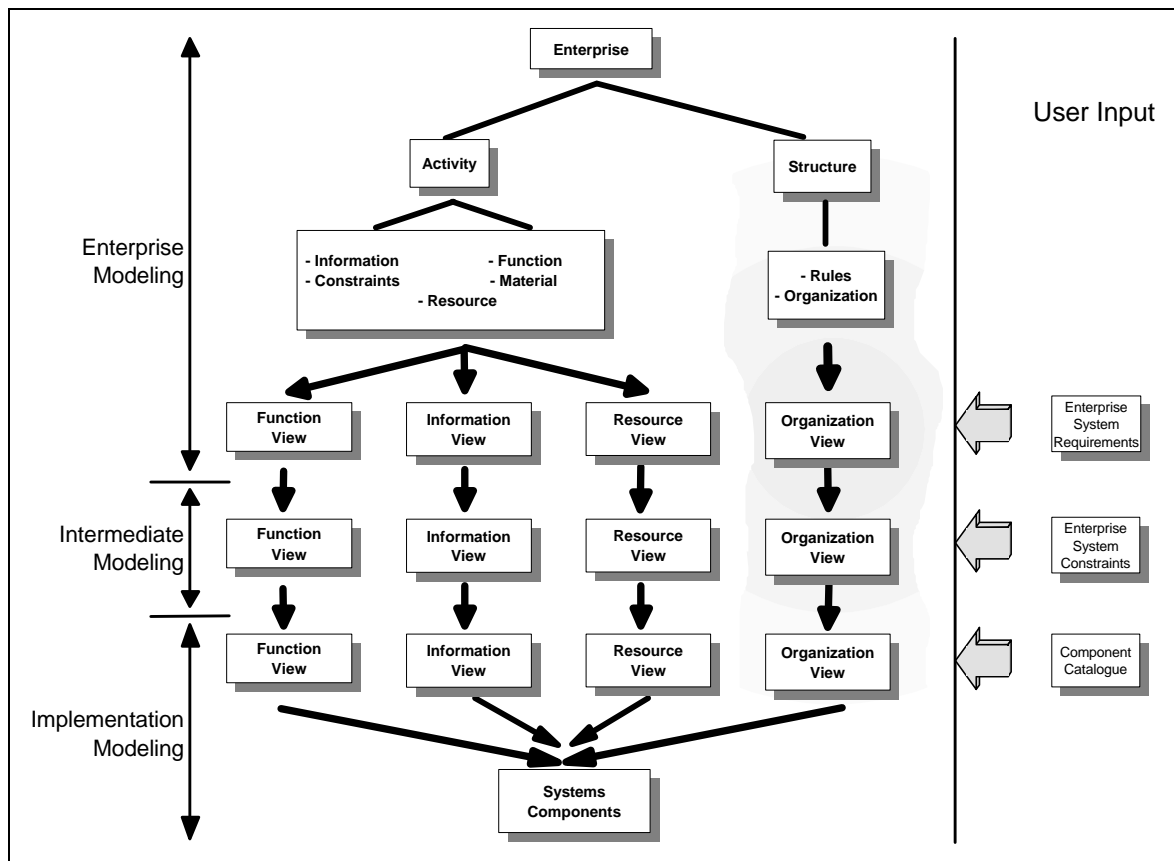


Figure A-1: The Organization View with the CIM-OSA model as described by Beekmann

Primary modeling focus

CIM-OSA's modeling focus is on manufacturing *processes*. A manufacturing enterprise consists of various departments and resources. CIM-OSA aims at an optimal use of those resources required to master the action flow and the information flow when they are part of a CIM system. Structure modeling is considered part of this, but not the main aspect.

Model entities

Enterprise activity: Enterprise activity is a construct which is used to define internal and external real world activities which are started as a result of the activation of the associated procedural rule set. A procedural rule set is triggered by an enterprise activity.

Business process: The business process is a construct which is used to define what has to be done (by enterprise activities and other business processes).

Procedural rule set: A procedural rule set defines the desired sequence of the enterprise activities in the form of a flow of control.

Transformation function: In the particular model the transformation function describes the required functionality (i.e. the activities) of the enterprise activity.

Resource inputs/outputs: The resource inputs are enterprise objects which are utilized by the enterprise activity. The resource inputs may be partially or wholly consumed by the enterprise

activity. Resource outputs describe the resources as they are returned to the resource pool after execution of the enterprise activity.

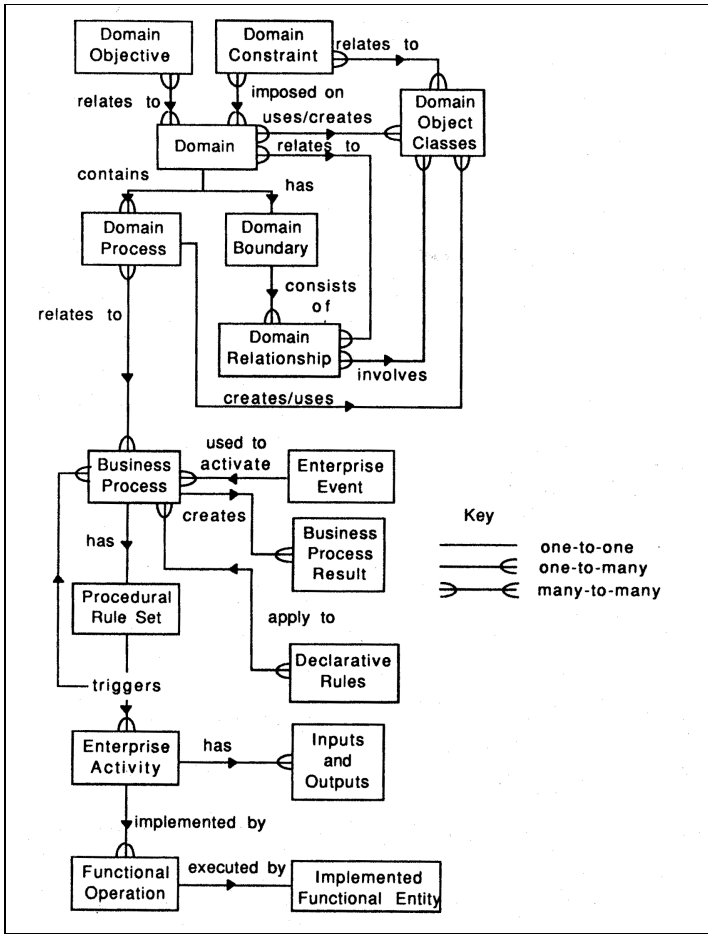


Figure A-2: Conceptual data model of CIM-OSA

Functional entity: A functional entity is an active functional object (such as machines, people, robots, computers, database management systems, etc.) able to send, receive, process, and optionally store information.

Functional operations: The functional operation is a construct which is used to specify how the functionality of the enterprise activity will be implemented in terms of functional entities.

Model relations

Since no further specification of *functional entity* is given, no explicit relations between the mentioned organizational entities (such as machines, people, robots, computers, database management systems, etc.) is defined.

Tool support

CIM-OSA has developed architectural concepts facilitating the building and updating of a CIM system and a build-time support toolset: the Integrated Enterprise Engineering toolset which is graphically oriented.

Authors and references

The CIM-OSA architecture is developed by a research project which has been setup by the Commission of the European Community in the framework of the ESPRIT program. Explanations and further references can, for instance, be found in:

Beekmann, Dirk: CIM-OSA: Computer Integrated Manufacturing - Open Systems Architecture, in: Computer Integrated Manufacturing, Vol. 2, No. 2, 1989, pp. 94-105.

Joryz, H.R.; Vernadat, F.B.: CIM-OSA Part1: Total Enterprise Modelling and Function View, in: Int. J. Computer Integrated Manufacturing, 3(3,4), 1990, pp. 144-156.

A.4 LIBERO organization model by Seitz et al.

Description

The organizational model of LIBERO (Literatur-Bestellung, -Retrieval und -Organisation) which is described in [Seitz/Galster/Lang 1993] has been developed for a specific purpose: the description of a university research institute in order to implement the particular workflow for ordering literature. What seems to be a very narrow restriction in the first place, appears to be a rather general data model after all. LIBERO describes a simple, generic data model for organizational structures and it provides constructs for process modeling.

Theoretical model, prototype, or product

The organization model has been implemented in Superbase 4 V1.2, a MS-Windows-based network enabled database environment. Both, the organization model and the ordering process have been realized with this environment as the prototype system LIBERO.

Primary modeling focus

LIBERO's main use is for ordering books from a library. However, from the conceptual viewpoint it focuses at both, the procedural and the organizational model in an equal way.

Model entities

Position: Positions are the smallest organizational entity of LIBERO. They represent a combination of similar activities which are oriented towards a specific task. A position can be occupied by one or more members of *personnel*.

Organizational unit: Positions are aggregated to organizational units. An organizational unit can be both, a project-oriented grouping of persons or an organizational grouping in the structural hierarchy (such as a department, a unit, etc.).

Resource: Resources, such as working material, tools, financial resources, information, represent the means for persons to perform the tasks in the business processes.

Personnel: Instances of the entity *personnel* relate to the actual employees of an enterprise. They are integrated into the organizational hierarchy. By assigning a person to a position, this person becomes the holder of the position.

Role: A role represents a combination of all expected behavioral patterns of a person. These expectations are represented by a person's environment. By assigning roles to holders of positions, authorizations and rights are transferred to the person in question.

Task: Tasks are defined in respect to a specific outcome, i.e. not in respect to what has to be done. In other words, tasks represent overall goals that have to be reached, rather than particular activities that have to be carried out.

Group: The entity *group* is not explicitly defined in the LIBERO organization model, however, it is considered an organizational unit, as well. Due to this definition, Seitz, Galster, and Lang speak about groups as single entities.

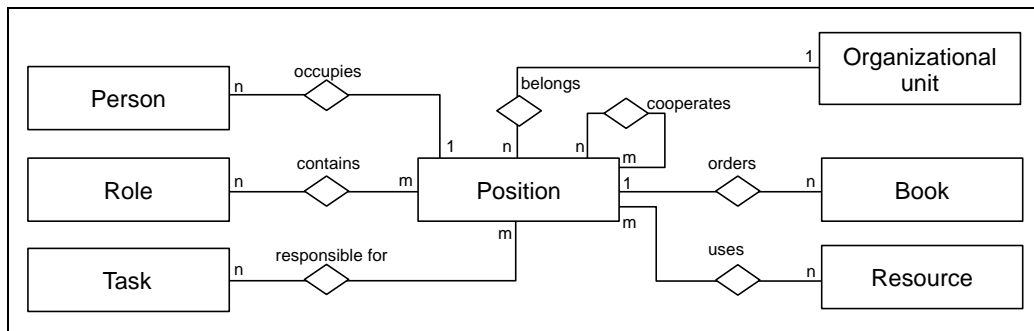


Figure A-3: LIBERO organization model

Model relations

A person occupies a position and a position may be occupied by many persons. Positions contain many roles and vice versa, positions are responsible for many tasks and many positions may be responsible for the same task. A position belongs to an organizational unit, while an organizational unit may comprise many positions. A book is ordered by exactly one position, but positions may order many books. For doing this, positions use many resources. All entities and relations of the LIBERO organization model are shown in Figure A-3.

Tool support

The LIBERO system provides tool support for the ordering process as such, but no tool support for the maintenance of the data models in the databases (except the DBMS's front-end itself).

Authors and references

The organizational model described here has been presented by Seitz, Galster, and Lang—a team of Bodendorf and Mertens at the Institute of Wirtschaftsinformatik 2 at the University of Erlangen-Nuremberg.

R. Seitz, C. Galster, A. Lang, Freimut Bodendorf, and Peter Mertens

[Seitz/Galster/Lang 1993]

A.5 ODB/OIS by Heilmann et al.

Description

The starting point of the ODB/OIS project (Organizational Database/Organization Information System) was the unnecessarily high effort and time to update and administrate conventional organizational handbooks. An additional reason for its development is the limited ability to analyze the data in conventional handbooks. The goal of the Organization Database part is the

complete representation of all organizationally relevant aspects within an enterprise in a structural and procedural model as well as their subsequent analysis. For the representation of an organization, organizational entities (with attributes) and relations (in form of tables) are defined according to specifications in the entity relationship model (cp. chapter D). The Organization Information System allows to create and manage the data stored in the ODB, to retrieve single entities and their attributes, to poll predefined analyses, to find organizational weak points, to perform *What-If* analyses, and to retrieve statistical, organizational characteristics.

Theoretical model, prototype, or product

According to v. Kortzfleisch ([1993], p. 32), both ODB and OIS are no market-ready products but—in part incomplete—prototypes.

Primary modeling focus

ODB/OIS's main goal is to define a general organizational model for the representation of organizational structures and their environment which is represented in a meta-database. The result aims at a simplification of producing and managing organizational structures and organizational handbooks.

Model entities

The entities of OIS are numbered according to their appearance in Figure A-4 and are transferred into English.

1. *Position*: Used in the common, organizational sense.
2. *Instance*: Each performing entity in an enterprise, as well as temporary teams.
3. *Personell*: Represents the personal data in human resource systems or copies of it in OIS.
4. *Decision powers*: Can be assigned to a position, a job, or a specific person.
5. *Training*: Any sort of training that is related to the job itself, i.e. not to a particular person.
6. *Requirements*: To-be-qualifications of holders of jobs or positions. It can also be assigned to persons.
7. *Workspace/Location*
8. *Room*
9. *Organizational means*: Used in the common, organizational sense.
10. *Reports*: Can be created manually or with computer support.
11. *Formats*: Can be represented as forms or as descriptions about screen layouts and lists.
12. *Information*: May represent manual or computer-supported editing or storing of information.

- 13. *Information management*: This entity serves as interface to models of other computer-based applications for information management.
- 14. *Tasks*: Are assigned to jobs or positions.
- 15. *Work descriptions*: Specify how tasks have to be performed, in which order, by whom.
- 16. *Work flows*: Further refine and detail work descriptions.
- 17. *Activities*: Further refine and detail work descriptions.
- 18. *Business processes*: Represent triggers that invoke work descriptions.
- 19. *Rank*: The rank of a position defines ranges of wages and promotion prospects.
- 20. *Signatory power*: Defines which business processes can be decided about by whom.
- 21. *Job*: Subsumes positions with more or less similar characteristics. Differing positions are distinguished by their attributes.

Figure A-4 shows the entities in the complete conceptual model of ODB/OIS (from [Heilmann/Simon 1989], p. 200).

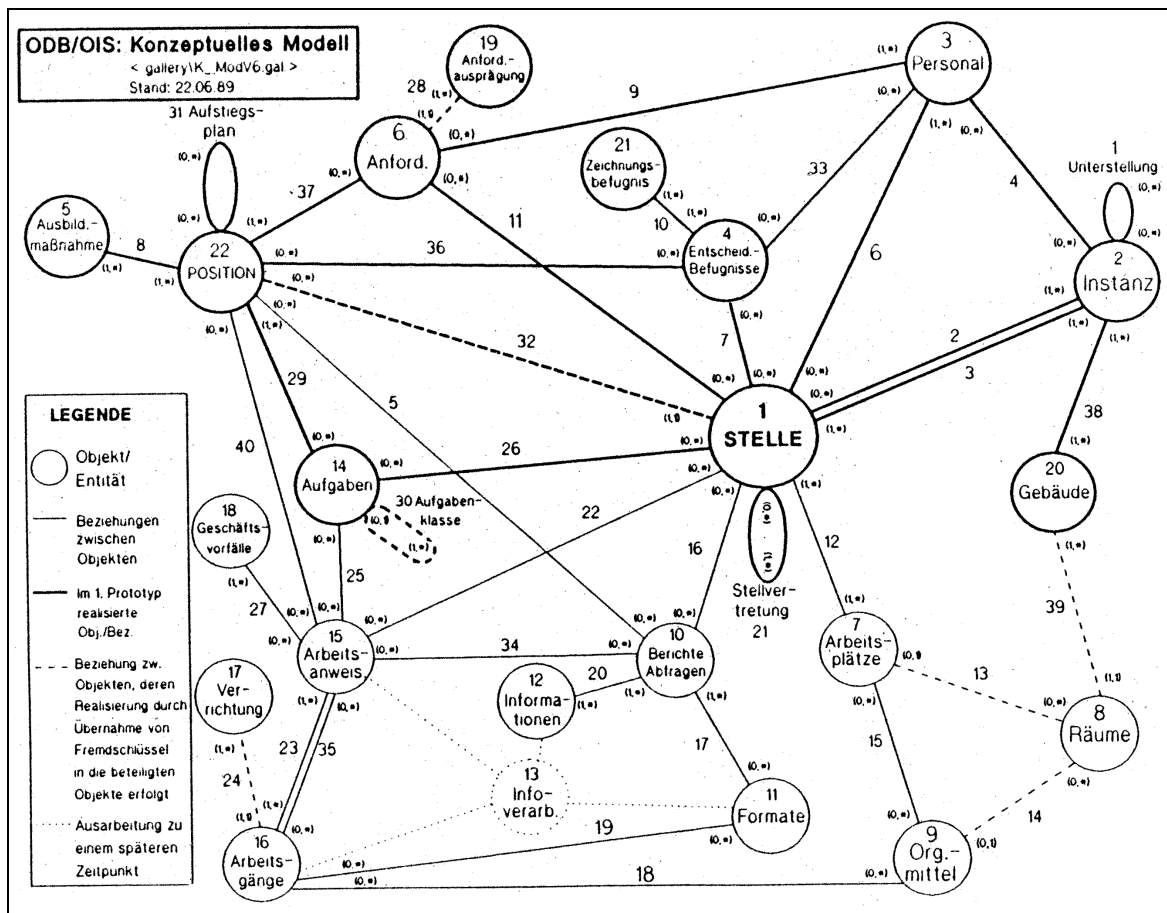


Figure A-4: Conceptual model of ODB/OIS

Model relations

Figure A-5 has the original list of all existing relations in the conceptual data model as found in [Heilmann/Simon 1989] (p. 201).

1 Instanz -ist unterstellt- Instanz	21 Stelle -vertritt- Stelle
2 Stelle -leitet- Instanz	22 Stelle -arbeitet mit- Arbeitsanweisung (insbesondere Universalanweisung)
3 Stelle -gehört zu- Instanz	23 Arbeitsanweisung -besteht aus- Arbeitsgängen
4 Gremien-Beziehung (Diese Beziehung darf nur auf ein Kollegium/Team gerichtet sein, nicht auf eine Instanz im engeren Sinne)	24 Arbeitsgang -beinhaltet- Verrichtungen
5 Position -bearbeitet- Berichte/Abfragen	25 Aufgabe -erfolgt nach- Arbeitsanweisungen
6 Stelle -ist besetzt von- Personal	26 Stelle -hat- Aufgaben
7 Stelle -hat- Entscheidungsbefugnisse	27 Geschäftsvorfall -wird bearbeitet mit- Arbeits- anweisungen
8 Ausbildungsmaßnahme -ist vorgesehen für- Position	28 Anforderung -hat- Anforderungsausprägung
9 Personal -besitzt- Anforderungen (Istqualifikation)	29 Position -hat positionstypische- Aufgaben
10 Zeichnungsbefugnis -ist verbunden mit- Entscheidungs- befugnis	30 Aufgabenklasse -beinhaltet- Aufgaben
11 Stelle -hat- Anforderungen (Sollqualifikation)	31 Position -hat- Aufstiegsplan
12 Stelle -hat- Arbeitsplatz	32 Stelle -entspricht- Position
13 Arbeitsplatz -befindet sich im- Raum	33 Personal -hat- Entscheidungsbefugnisse
14 Raum -hat feste- Organisationsmittel	34 Arbeitsanweisung -beschreibt Erstellung von- Bericht
15 Arbeitsplatz -hat- Organisationsmittel	35 Arbeitsanweisung -als Zusatzinformation bei- Arbeitsgang
16 Stelle -bearbeitet- Berichte/Abfragen	36 Position -hat positionstypische- Entscheidungs- befugnisse
17 Bericht -hat- Format	37 Position -hat positionstypische- Anforderungen
18 Organisationsmittel -wird verwendet bei- Arbeitsgang	38 Instanz -befindet sich im- Gebäude
19 Format -wird verwendet bei- Arbeitsgang	39 Raum -befindet sich im- Gebäude
20 Bericht -enthält- Information	40 Position -hat positionstypische- Arbeitsanweisung

Figure A-5: Relations in the ODB/OIS conceptual model

Tool support

The organizational model and its relations are modeled in the relational database system *Professional Oracle*. Its own tools can be used to design the data model and the system and integrated C-routines also provide access to the data.

Authors and references

Heidi Heilmann, Wolfgang Sach, and Manfred Simon

[Heilmann/Simon 1989], [v. Kortzfleisch 1993] (pp. 31f.)

Heilmann, Heidi: Entwurfsentscheidungen bei der Gestaltung eines Organisationsinformationssystem, in: Kurbel, K.; Mertens, P.; Scheer, A.-W. (Hrsg): Interaktive betriebswirtschaftliche Informations- und Kommunikationssysteme, Berlin, 1989, pp. 315-328.

Heilmann, Heidi; Sach, W.; Simon, M.: Organisationsdatenbank und Organisationsinformationssystem, in: Handbuch der modernen Datenverarbeitung, 25, 142, 1988, pp. 119-129.

A.6 Organization Resource Model (ORM/OIS) by Rupietta

Description

The Organization & Resources model (OR model) defines a conceptual data model for representing organizational structures and resources. Its implementation is a database containing a representation of the organizational structure of a company, its actors and resources. This database provides the information necessary to adapt services of an office system to organizational requirements. For OR model relational databases were preferred to object-oriented approaches because of their general and widespread availability.

The OR conceptual model emerges as an object-oriented design consisting of a set of related object classes which represent concepts of organization theory. The object classes are then mapped onto relations to be stored in a relational database with a SQL interface.

Theoretical model, prototype, or product

The OR model is the base for the Organization and Resource Management system (ORM) which provides application systems with information about the underlying enterprise organization. A single database is available to different application systems (cp. section 5.5.5). ORM has recently been renamed to OIS (Organization Information System) and serves as the commercially available organizational design environment for the WorkParty WfMS ([Siemens Nixdorf 1997b]).

Primary modeling focus

The OR model aims at representing the organizational structure of an enterprise or public authority. Except the *task* entity it does not cover any other but the organizational sub-model.

Model entities

Employee: The people working in the organization. Employees are integrated into the organization by their assignment to positions.

Organizational unit: Sets of positions with common or related tasks. They are grouped to form a larger operational unit. They can form hierarchies by means of sub- or superordination.

Position: It represents the workplace of an employee. Positions are the basic entities to make up an organization in ORM.

Organizational role: Functions such as department manager, secretary or clerk which are assigned to positions. Organizational roles summarize all employees who bear certain characteristic tasks or authorizations in common. Organizational roles are *process-related*.

Authority: Authority is a combination of authorizations, rights and responsibilities. Authorizations are expressed by resource assignments or for tasks like signatures. Rights are given to access or use resources, and responsibilities, in turn, are expressed by task assignments.

Task: A task is a goal set for the activities performed by employees.

Resource: Work objects like documents, working materials and tools like application programs. Resources may be compound objects, i.e. they can be subordinated and can have subordinate resources themselves.

Model relations

Figure A-6 shows the conceptual model of ORM (cp. [Rupietta 1994], p. 4). Its relations are also displayed and are explained in the following.

Authorities can be assigned to either employees, positions, roles, or organizational units. Tasks and resources can be assigned to authorities which then bundle tasks and required resources. Organizational units and roles can be hierarchically ordered, positions are assigned to organizational units and employees as well as roles can be assigned to positions. One of an organizational unit's positions can be marked as lead position and one of an employee's positions is the regular position. Employees can be assigned substitutes for other positions.

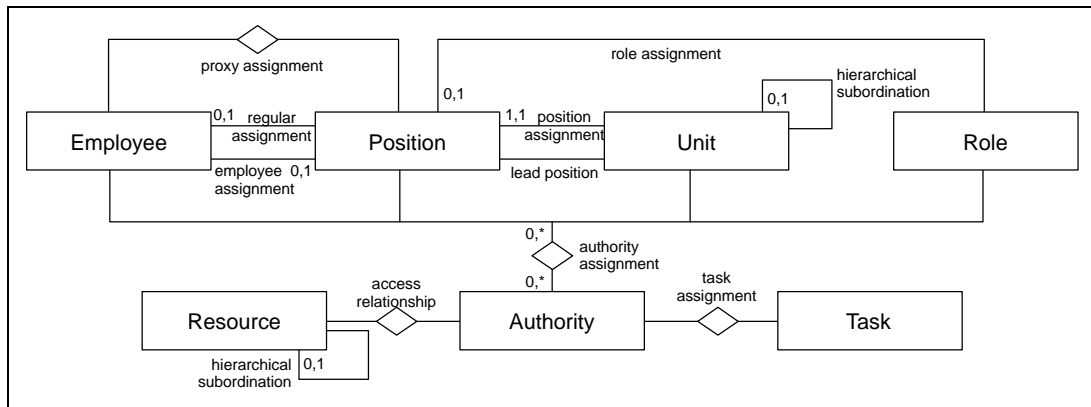


Figure A-6: ORM conceptual model

Tool support

OIS has a non-graphical, windows based front-end to create, modify, and display the organizational model. It, for instance, allows to create and edit entities and their attributes, to display the relationship of entities with each other, and to define new relations. The front-end has no graphical view of the model.

Authors and references

Walter Rupiotta

[Rupiotta 1990], [Rupiotta 1992], [Rupiotta 1994], [Rupiotta 1997]

Rupiotta, Walter; Wernke, Gerd: Umsetzung organisatorischer Regelungen in der Vorgangsbearbeitung mit WorkParty und ORM, in: U. Hasenkamp, K. Kirn, M. Syring (eds.): CSCW Computer Supported Cooperative Work: Informationssysteme für dezentralisierte Unternehmensstrukturen, Addison-Wesley, Bonn etc., 1994, pp. 135-154.

Rupiotta, Walter: Ein Modell zur organisationsbestimmten Verwaltung von Zugriffsrechten, in: Bauknecht, K.; Karagiannis, D.; Teufel, S. (eds.): Sicherheit in Informationssystemen, vdf Hochschulverlag AG, Zürich, 1996, pp. 53-67.

A.7 The Model of ProMinanD by Karbe et al.

Description

According to Karbe, Ramsperger, and Weiss ([1990], p. 110), a common conventional tool for supporting the processing of office tasks is the circulation folder. It consists of various related

tasks being worked on by office workers. ProMInanD mimics the circulation folders by electronic circulation folders. The base of the ProMInanD system is a description of the organizational structure according to traditional theoretical definitions in the relational *organization database*. For this description, the authors of ProMInanD have defined an organization model for this organizational handbook which relates to organizational roles.

A characteristic of the organization model is the assignment of a person to a task rather by means of its function in the organization than by its hierarchical subordination, its rank, or position. Such a function, which is called *role*, can be defined in two ways: roles may be generally valid for the whole organization or roles may only be valid in relation to a particular process or task performance. In other words, such relative roles may exist only temporarily or only in relation to a very specific workflow.

Theoretical model, prototype, or product

The organization model described here has been defined as the base of the commercially available system ProMInanD. ProMInanD is implemented in Objective-C on Sun workstations and it employs the TransBase database system.

Primary modeling focus

ProMInanD is a system for control and steering of business processes. The primary goal of its model is to support formalized and unstructured processes, as well as their deviations from predefined situations. Organizational structures are modeled as a secondary matter.

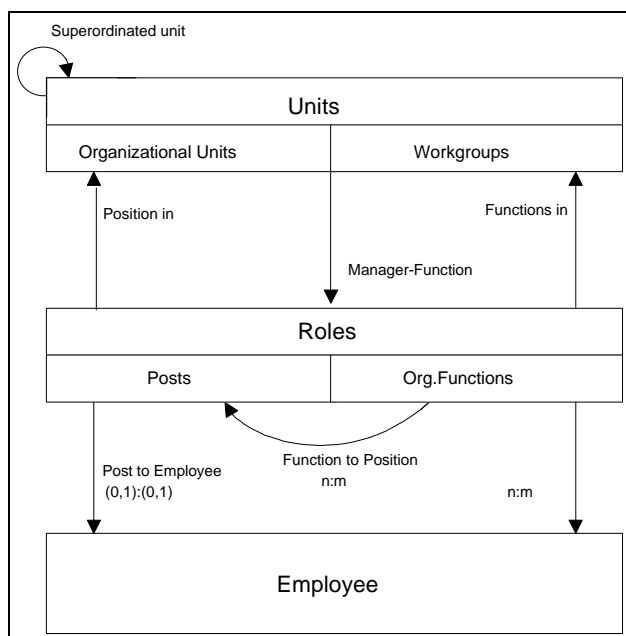


Figure A-7: Conceptual model of ProMInanD

Model entities

Units: Units are organizational units, workgroups, and groupings of units.

Posts: Descriptions such as "head of department <dept>". They can be hierarchically subordinated and held by employees.

Employees: The office workers employed in the organization.

Organizational functions: Functions for posts such as "managing department <dept>" or "member of project <proj>" for role.

Role: Describes a function in the organization. Roles exist as generally valid roles or as relative roles. Posts and organizational functions are called (global) roles.

Figure A-7 shows ProMInanD's organizational model as found in [Karbe 1994] (p. 127).

Model relations

Organizational relations exist between units, e.g. "<unit1> is superior to <unit2>", between organizational functions, such as "reports to", and in form of the actual assignments of office workers to roles and posts, such as "<employee> is head of department <dept>". Relations also exist between organizational functions and roles, describing that such functions and roles are always identical. Additional relations are shown in Figure A-7.

Tool support

In order to maintain the electronic organization handbook, ProMInanD offers a graphical organizational chart editor ([Karbe/Ramsperger/Weiss 1990], p. 115).

Authors and references

Berhard Karbe, Norbert Ramsperger, and P. Weiss

[Karbe 1994], [Karbe/Ramsperger/Weiss 1990], [Karbe/Ramsperger 1991]

IABG GmbH - ProMInanD, in: Workflow Management - Groupware Computing, 1997, pp. 175 - 198.

A.8 The Model of BONAPART by Krallmann/UBIS

Description

BONAPART is a complex application environment for organizational modeling. With the application the organizational situation is described in form of meta-models. Through these meta-models no concrete objects are defined yet, but abstract classes of objects are in the modeling focus. For instance, instead of creating the unit *Marketing*, the modeling process first defines such general entities as *main units*, *units*, and *project groups*, and their attributes and relations. With regard to organizational modeling, the BONAPART model knows three different views onto the overall meta-model: the organizational unity model, the leader model, and the position model. Section B.4.2 explores this concept in more detail. Relations between these meta-models specify, for instance, which leader classes are entitled to manage which classes of organizational unities, etc. An organizational chart is the instantiation of objects from the three meta-models.

Theoretical model, prototype, or product

Initially, the development of the various meta-models in BONAPART was lead by Krallmann and his team at the Technical University of Berlin in the course of his research regarding the *Kommunikationsstrukturanalyse* (KSA) (cp. [Krallmann/Klotz 1994] and [Krallmann et al. 1989]). A graphical front-end for the design of the organizational model has been added, and to date BONAPART is a commercially available product.

Primary modeling focus

BONAPART focuses at the optimization of business processes and organizational structures, office automation, computer integrated business, and reorganization in the use of forms (cp. [Krallmann/Klotz 1994], pp. 34f.).

Model entities

Organizational unity: An organizational unity is any type of grouping that is lead by a leader.

Leader of organizational unity: Leaders are those employees that direct organizational unities. Leaders are already assigned to units on an abstract level. The organizational chart automatically shows the correct leader type within the unit.

Position: A position models the scope of duties of one or more persons. Positions are subordinated to organizational units and can be occupied by more than one person.

Person: Employee in an organization which exists in form of an entry in an employee database. No real entity of the organizational model as such.

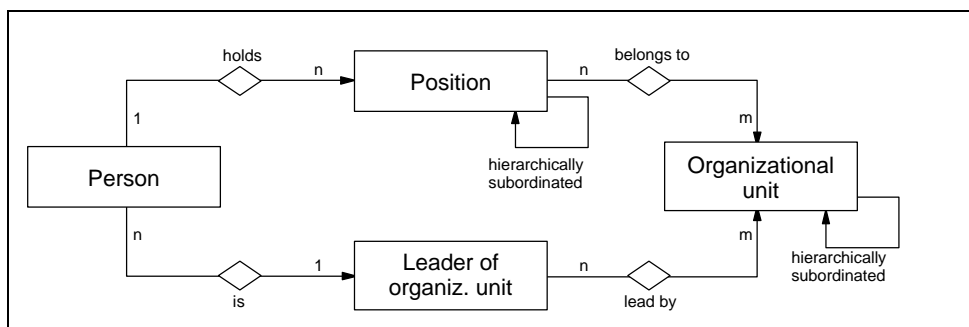


Figure A-8: Conceptual model of BONAPART

Model relations

As shown in Figure A-8, a person can hold many positions, and a person is leader of an organizational unit. A position can be occupied by only one person, many positions may belong to an organizational unit, and positions and organizational units can be hierarchically sub- and superordinated. Moreover, an organizational unit consists of many positions and it may be lead by many leaders. Leaders of organizational units are one or more persons who in turn may lead many organizational units. Further relationship types which are pre-defined by the system are used for specialization (*is_a*, such as *main unit is_a unit*) and hierarchy (*never_over*, such as *project group never_over unit* in a hierarchy). Additional flexibility in the model is gained by the fact that the user can modify and adapt the meta-models with regard to the relations and entities.

Tool support

Graphical application environments are available for the various meta-models in BONAPART, among them a tool for organizational design and analysis.

Authors and references

Herrmann Krallmann

[Krallmann et al. 1989], [Krallmann/Klotz 1994], [Krallmann/Derszteler 1996] and [Bach/Brecht/Österle 1995], pp. 60ff.

UBIS GmbH: BONAPART, Model your own business, Berlin, Dezember, 1992.

A.9 ITHACA Office Object Model by Ang and Conrath

Description

The modeling framework of the ITHACA (Integrated Toolkit for Highly Advanced Computer Applications) is enclosed in a meta-object model. An organization can be viewed as a set of office objects that can be split up ("specialized") into active objects and passive objects (see Figure A-9, from [Ang/Conrath

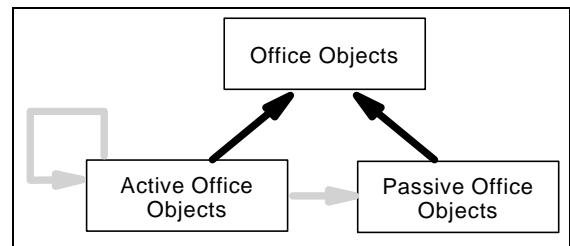


Figure A-9: ITHACA meta-model

1993], p. 7). Each object in ITHACA is described by a set of instance variables and operations. ITHACA clearly distinguishes objects from entities, in that objects have the feature of inheritance, which is the basis for reusability. Active objects are those that can act on other (mostly passive) objects. Office facilities and information are captured by the passive objects. At the application level this distinction is important. For instance, the role "Marketing Representative" (active object) approves a "Marketing Letter" (a passive object). The class of active objects in ITAHAC can be divided into three subclasses: actors, roles, and organizational groupings (see Figure A-10, from [Ang/Conrath 1993], p. 8). For the ITHACA authors, roles are the basic elements of an organizational structure. In [Ang/Conrath 1993] (p. 7) a comprehensive explanation why roles are important is given. All three active objects are defined by a set of instance variables and operations. Some are, for instance, name or title, tasks that the active object is to perform, skill capabilities (actor) or requirements (role).

Theoretical model, prototype, or product

In the literature the model is being used to develop software for the ITHACA project. A prototype of the active office object model had been developed then, written in Cool (Combined object-oriented Language). The prototype was later used as a basis for the development of a more complete systems architecture.

Primary modeling focus

The ITHACA office object model is a part of a larger research effort. Its purpose is to provide programmers and software engineers with a generic framework which forms the basis to develop large, distributed office support systems that focus at business processes.

Model entities

While some information about active office objects in ITHACA is expressed as attributes (for example skills, location, tasks), three objects are defined independently.

Actor: Refers to any person, fulfilling a role.

Role: Describes a set of functions or actions that are performed on passive office objects.

Organizational grouping: A combination of organizational roles, not actors.

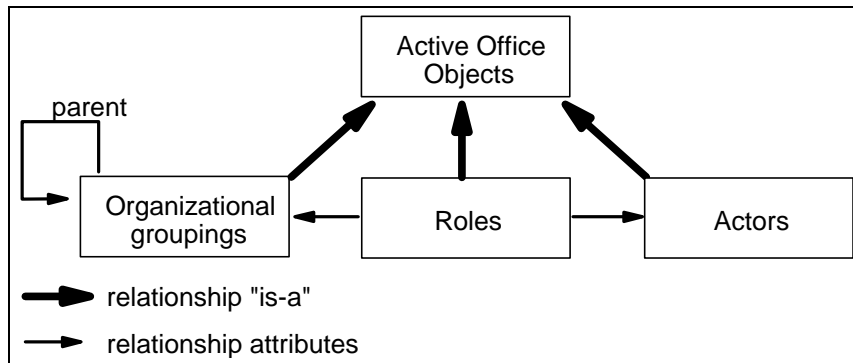


Figure A-10: Conceptual model of the ITHACA office object model

Model relations

Roles are part of organizational groupings and are held by actors. Moreover, an organizational grouping can be parent of an organizational grouping. All three objects are (is-a) active office objects. Furthermore, the office object model allows for the specification of derived attribute relationships (DAs) which can logically be derived from the simple relationships. For example, a derived relationship "managerOfUnit" selects the actor who holds the role "UnitManager" in the "partOf" list of roles of that permanent unit. Similarly, the relation "belongsTo" can be derived from the fact that a role is held by an actor, and that this particular role is part of an organizational grouping. This implies that the actor belongs to the organizational grouping. However, these relations are not displayed in Figure A-10.

Tool support

During the project's lifetime a programming environment was provided where each of the active objects automatically had a record that resided in a permanent database. Whether additional easy-to-use graphical tools have been developed in the meantime is not known.

Authors and references

James S.K. Ang and D.W. Conrath

[Ang/Conrath 1993], [Ang 1993]

A.10 Meta-Model Workflow by Derungs, Vogler, and Österle

Description

The meta-model workflow covers various components of workflow and organizational modeling: the workflow system steers and controls the flow of work, the information system provides necessary data and information, and the desktop system integrates the activities with applications. In addition to these process-oriented aspects, an authorization concept covers the relevant infrastructure questions. This authorization concept (*Berechtigungskonzept*) is this section's focus. Its main purpose is to assign responsibility to tasks in the business processes.

Theoretical model, prototype, or product

The meta-model has been developed parallel to a workflow method at the University St. Gallen as a theoretical, methodological approach.

Primary modeling focus

The meta-model workflow determines and defines necessary basics in the field of workflow management. It creates a clear terminology and specifies the main components of a workflow system (cp. [Derungs/Vogler/Österle 1995], p. 2).

Model entities

Organizational unit: A self-responsible, permanent part of the infrastructure of an enterprise. For example, business units, organizational units, departments, groups, offices.

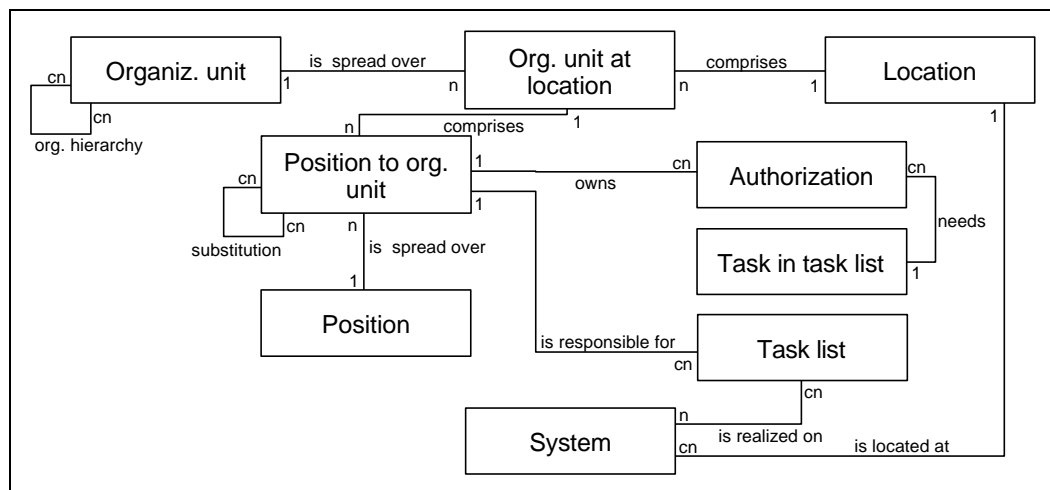


Figure A-11: Conceptual model of Meta-model Workflow

Position: A description of all expectations regarding the employee assigned to this position. It comprises all rights, privileges, duties, and obligations in respect to the enterprise.

Location: A physical, geographical point. Organizational units reside at locations. Examples are land, region, city, suburb.

Authorization: Describes which position is allowed to perform which task. Authorizations of this kind only cover the respective workflow system and no external (network) resources and the like.

System: A platform on which applications, data storage or the performing of tasks is realized.

Task: Compound units of activities to be performed which are controlled and steered by the workflow system. Tasks can be performed in dialog with the user or in the background.

Task list: A definition of tasks and the sequence of their performance in order to reach a given goal.

Model relations

Organizational units reside at geographical locations and they have positions assigned. The authorization for carrying out tasks is assigned to positions and positions may have substitution rules defined between them. The authorization for a position always relates to a task, which in turn is comprised in a task list. A position is responsible for a task list as process administrator. A task list is realized on one or more systems (i.e. computers or application platforms) and each system is based at a specific location as Figure A-11 shows.

Tool support

The meta-model workflow is a method and model only. No tool support exists.

Authors and references

Marc Derungs, Petra Vogler, and Hubert Österle

[Österle 1993], [Österle 1995], [Derungs/Vogler/Österle 1995]

Derungs, Marc; Vogler, Petra; Österle, Hubert: From BPR Models to Workflow Applications, in: Lawrence, P. (Ed.): Workflow Handbook 1997, Wiley, Chichester etc., 1997, pp. 49-59.

A.11 The FUNSOFT Meta-Model by Gruhn et al.

Description

The Leu (LION GmbH Entwicklungsumgebung) approach to workflow management considers data models (describing types of objects to be manipulated in a business process and their relationships), activity models (describing activities to be carried out in a process), and organization models as separate components. The organization models are used to define which organizational entities are involved in a business process. The modeling of the three separate models is based on the FUNSOFT-approach, which is a high level Petri Net approach. Organization models are described by organization diagrams. The relationship between organizational entities, roles, persons, and permissions is defined in tabular form.

Theoretical model, prototype, or product

The FUNSOFT approach for an integrated business process management is a theoretical method. However, based on the FUNSOFT approach the research prototype CORMAN (Coordination Manager), and the commercially available systems Leu and WIS (Wohnungswirtschaftliches Informationssystem) have been developed ([Deiters/Gruhn/Striemer 1995], p. 464).

Primary modeling focus

As the WfMS Leu supports process modeling, process model analysis and process enacting, its main focus is on processes.

Model entities

Role: Roles are sets of permissions for the execution of activities. They are assigned to the organizational entities and they define the sets of permissions and obligations the organizational entities have. The model is based on a hierarchical role concept.

Person: Persons are real-life process participants which are assigned to roles.

Permission: Describes the right a role has to have in order to carry out a certain activity. Permissions on object types define whether or not a process participant can insert, modify, or delete objects of this object type.

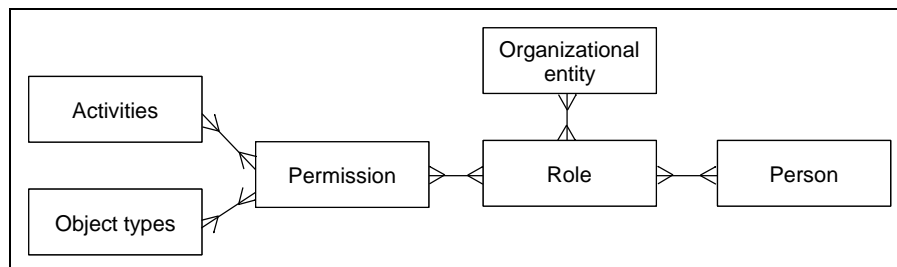


Figure A-12: Conceptual data model of the WfMS Leu

Organizational entity: Defines groupings of process participants and identifies organization diagrams (i.e. hierarchy) in an organization.

Figure A-12 shows the relationships among the organizational entities, roles, persons, and permissions.

Model relations

Roles are assigned to organizational entities, persons play roles, and permissions are contained in roles. Permissions are given for the performance of activities and for modification of object types.

Tool support

According to [Bach/Brecht/Österle 1995] (p. 111), Leu provides a graphical editor for the modeling of the infrastructure model.

Authors and references

W. Emmerich, Volker Gruhn, Guido Dinkhoff, and Rüdiger Striemer

[Deiters/Gruhn/Striemer 1995], [Bach/Brecht/Österle 1995] pp. 111ff.

Dinkhoff, Guido; Gruhn, Volker: Entwicklung Workflow-Management-geeigneter Software-Systeme, in: Vossen, G.; Becker, J. (Hrsg.): Geschäftsprozeßmodellierung und Workflow Management: Modelle, Methoden, Werkzeuge, International Thomson Publishing, Bonn, Albany, 1996, pp. 405-421.

Gruhn, Volker; Kampmann, Martin: Modellierung unternehmensübergreifender Geschäftsprozesse mit FUNSOFT-Netzen, in: Wirtschaftsinformatik, Vol. 38, Nr. 4, 1996, pp. 383-390.

Dinkhoff, Guido; Gruhn, Volker; Saalman, Armin; Zielonka, Michael: Business Process Modeling in the Workflow Management Environment Leu, in: Loucopoulos, P. (Ed.): Entity Relationship Approach - ER'94, Business Modelling and Re-Engineering, Proc. of the 13th Int. Conf. on the Entity Relationship Approach, Manchester, UK, 13-16.12.1994, pp. 46-63.

Gruhn, Volker: Entwicklung von Informationssystemen in der LION-Entwicklungsumgebung, in: Scheschonk, G., Reisig, W. (Eds.): Petri-Netze im Einsatz für Entwurf und Entwicklung von Informationssystemen, Berlin, 1993.

Emmerich, W.; Gruhn, Volker: FUNSOFT Nets, A Petri-Net based Software Process Modeling Language, in: Proceedings of the 6th International Workshop on Software Specification and Design (Como), Washington, 1991.

A.12 Business Model by Reim and Rathgeb

Description

In the course of describing a method for the design of computer-supported business processes, Rathgeb defines a business model for the abstract specification of organizational circumstances ([Rathgeb 1996], pp. 183ff.). The model consists of four distinct, yet connected views: a view that deals with processes and activities forms the base of the modeling, an infrastructure view defines positions and groupings, a view that concentrates on information objects and documents, and a view that specifies elements of the (software) system. The infrastructure model is what this section is concerned with. Relationships between the entities of the four views are modeled by means of the ERM approach. The initial model has been proposed by Reim [1992] and refined by Rathgeb afterwards.

Theoretical model, prototype, or product

Reim has defined the business model as part of the project COMANDOS (Construction and Management of Distributed Open Systems). DISDES (Distributed Information System Designer) is a prototype system which allows for the computer-based modeling of information systems.

Primary modeling focus

As the title "method for the design of computer-supported business processes" suggests, the project mainly focuses on business processes.

Model entities

The organizational entities of the business model are depicted in Figure A-13 and explained in the following.

Position: A position is the smallest organizational entity which can perform activities.

Person: A single subject (employee) who can become active in processes.

Organizational unit: A grouping of positions. The structuring of organizational units forms a hierarchy.

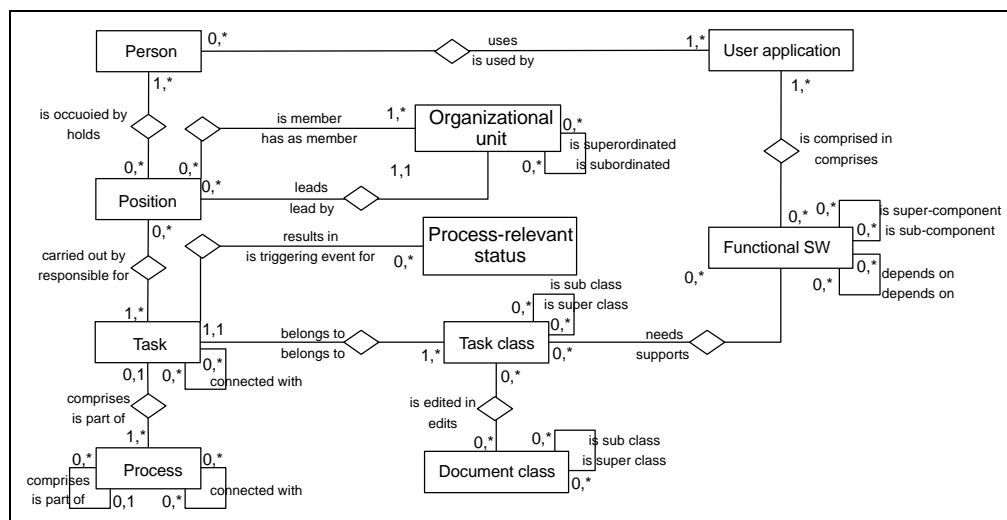


Figure A-13: Conceptual model of Rathgeb and Reim's business model

Model relations

A position can be responsible for various activities and is member of organizational entities. A position can be filled by persons and can be responsible for leading organizational units. A person holds one or more positions. When working, a person may use various application environments. An organizational unit has one or more positions as members. It can be subordinated to other units or it can superordinate other units. An organizational unit is lead by a position.

Tool support

DISDES supports the incremental development of a graphical representation for infrastructure models. Graphical modeling editors support in this task (cp. [Rathgeb 1994], p. 194).

Authors and references

Friedeman Reim and Michael Rathgeb

[Reim 1992], [Rathgeb 1994], [Rathgeb 1996]

A.13 Architecture of Integrated Systems (ARIS) by Scheer**Description**

With the Architecture of Integrated Systems (ARIS), Scheer has developed a concept which covers four necessary views for the modeling and development of workflow management systems: organization, data, functions, and steering. The relationship between the first three views is gained by means of the steering view, however, the organizational view is important here. Galler [1995] has examined meta-models of workflow management and has taken ARIS as the exemplary basis for his description. Minor variations may exist between the ARIS model and Galler's meta-model for workflow management as it is described here.

Theoretical model, prototype, or product

The meta-model described here has been developed as a general architectural base and has found an implementation in the ARIS-Toolset which is the topic of section B.4.1.

Primary modeling focus

The ARIS meta-model claims to cover all four views equivalently sound. Hence, no primary focus can be identified for one of the four modeling domains.

Model entities

Galler's original conceptual meta-model shown in Figure A-14 (see [Galler 1995], Figure 20) knows various organizational entities which are briefly explained in the following:

Organizational grouping: The organizational grouping is the central entity of the model. It can be an actor, a position, or an organizational unit. It represents the hierarchical aspect of an organization.

Actor: Everybody who is internally or externally concerned with the process enactment.

Position: The position is the elementary entity of an organizational structure. It contains a defined range of functions and is assigned to exactly one actor. Positions are distinguished into main positions, and substitution positions and the substitutions may be dependent on a particular period of time.

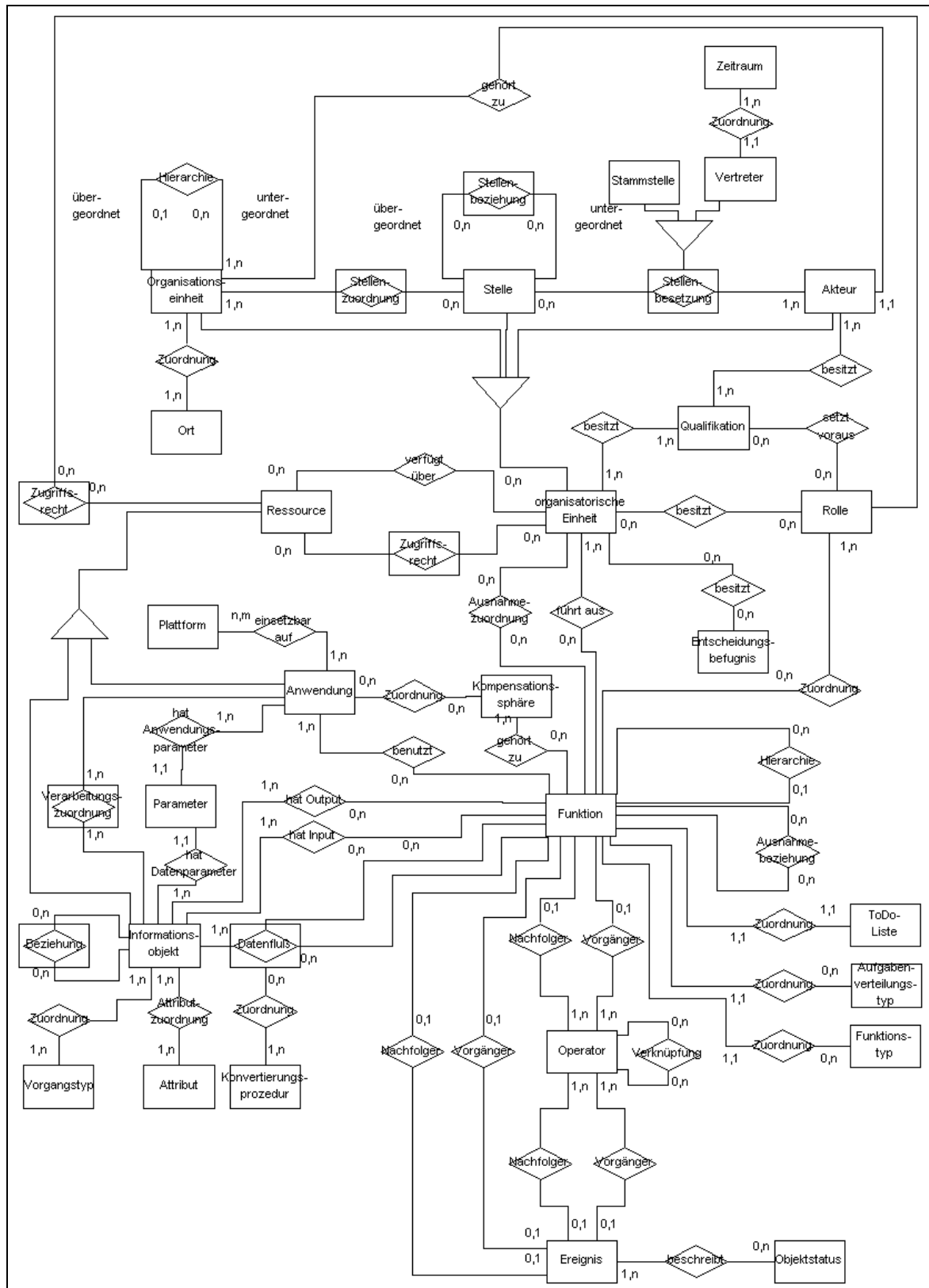


Figure A-14: Conceptual model of ARIS as presented by Galler

Organizational unit: Organizational units are combinations of positions into larger units, such as line units or project groups.

Location: The location of an organizational unit defines its geographical residence.

Role: A role is an aggregation of positions.

Function: Defines activities which are controlled and steered by the WfMS.

Resource: Resources are means to perform activities in the process enactment.

Model relations

The relations and cardinalities of the meta-model are not further explained by Galler. The paper considers their semantics as being intuitive and refers to the graphical representation, as reproduced in Figure A-14.

Tool support

The modeling of the organizational view is supported by graphical tools in the ARIS-Toolset (see section B.4.1).

Authors and references

August-Wilhelm Scheer and Jürgen Galler

[Galler 1995], [Scheer 1992], [Scheer 1995], [Scheer/Nüttgens/Zimmermann 1995]

A.14 ROM by Esswein

Description

The object model ROM (Rollenmodell der Organisation) presented by Esswein [1993] supports the specification of problem domain objects. Esswein's model focuses solely on roles played by the problem domain objects. It is a model for the comprehensive representation and documentation of an organization's structural model. The ROM is based on the ORM, the Object-Role Model, which specifies that objects (in other words, persons in an organization) can hold different roles at different points of time (cp. [Esswein 1993], p. 555). ORM is a strictly hierarchical object model for organizational roles. *Hierarchical* means that roles which are defined at the root-level are later refined into more specific role descriptions at lower levels. ROM and ORM present a very comprehensive organizational role model which is much more detailed than the role definitions of other models. Consequently, it neglects other entities which are only touched in the description by Esswein.

Theoretical model, prototype, or product

ROM is a theoretical model and does not directly refer to workflow management, but to general organizational design.

Primary modeling focus

The ROM strives for an integrated description and documentation of an organization's structural reality. It thus focuses on the organization model.

Model entities

Role type: A role type is a combination of task types which will be assigned to the same actor in a process definition.

Positions: Positions are described by a combination of role types and tasks which describe the skill requirements of the person that holds the position.

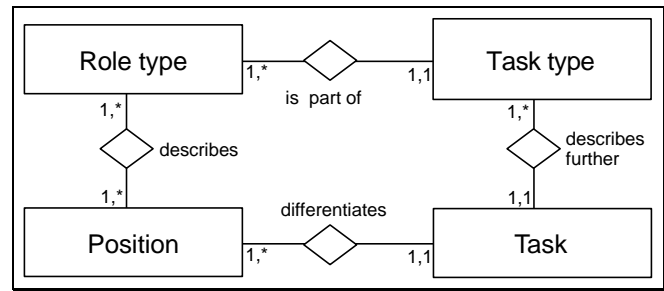


Figure A-15: Conceptual model of ROM

Task type: A task type generally describes a set of tasks that may be performed by a role.

Task: A concrete task that makes up a task type when combined with other tasks.

Model relations

Role types describe a position and a position differentiates tasks. A task type describes the tasks in detail and is part of a relation type. Figure A-15 shows the ROM.

Tool support

Since ROM is a theoretical model, no (graphical) tools support exists.

Authors and references

Werner Esswein

[Esswein 1993], [Galler 1995]

Esswein, Werner: Dynamische Objektbeschreibung durch Rollen, in: Gaul, W.; Bachem, A.; Habenicht, W.; Runge, W.; Stahl, W.W. (Eds.): Proceedings Jahrestagung der DEGOR 1991, Berlin, 1992, pp. 682-689.

A.15 The SAP Business Workflow Organizational Model

Description

The SAP R/3 system is a widespread standard software for business applications. Since its release 3.0, the R/3 basic component is equipped with an integrated set of workflow tools. These tools conceptually rely on an organizational data and process model, which describe the functionality and the structure of the associated applications. The organizational view, which exists next to a process view and an object view, is the focus here. It defines the organizational model for the SAP Business Workflow product.

The organizational model specifies all organizational entities that may be involved in a business process: (permanent) positions, their combination into organizational groupings (workgroups, units), persons, and tasks. The central entity is the task, which describes things

that have to be performed in order to reach a given goal and which bridge the gap between process and organizational model (generally the positions).

Theoretical model, prototype, or product

The organizational model has been implemented in the product SAP Business Workflow.

Primary modeling focus

Although the organization model stresses the infrastructure aspect, the product SAP Business Workflow as such mainly concentrates on the business process aspect of an enterprise.

Model entities

Task: A description of something to be performed in order to reach a given goal. A task's attributes describe the event which triggers and ends the task performance. Additional comments on the task describe what has to be done.

(Permanent) position: In the available literature no explicit statement about positions has been made. However, it implicitly defines a position as a combination of similar functions that may be performed by one person who holds the position.

Person: A human being, i.e. an employee or an external partner.

Organizational grouping: Any grouping of human beings (employees) in the organization, such as organizational units, project teams or workgroups.

Role: The available literature has made no explicit statement about roles. Roles appear to have been added to the model very recently. The company brochure displays roles, but does not explain this entity's meaning.

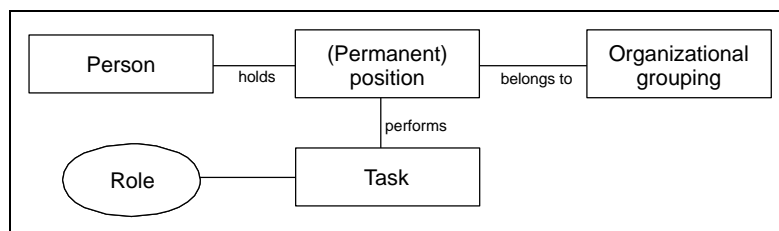


Figure A-16: Conceptual model of SAP Business Workflow

Model relations

Due to a shortage of detailed information on the SAP Business Workflow *model*, Figure A-16, which shows the conceptual model, has little information about relationships and cardinalities.

Tool support

The workflow tools of SAP Business Workflow have various software components for modeling, administrating, and managing infrastructure and organization models.

Authors and references

SAP AG, Walldorf

Wächter, Helmut; Fritz, Franz J.; Berthold, Andreas; Drittler, Bernhard; Eckert, Harald; Gerstner, Ralf; Götzinger, Ralf; Kraus, Ralf; Schaeff, Andreas; Schlögel, Christian; Weber, Rainer: Modellierung und Ausführung flexibler Geschäftsprozesse mit SAP Business Workflow 3.0, in: Huber-Wäschle, F.; Schauer, H.; Widmayer, P. (Hrsg.): Proceedings GISI-Jahrestagung 1995, Herausforderungen eines globalen Informationsverbundes für die Informatik, Zürich, September, 1995.

SAP AG: SAP Business Workflow, Ihr Erfolgsrezept: Flexible evolutionäre Geschäftsprozesse, Walldorf, 1996.

SAP AG: Organization à la carte, SAP INFO, Walldorf, 56, 4, 1998, pp. 54-57.

SAP AG: Structure Modeler, Structure modelling as success factor, SAP INFO D&T, Walldorf, 57, 6, 1998, pp. 28-29.

A.16 Office Model One (OM-1) by Ishii et al.**Description**

COOKBOOK (CoOperative Office worK Based On Office Knowledge) is a project which integrates office modeling and office procedure automation centered around an office procedure knowledge base. Ishii presents the semantic data model Office Model One (OM-1) to support organizational office work throughout the design, guidance, planning, and execution stages. The OM-1 is intended to give a framework for office knowledge representation and to serve as the starting point for a knowledge base. OM-1 specifies office procedures, activities, documents, files, roles, and agents. It represents the structure and relations of these objects using a strictly hierarchical layout shown in Figure A-17, and it provides an iconic symbol for each object and a graph representation.

Theoretical model, prototype, or product

Besides the theoretical framework of OM-1, Ishii presents a prototype knowledge base system with graphical user interface BOOK (Browser Of Office Knowledge Base), however, a product has not been developed.

Primary modeling focus

COOKBOOK mainly deals with office procedure automation and concentrates on an office procedure knowledge base.

Model entities

Office procedures: Tasks are represented as office procedures. They represent control structures which are depicted by directed activity graphs, responsible agents, and accessed data.

Activities: An activity is a task unit executed by agents. It is a component of an office procedure. To represent less-structured tasks (office procedures), a set of activities without order relation can also be accepted as an office procedure.

Agent: Abstract office workers or organizational units, such as sections or departments, that play specific *roles* in executing activities.

Data: Data is a document or a file.

Document: A document is an object composed of free text, structured fields, tables, graphs, and so on. A document is modeled as a temporary medium to carry information for the execution of a task. Documents may be forms, memos, or mails.

File: A file is used as a repository to store and retrieve documents.

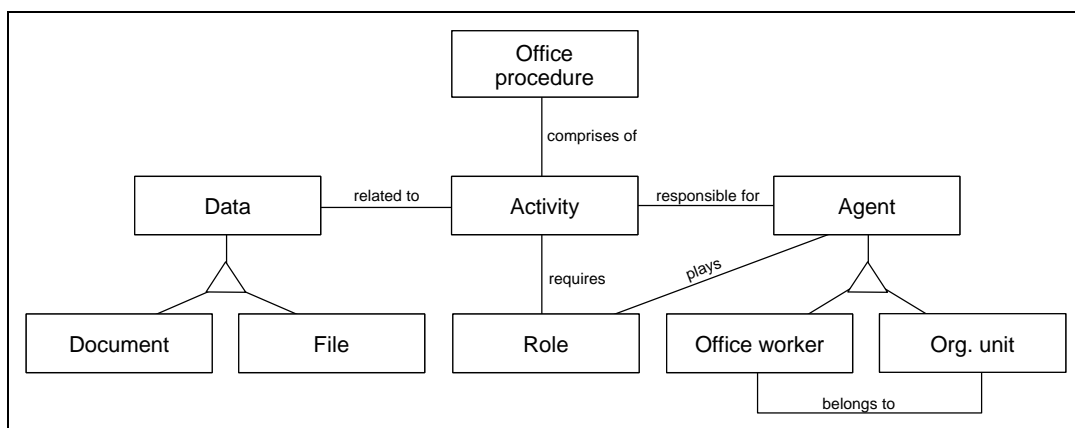


Figure A-17: Conceptual model of OM-1

Model relations

Agents usually play a number of different roles, and the roles specify which activities the agents are responsible for. Documents and files are related to activities as input or output information of the activities. Office workers may belong to an organizational unit and may have a role assigned.

Tool support

The project developed BOOK, which is a kind of visual semantic net editor with general graph and tree editing features.

Authors and references

Hiroshi Ishii, Kazunari Kubota, and Masaaki Ohkubo

Part of the work on OM-1 was carried out by Ishii during his involvement at the Gesellschaft für Mathematik und Datenverarbeitung mbH (GMD) in 1986/87 (cp. [Ishii/Ohkubo 1991]). Hence, members of the GMD team, such as Thomas Kreifelts and Frank von Martial, have also contributed to the OM-1 development.

[Ishii/Kubota 1989], [Ishii/Ohkubo 1991]

A.17 The OGM Meta-Model for Business Process Modeling by Rohloff

Description

Rohloff introduces an object oriented approach to business process modeling. His approach integrates the organizational design with the systems development. The object oriented meta-model in Figure A-18 shows the object classes relevant for modeling business processes. The model's object classes are divided into three groups, which address: (1) business objectives, tasks, and performance measures, (2) process input and resources allocated to processes, and (3) personnel engaged with the performance of the process and its tasks. The third group's entities process owner, performer, and role are of further interest here. According to Rohloff, a fourth group of object classes addresses the corresponding organizational structure ([Rohloff 1996], p. 254), which has not been explained in the available literature on the meta-model for business process modeling.

Theoretical model, prototype, or product

The OGM (Objektorientierte GeschäftsprozeßModellierung) approach outlined by Rohloff is based on OMT as a widely used object oriented modeling technique. OGM only theoretically demonstrates how object oriented modeling techniques can be extended in order to address business process design.

Primary modeling focus

"The meta model is the basis for the description of business processes and their performance within a business organisation." "Primary focus is on the core entities process/process step, event, and state in order to describe the business flow .." ([Rohloff 1996], p. 254).

Model entities

Process owner: Any personnel involved in the performance of the process that is responsible for the process and its performance.

Performer: For example employees who are involved in the performance of the process as a customer or supplier.

Role: A set of functions assigned to a performer in a specific process.

Model relations

A process owner manages a process and performs the function of a role. A performer participates in a process. A performer may be a customer or a supplier and executes one or more roles. Performers trigger events which are released by processes. Tasks are allocated to roles. For further relations, which are not directly associated with the organizational entities, refer to Figure A-18.

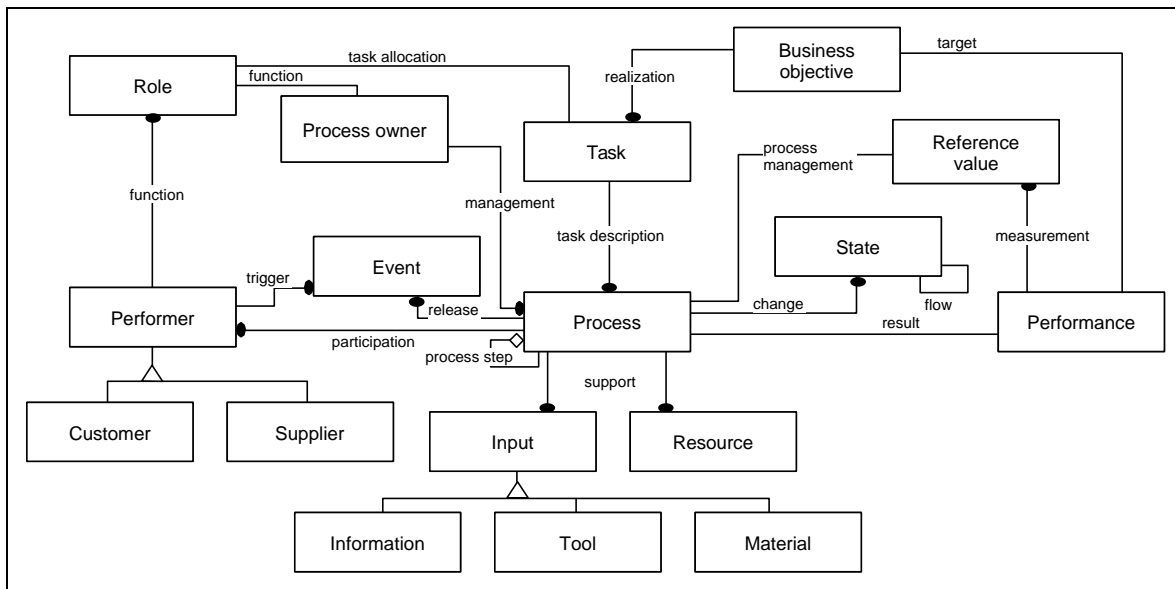


Figure A-18: Conceptual model by Rohloff

Tool support

As OGM is a theoretical enhancement of OMT modeling, no tool support exists yet.

Authors and references

Michael Rohloff

Rohloff, Michael: An object oriented approach to business process modelling, in: Scholz-Reiter, B.; Stickel, E. (Eds.): Business Process Modelling, Springer Verlag, Berlin, 1996, pp. 251-264.

Rohloff, Michael: Objektorientierte Modellierung betrieblicher Abläufe im OGM Ansatz, in: EMISA-Forum, Mitteilungen der GI-Fachgruppe "Entwicklungsmethoden für Informationssysteme und deren Anwendung, 1, 1996, pp. 100-110.

A.18 The IBM-FlowMark Organization Model

Description

The WfMS FlowMark by IBM consists of a workflow server, a built-time, and a run-time component. The built-time component is the application used for the (graphical) description of business circumstances with a focus on business processes. FlowMark uses PM-graphs

(Process Model graphs) with the main characteristic, that the data flow and the control flow is designed separate from each other. Together with the structural information examined here, these graphs serve as the base for the run-time component.

Theoretical model, prototype, or product

FlowMark is a commercially available WfMS by IBM.

Primary modeling focus

The center of FlowMark (and thus its conceptual data model) is the (graphical) modeling of task-sequences and their support through a WfMS.

Model entities

Figure A-19 shows the entities and relations of FlowMark's organization model.

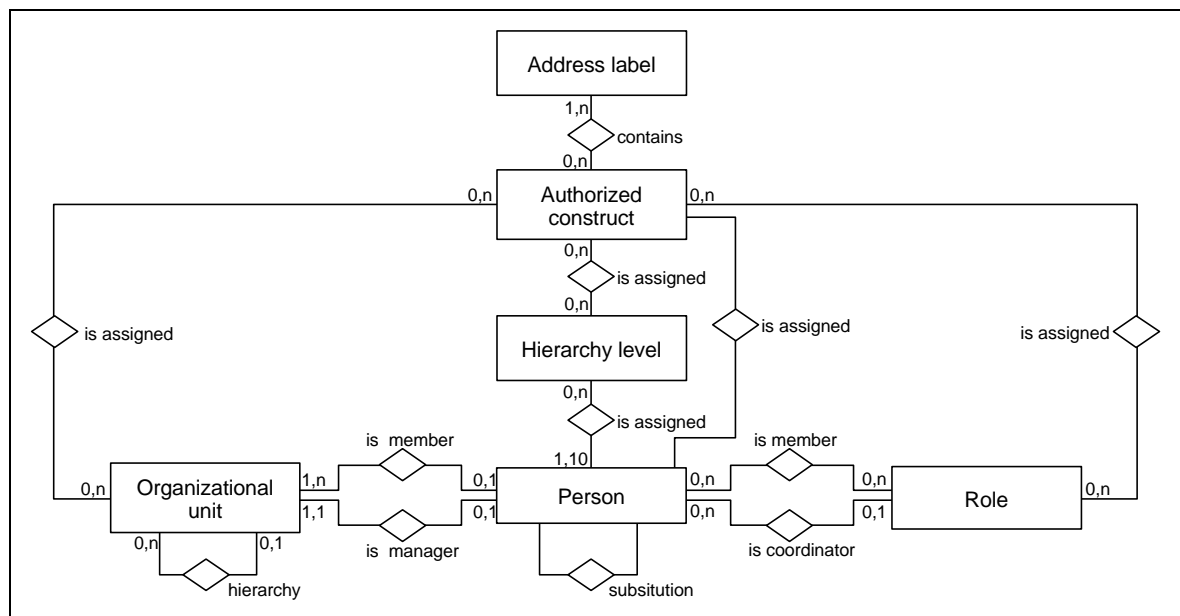


Figure A-19: Conceptual model of IBM-FlowMark

Person: Employees of an organization and external partners.

Role: Describes the competencies a person has in order to carry out tasks.

Organizational unit: Groupings of persons in form of departments, sections, workgroups, etc.

Authorized construct: An authorized construct restricts the number of possible performers according to criteria defined which specify (narrow down) particular roles, persons, or units.

Address label: Interface between organization and process model which abstractly defines actors allowed to perform a particular task.

Model relations

A person is assigned to none or exactly one organizational unit and an organizational unit has at least one or many persons as members. Additionally, one organizational unit has one person

assigned as manager and a person may at most be assigned manager to one unit. Many organizational units may be subordinated to one unit, and only one unit may be superordinated to another. One person may have none or another person as substitute. A person is member of none or many roles and a role has none or many members. Moreover, a person may be coordinator of none or many roles, however, one role may at most be coordinated by one person. Persons are assigned to one or at most ten hierarchy levels.

Tool support

The FlowMark modeling environment provides with a graphical design environment for the process and the organizational structure definition.

Authors and references

Frank Leymann et al.

Frank Leymann is not the only developer of the business model behind IBM's FlowMark, however, he has published various reports and papers on it which is why his name is listed as the main resource here.

[Leymann 1997], [Bach/Brecht/Österle 1995] pp. 94ff.

Leymann, Frank; Roller, Dieter: Business process management with FlowMark, in: Proc. IEEE COMPCON Spring 1994, IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 230-234.

Leymann, Frank; Altenhuber, W.: Managing Business Processes as an Information Resource, in: IBM Systems Journal, 33, 2, 1994, pp. 326-348.

Leymann, Frank: Transaktionskonzepte für Workflow-Management-Systeme, in: Vossen, G.; Becker, J. (Eds.): Geschäftsprozeßmodellierung und Workflow Management: Modelle, Methoden, Werkzeuge, International Thomson Publishing, Bonn, Albany, 1996, pp. 335-352.

A.19 The Model of TOSCA by Prinz

Description

Prinz [1996] describes TOSCA, an implementation of an organizational information system. TOSCA is both, a framework and system for organizational information support in cooperative environments. The kernel of its conceptual framework is a meta-object model that defines the construction rules for a specific organization object model. Prinz first describes three building blocks of an office object model: organization objects, relationship objects, and event objects. The aim of the proposed schema is the provision of a toolkit that can be adopted to different organizational settings. TOSCA's *structure* domain (cp. [Prinz 1996], p. 114), which describes the organizational entities and relations, and its *people* domain, which maps the job or role descriptions to the members of the organization (i.e. the people), are of major

importance here. Moreover, the *location* domain and the *resource* domain contain organizational entities which are considered in this regard. The *procedure* domain focuses at workflow aspects. Based on these considerations, Prinz proposes an organization object class hierarchy similar to the model shown in Figure A-20.

Theoretical model, prototype, or product

TOSCA is a conceptual framework and a prototype system. Its practical applicability has been demonstrated by its application for a partial modeling of the German government during the POLIKOM demonstration.

Primary modeling focus

TOSCA's very comprehensive model focuses on six organizational domains of which five are concerned with non-procedural matters: structure, people, location, and resource, plus the role-related entities. Hence, its focus is strongly on organizational and structural information and less on process aspects.

Model entities

Prinz's model is an object oriented model and his descriptions use terminology from the object oriented domain. For reasons of comparability the objects of the TOSCA model have been expressed here in form of entities and relations. TOSCA's class hierarchy distinguishes between six different object classes: structure, procedure, role, person, resources, and locality. Each class is regarded as the root class of a subtree which contains object classes that model specific domains. However, "all root classes are abstract classes, i.e. classes which should not be instantiated" ([Prinz 1996], p. 121). Due to this, only object classes from the six subtrees are listed as entities of their own in the following:

Organization: Represents the basic properties of an enterprise and is the root entity for an organizational structure. Exists only once in a concrete organization model.

Task unit: Task or goal oriented units that group employees who fulfill a common task, such as a project group.

OrgUnit: Organizational units are applied for the basic structuring of an organization, such as departments or institutes.

Organizational role: Defines roles in the context of the organizational function. These are roles which fulfill a function in respect to a particular project, department, or committee.

Procedural role: Roles which are used in a procedural context in that they only exist in the context of the procedure description. The period of the binding or the role player to such a role depends on the lifetime of the procedure.

Employee: Represents the members of an organization. In Prinz's model this class may be subtyped into administrative or production oriented employees to describe different job types.

External resource: External resources refer to external data.

Device: Refers to any sort of computer-based application.

Software: Any program that can be run on a device.

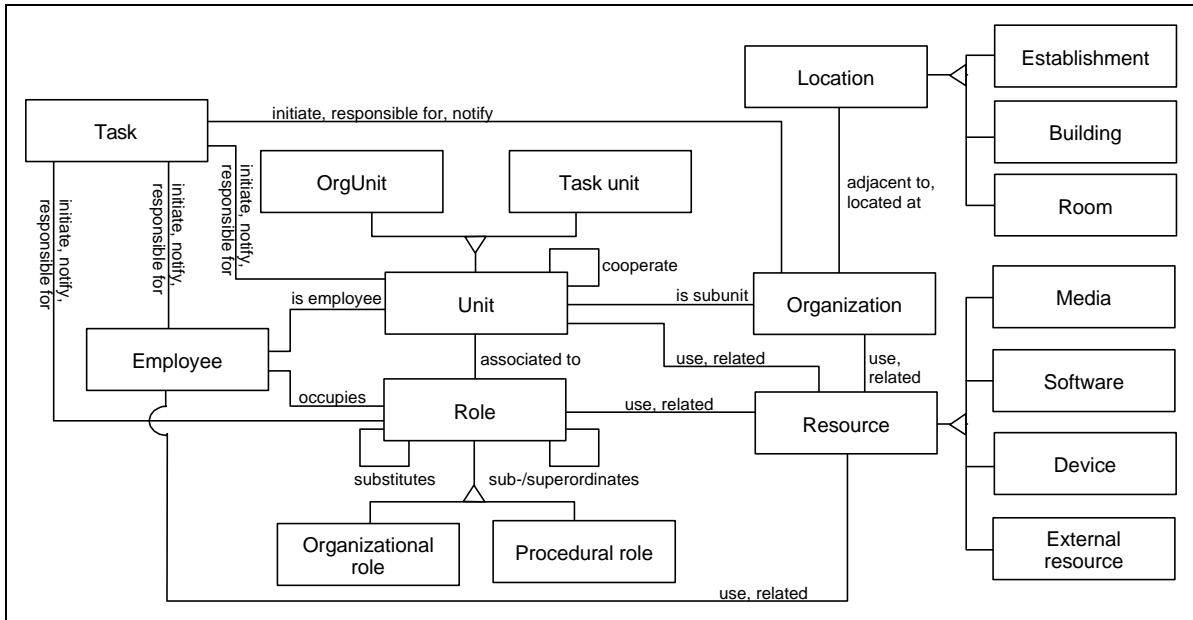


Figure A-20: Conceptual model of TOSCA

Building: House or place, i.e. part of the description to model geographical and spatial distribution of an organization.

Establishment: More coarse part of the description to model distribution of an organization.

Room: Smallest part to model geographical and spatial distribution.

Task: The basic unit of work. Tasks represent the interface to the procedure domain and its applications.

Model relations

Similar to the organizational entities, TOSCA defines four general relationship classes: structural, procedural, resource, and spatial. Again, only organizational relationship classes of the respective subtrees may be instantiated and are discussed here (cp. [Prinz 1996], p. 129).

Structural relationships may be instantiated from Task Units and OrgUnits to Organization as *subunit* relations, for instance in form of department, office, committee, institute, or project. In other words, an OrgUnit may be related to an Organization as a department, as a project, and so on. Two such organizational units may also *cooperate* when they have no hierarchical, but a cooperative relationship. Role relationships are distinguished into three different relationships. A role may be *superior* or *subordinated* to another, it may *substitute* another role, and it may be *associated to* an organizational unit. A role may also provide a *service* to

an organizational unit. The *occupy* relationship associates an employee to a role and the *employee* relationship associates an employee with an organizational unit.

Organizational entities may *initiate* a task, they may be *responsible for* a task, and they may be *notified* about a task.

Resource relationships associate resources with any other organization entity in form of *use*, *part of*, and *related* relations.

Spatial relationships allow the description of the spatial distribution of an organization to locations, such as *adjacent to* and *located at*.

Tool support

Part of the TOSCA system is an organizational information browser that provides a graphical user interface to the organizational information server.

Authors and references

Wolfgang Prinz, Uta Pankoke-Babatz, et al.

[Prinz 1993], [Prinz 1996], [Prinz/Kolvenbach 1996]

Pankoke-Babatz, Uta; Prinz, Wolfgang; Syri, Anja: Die Organisationswissenbasis TOSCA, in: Klöckner, K. (Hrsg.): Groupware-Einsatz in Organisationen, "Personal Computing", Symposium Marburg 14.15.10.1993, GMD-Studien Nr. 220, GMD, Sankt Augustin, September, 1993, pp. 117-131.

Klöckner, K.; Mambrey, P.; Solenkamp, M.; Prinz, W.; Fuchs, L.; Kolvenbach, S.; Pankoke-Babatz, U.; Syri, A.: POLITeam - Bridging the Gap between Bonn and Berlin for and with the Users, in: Proceedings of 4th European Conference on Computer Supported Cooperative Work, Kluwer Academic Publishers, Dordrecht, 1995, pp. 17-32.

POLIKOM: Telekooperation - POLIKOM,

<http://www.ba.dlr.de/md/it/iv/tk/polikom/index.html>, 1997.

POLITeam: Entwicklung von Kooperationswerkzeugen zur Unterstützung der Regierungsfunktionen in Berlin und Bonn,

<http://orgwis.gmd.de/projects/POLITeam/POLIKOM/politeam.html>, 1997.

Chapter B

Evaluation of Tools for Organization Design

During the GroupOrga project's lifetime, various tools and application environments for organizational design have been tested and evaluated against the criteria set up in chapters 2 and 3 of the GroupOrga report. Most of the applications have been tested by means of trial software, as demonstration versions, or in form of their evaluation at fairs or during visits to partner organizations or resellers. Others, in turn, could only be assessed by means of information brochures, through telephone interviews, or by similar preliminary tests and evaluations.

As not all of these examined applications have exclusively been mentioned in the main report on GroupOrga, this chapter briefly introduces most of the applications that have been tested. A list of vendors of BPR tools that contain an organizational modeling module is provided in section B.1. Section B.2 then takes a closer look at some of the minor tools which have been tested and evaluated. This chapter proceeds to focus on one specific GroupOrga subproject in greater detail which has examined four major BPR applications in terms of their organizational modeling capabilities in sections B.3 and B.4. To conclude, section B.5 shows the results of a benefit analysis which compares the four examined applications.

B.1 List of Vendors of BPR Applications

The following list of vendors of BPR applications is an incomplete list that has been compiled during the course of the GroupOrga project. Most of the vendors have been contacted for demonstration or trial versions of their software. Those products that offered complex support of organizational design have been tested more thoroughly as shown in section B.2. Besides own results, the evaluation has drawn on results from [Bach/Brecht/Österle 1995], [Tiemeyer/Chrobok 1996], [Tiemeyer/Chrobok 1997], [Lehner et al. 1991], and [Kirn 1995].

Product name	Product vendor
AENEIS	ipro Tool GmbH, Hebrühlstraße 21 B, 70565 Stuttgart
ALF Ablauf	ALF Gesellschaft für Softwareentwicklung und Vertrieb GmbH, Liebigstr. 23, 74207 Leingarten
ALF ORGA	ALF Gesellschaft für Softwareentwicklung und Vertrieb GmbH, Liebigstr. 23, 74207 Leingarten
ARIS-Toolset	IDS Prof. Scheer GmbH, Postfach 10 15 34, 66015 Saarbrücken
Aufbau Profi	ibo Software, Sandusweg 3, 35435 Wettengel
BONAPART	UBIS Unternehmensberatung für integrierte Systeme GmbH, Alt Moabit 98, 10559 Berlin
CAIPLAN-process	Von der Wense & Partner GmbH, Ohestr. 24, 38162 Destedt
Chartist	Novagraph Inc., PO Box 850115, Richardson TX 75085-0115, USA
MetaDesign/IDEF	C.I.T. GmbH Communication and Information Technology, Ackerrstr. 71-76, 13355 Berlin
ICESoft	Bremerhavener Institut für Organisation und Software (BRIG), Stresemannstr. 4, 27570 Bremerhaven
INCOME Mobile	PROMATIS Informatik, Descostr. 10, 76307 Karlsbad
Mosaik-PU (Prozeßuntersuchung)	Sietec Consulting GmbH, St. Martin-Str. 53, 81514 München
Nautilus	integralSA GmbH, Willy-Brandt-Platz 2, 33602 Bielefeld
Orgline	ALLDATA SDV GmbH, Thomas-Dehler-Str. 9, 81737 München
OrgSolution	IOT Software & Medien GmbH, Ollenhauser Str. 23, 13403 Berlin
PRISMA-Tool	Computer Science Research Center (FZI), Haid-und-Neu-Str. 10-14, 76131 Karlsruhe
Process Charter	Scitor GmbH, Platter Str. 79, 65232 Taunusstein
ProzeßMonitor	Simma & Partner, Marktplatz 9, A-6850 Dornbirn
SDW-Tools	SDW Software GmbH, Kattegategweg 7, 46446 Emmerich
StructWare	obus GmbH, Carl-Zeiss-Ring 14, 85737 Ismaning
System Architect	Management Informations Systeme M.I.S. GmbH, Landwehrstr. 50, 64293 Darmstadt
Vamos-BE	innovative software technologie GmbH, Eschenstr. 22, 82024 Taufkirchen
VISIO	VISIO GmbH, Boschetsrieder Straße 67, 81379 München

Table B-1: Product vendors of BPR tools

B.2 Overview of the Results of a Product Evaluation

The overview is given according to a classification that has been introduced in section 3.5.1:

- ❑ tools for mere presentation
- ❑ tools for design and analysis
- ❑ tools for modeling and optimization
- ❑ complex tools with integrated process functionality

This section will briefly introduce selected examples of those products mentioned above that were available as software releases during the testing.

B.2.1 Tools for Drawing and Presentation Purposes

The simplest form of computer support in the organizational design process is provided with software that allows for graphical drawings of organizational circumstances. An important, first step to organizational design is to represent the organizational structures, as it helps to describe the facts of the organizational situation.

VISIO

VISIO has been tested against other more complex applications in a comprehensive product evaluation and will thus be the topic of sections B.3 and B.4.

Chartist 1.7

Chartist is a Microsoft Windows 3.1+ application. Chartist can create, edit, and print flow charts, organization charts, or other charts that use similar components. Through the use of the Windows clipboard, all or part of the charts may be pasted into other documents, providing that the application can read clipboard bitmaps or metafile pictures.

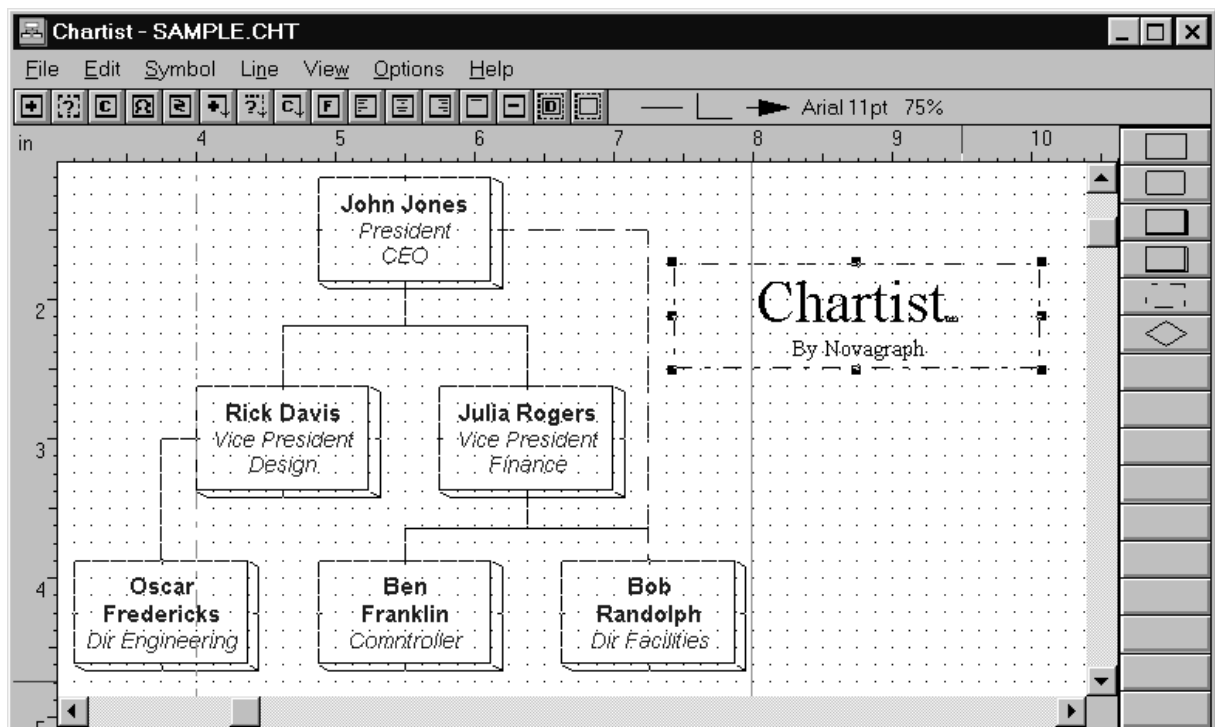


Figure B-1: Sample organization drawn with Chartist

Chartist supports the OLE feature of Windows, allowing other applications to link or embed Chartist information. As with most Windows applications, multiple copies of Chartist may be run on a computer, with each copy simultaneously operating on a different document. Data may be exchanged among copies of Chartist, using the Clipboard.

A Chartist document is one or more pages in size. The orientation of the pages depends upon the settings of the printer used. Within a Chartist document there may be one or more symbols. A symbol is a polygonal shape with optional text or graphics inside it.

Lines in various shapes and with three different routing styles may be drawn between symbols. When symbols are moved around the document, any associated lines are re-routed and re-drawn as necessary. Lines may also be labeled with text.

The goal of Chartist is to provide an easy method for the quick creation of charts, while taking advantage of the advanced features of a printer, and providing data interchange with other Windows applications. The program is mainly aiming at printing organizational or other charts.

B.2.2 Tools for Design and Subsequent Analysis

A further group of programs concentrates on the careful coverage, representation, and analysis of current organizational circumstances. These tools allow to assign further criteria such as costs and time which helps to create or improve future processes and structures. Exemplary products include the following:

Aufbau-Profi

Aufbau-Profi for Windows is a solution for the management of position plans with position descriptions in an organization, for the generation of organizational and functional charts, and for further forms of documentation of structural organization. According to the vendor's documentation, Aufbau-Profi supports the certification of ISO-9000 conform documentation.

Various features are available:

Catalogue

- ❑ Analysis of tasks in an organization
- ❑ Diagrams of functions in an organization
- ❑ Matrices

Organizational charts

- ❑ can be created from the description of organizational units, positions, and persons
- ❑ can be freely formatted in terms of size, color, content, etc.
- ❑ are flexible in the positioning of entities within chart
- ❑ support selected forms of organizational charts

The management of positions takes place in the form of automatic generation of position definitions from organizational data collected. Freely definable lists, such as telephone directories, position listings, and agents can be generated, printed, and exported. The

Aufbau-Profi data is stored in a database. Several databases can be created and managed independently with multiple-user access via passwords and user-profiles. The product can be used as a stand-alone application or on the network. Organizational data can be shared with other ibo Software products, but not with external programs. It is however possible to export graphics (bitmaps) into other Windows programs, and OLE connectivity (restricted to the graphics exported) is provided.

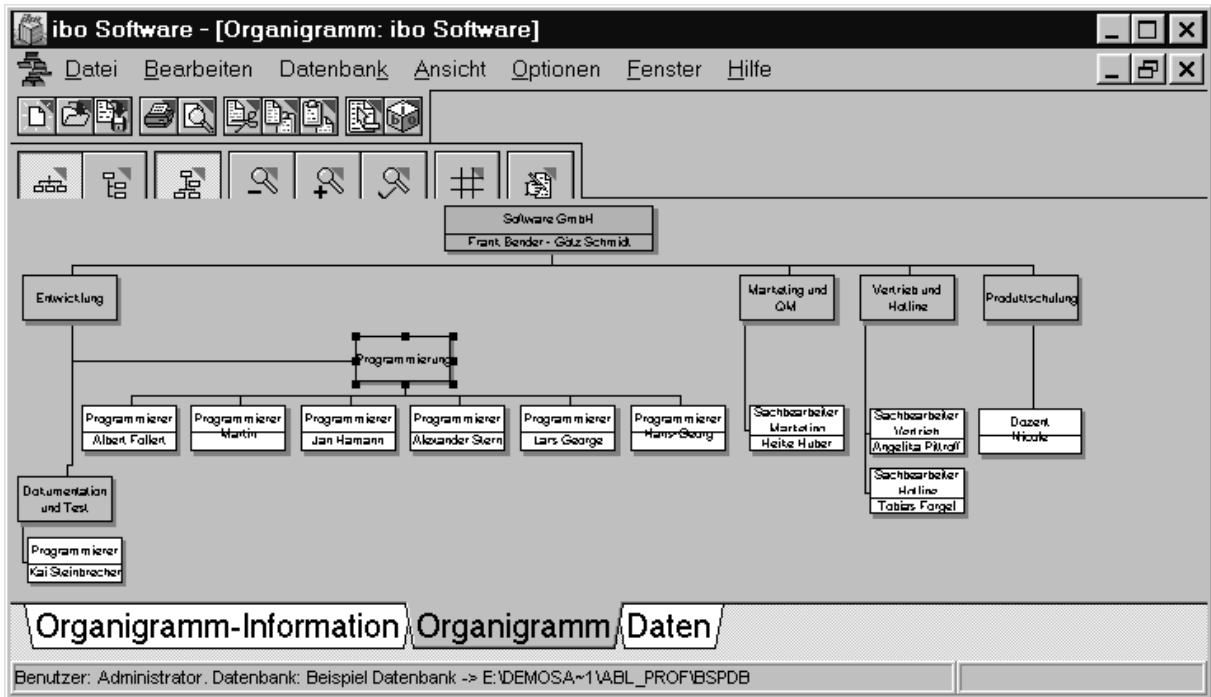


Figure B-2: Sample organization drawn with Aufbau-Profi

ProAS\Doc

ProAS\Doc is an add-on to the product ProAS\Process. ProAS\Process records and analyzes an organization's processes and thus mainly concentrates on the procedural aspects of an organization. Its only reference to the structural organization of a firm is that of a model of the positions in an organization. Such a position model is arranged hierarchically which brings about the notion of units which are subordinated in hierarchies. ProAS\Doc in turn imports the collected data on positions and provides the means to specify the positions in greater detail. The positions can be rearranged in an organizational handbook ("Unternehmenshandbuch") in terms of superordination, subordination, change, and deletion of positions.

The coverage of data on organizational structure takes place in the form of tabular listings and forms that have to be filled in. A simple graphical representation that reminds one of a directory browsing tool depicts the organizational structure and allows for the expansion and collapse of the structural *tree* (see Figure B-3).

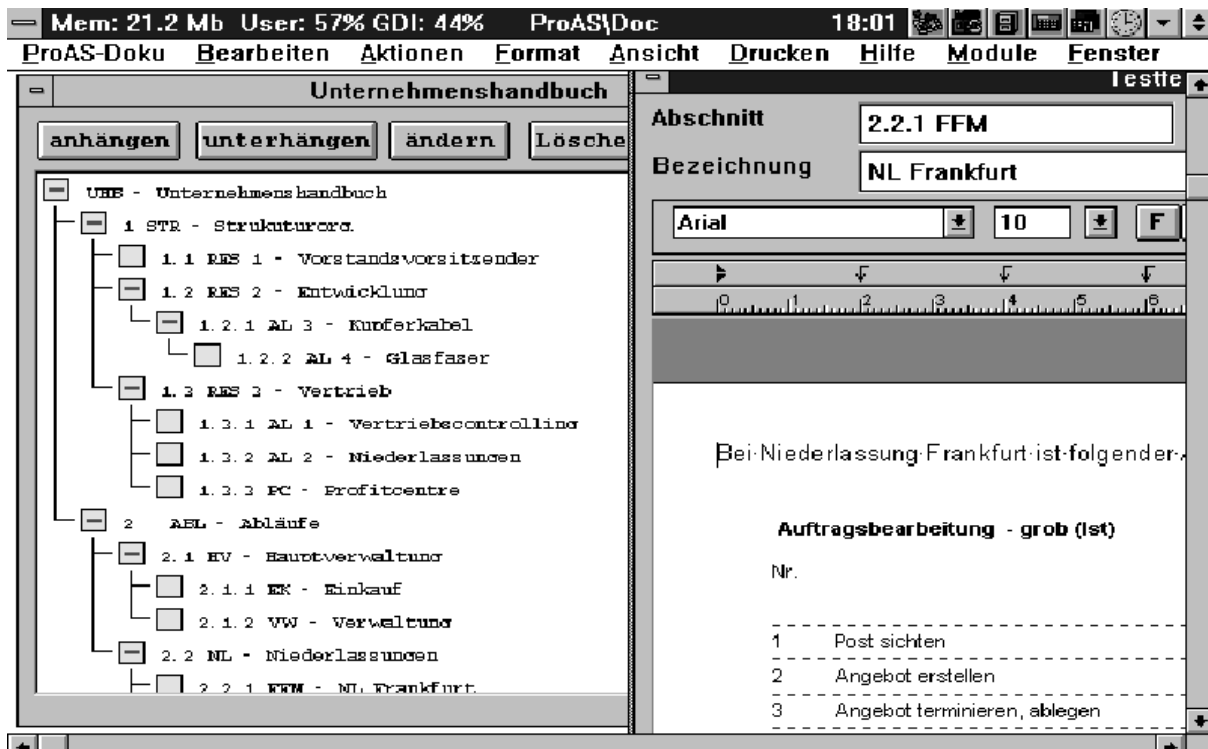


Figure B-3: Sample organization modeled with ProAS\Doc

B.2.3 Tools for Modeling and Optimization

Besides those products which document and analyze, there are tools offering capabilities for process and structure optimization. When searching for an applicable structural design, simulation can often provide various alternatives from which to choose. Products or tools having these features fall into this category.

MetaDesign/IDEF

MetaDesign was founded during its formative stages on theoretical foundations from the GMD and their system "Design". The product was then created by Meta Software Corporation in Cambridge, MA, USA.

It is a general object-oriented graphics and text editor which allows for the graphical representation of concepts, categories, databases, models, and systems requirements. Hence, MetaDesign is not a marked BPR or organizational design tool, but it provides with most of the techniques necessary for data modeling. The product allows for the creation and manipulation of a variety of graphic objects and establishes relationships between them. The standard drawing tools create boxes, ellipses, polygons, connectors, pictures, etc. Connectors can be created between the graphic objects. Once connected, these objects remain linked when their source and destination objects are moved or adjusted.

Unlike most other organizational tools tested, MetaDesign allows for graphic objects and text to be subordinated to other objects with a *Make Region* command. Graphic (and organizational) detail may thus be subordinated on subpages by coarsening it, and transferred

back by refining it. A collection of objects may be grouped temporarily and then manipulated and edited as a single object.

Although MetaDesign's BPR and optimization capabilities partly lag behind those of other tools, its ability to create an active hierarchy of information distinguishes it from competitors. Details can be placed on a subordinate page (of which hundreds can exist), in a subordinate node, or in a subordinate text block. The user can switch from one level to the next with a single command. Subordinate pages, nodes, and text are *children*, their related objects on the higher level in the diagram are *parents*.

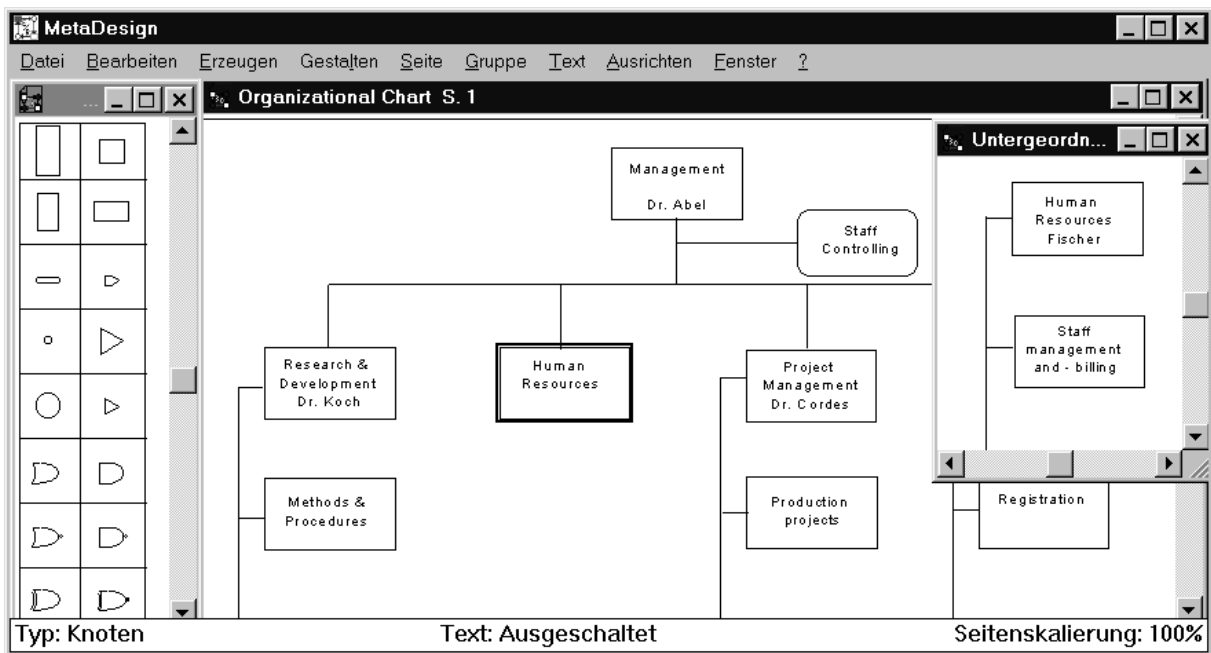


Figure B-4: Sample organization modeled with MetaDesign

Coarsening reduces the graphic complexity in a model by moving detail to a sub-model one level lower in the page hierarchy. This technique resembles organizational hierarchies of units or groups very well.

Traveling the hierarchy has been facilitated by simply double-clicking on the parent object, by choosing *Child* from the menu, or by pressing the keyboard's DOWN arrow. This technique supports the browsing of an organizational model in order to find persons responsible or to obtain information about group memberships, etc.

INCOME Mobile Aufbauorganisation

INCOME is a brand name for a method, for tools, and for services which are offered by PROMATIS and which aim at the modeling, simulation, and realization of business processes. In the GroupOrga evaluation, INCOME Mobile has been tested which represents the available INCOME tools.

INCOME offers flexible methods and user-oriented tools to optimize business processes. This optimization covers the modeling, the simulation, and the enactment of processes. With

INCOME the user can model organizational reality in simple, graphical models that cover various aspects, such as activities, business rules, organizational structure (see Figure B-5), resources, and information objects. INCOME Mobile Aufbauorganisation is the part of the toolset that focuses on the modeling of organizational structures and resources. The structural design takes place in the form of organizational charts only. Hence, only hierarchies can be depicted. Although, INCOME Mobile is intended as a mobile, independent component for BPR, external connectivity takes place only via exporting in form of text (i.e. ASCII) files saved to disk. Hence, no direct, run-time integration with other workflow or processing software is available. INCOME offers simulation, however the test has shown that it is mainly restricted to a process-optimization. Structural or hierarchical simulation or evaluation is not available. A downloading feature allows for a restricted distributed modeling since sub-models can for example be downloaded onto notebook computers.

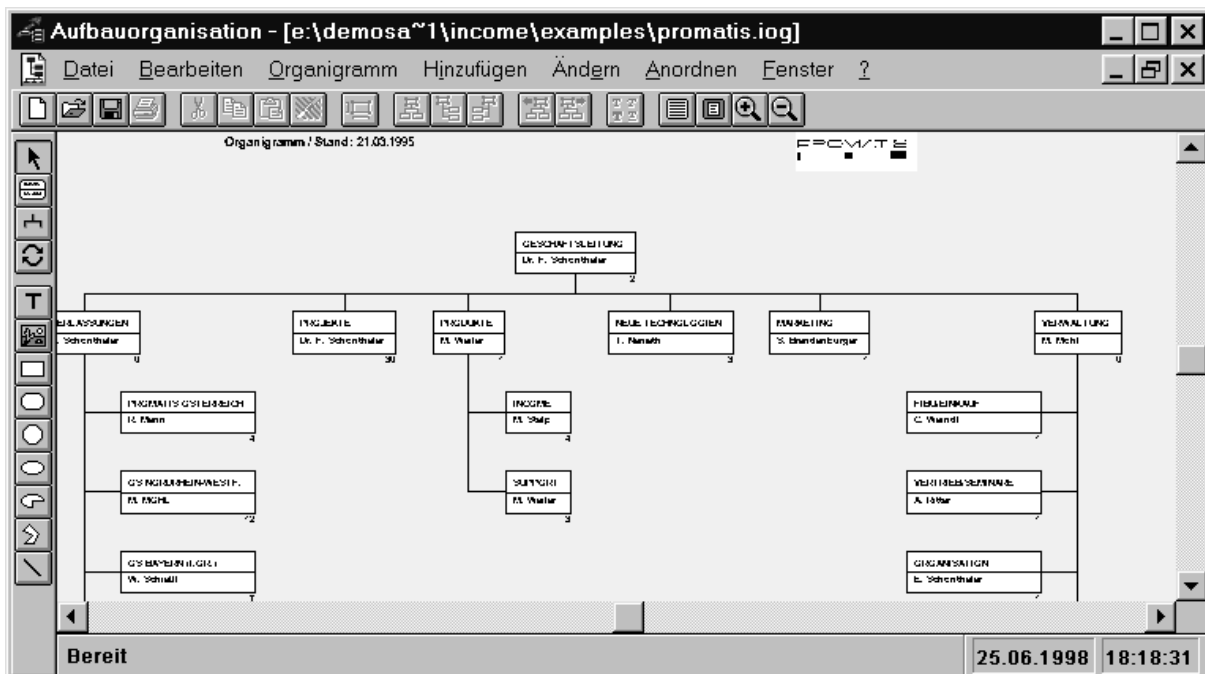


Figure B-5: Sample organization modeled with INCOME Mobile

Prisma-Tool

The PRISMA-Tool offers computer-supported business reengineering with a main focus on procedural and structural models in organizations. It supports the modeling of functional-, data-, process-, and organizational models. The organizational models are described by means of organizational units, employees (persons), responsibilities, and positions.

PRISMA-Tool is an analysis and planning tool, however, the transfer of its data and results has yet to take place manually. Interfaces to relevant workflow management applications were not available at the time of the testing. Some interfaces were mentioned in the documentation, but they are described as 'system specific'. Besides the documentation and presentation of organizational structures, the tool is also aiming at structural optimization.

Before modeling an organizational chart, managers and employees should be modeled in the *staff* module of the tool. It allows the user to add and delete persons, to edit personal data, to search for employees, and to print the staff data as a tabular list. A person's data sheet also lists their position in the organizational hierarchy and their job description and tasks in the organization.

The *organization* module graphically represents the structure of an organization. Although the tool's documentation mentions departments, units, and *groups*, only three types of positions, namely line- (□), staff- (◻), and project-positions (◻) are available. These are displayed in a hierarchical way. Most likely, the project-positions are thought to be used as a description of group-structures. The graphical representation takes place through an editor.

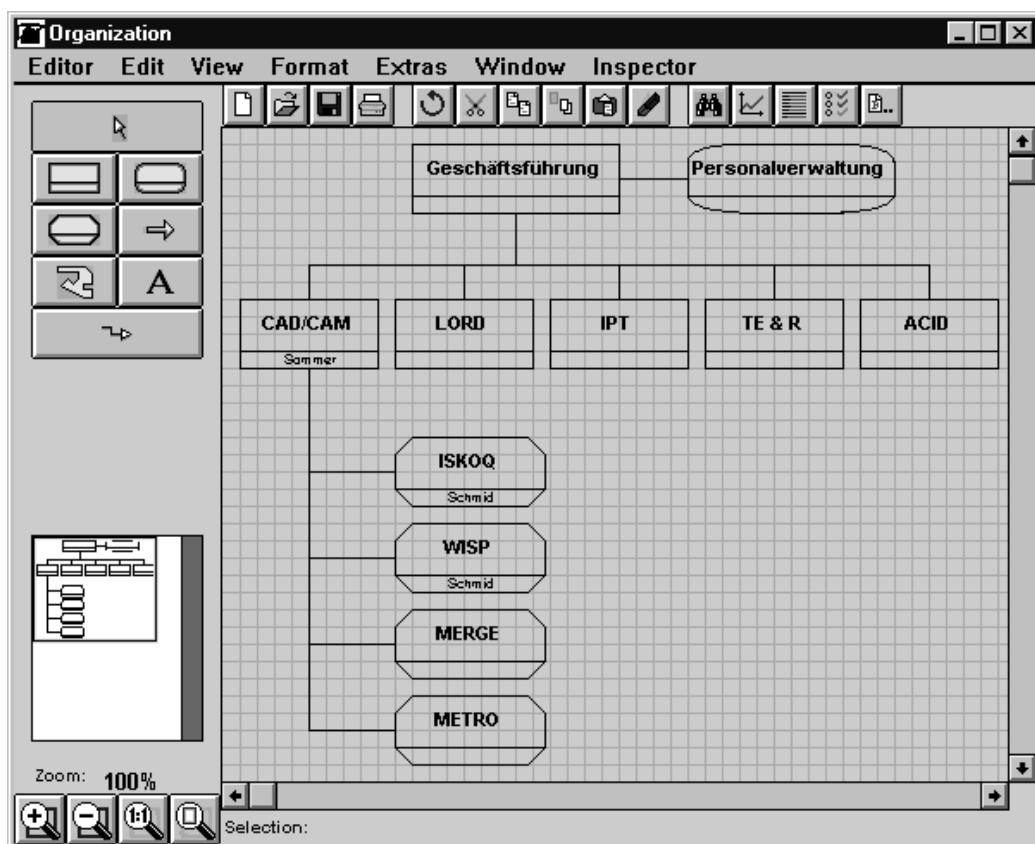


Figure B-6: Sample organization modeled with PRISMA-Tool

The organizational editor allows the user to create or add organizational units by mouse-click. It offers Drag&Drop functionality for most activities such as adding or removing persons or managers, changing a unit's name, editing a unit's description or creating links with other organizational units or positions. A unit's specification includes:

- Boss of the organizational unit
- Employees of the organizational unit
- Job descriptions within the organizational unit
- Data flow between organizational units

- Superordinated unit or subordinated units
- Tasks and activities performed in this unit

The appearance of an organizational chart can be individually adapted to the user's needs. Units can be sized and positioned individually and the form of relations (links) between units can be defined. Moreover, organizational units can be colored independently. Figure B-6 shows how the hierarchical model is represented in the PRISMA-Tool.

SDW-Tools

SDW is a modular toolset which offers modules for organizational design, systems design, and for the modeling of quality systems. All modules access a central repository. The module for organizational design, which was the focus of the test, comprises tools for process-, data-, and organizational modeling, as well as for process analysis and optimization.

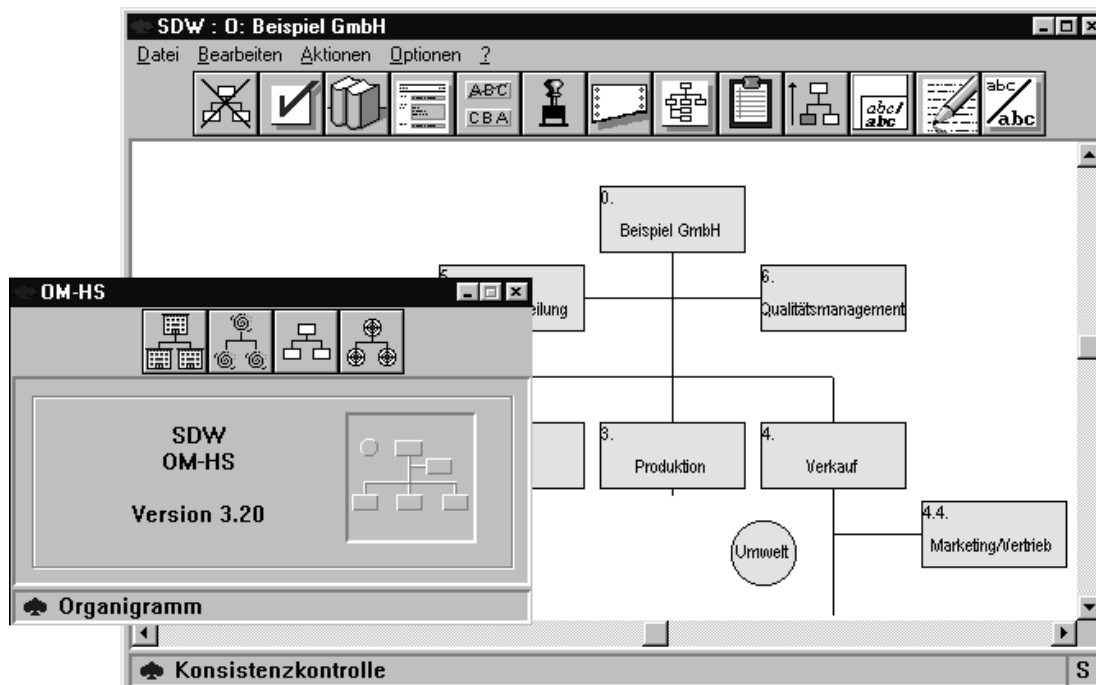


Figure B-7: Sample organization modeled with SDW-OM

The starting point for working with SDW-OM (Organization and Methods), which is the toolset's product for organizational design, is a comprehensive description of the organizational situation. Amongst the six forms of representation that OM offers, the organizational chart covers units, functions, and positions, as well as their relations in an organization. A matrix of relations describes how the organizational units relate to each other, or how functions or positions relate to processes. The representational forms, organizational chart and matrix of relations, are available through the module *structures and relations*.

B.2.4 Complex Organizational Tools with Integrated Functionality

All complex applications for organizational design are based on a comprehensive representation, analysis, and modeling of the complete system *organization*. Thereby, various viewpoints are distinguished which are often separated into the different layers of an organization, e.g.

- organizational and functional model (WHO performs)
- process model
- information/data model

Often, as a preliminary step, the organizational units and the agents who carry out the tasks have to be defined. The results of such modeling serve as a base for the procedural identification of processes, tasks, and activities. Some applications from the list in Table B-1 that fall into this category are described in the following. Whenever possible, the descriptions concentrate on the structural component of the application.

AENEIS

AENEIS allows for the modeling of a fraction of an organization's reality by means of a number of task performers which are arranged on three distinct levels: the level of an organization's environment (e.g. partners, competitors, public authorities), the internal organization (units, roles), and the level of the actual information processed. The modeling process takes into consideration that on all three levels services are provided by the task performers of an organization. The representation of organizational structures provides three different forms of *authorities to issue directives*, it allows grouping such as for team creation, and it provides inheritance information, e.g. for role relationships. The application generates organizational handbooks and job descriptions automatically, however, they result only in printed documentation and not in anything that is available electronically.

The overview of models (see right of Figure B-8) is the core of the application. It is the control panel to invoke all different views or level of modeling, including the organizational chart shown in Figure B-8. This representation of the overall model provides an overview over all modules designed and over the business processes established with AENEIS.

In the *static diagrams* all structural circumstances are modeled, such as the hierarchical layout and the organization's data model. The term *static* diagrams already suggests a drawback of AENEIS: It is not very flexible and it does not support any distributed design.

Organizational units are the main instrument to represent the organization's structure. The hierarchy of an organization is modeled by means of the *authorities to issue directives* between units. The product distinguishes ten different levels for an organization's hierarchy, such as plant, department, main unit, unit, etc. These ten levels must also be used to describe

any form of grouping, such as teams. Organizational units can relate to each other, moreover there can be authority and inheritance relations between roles and units.

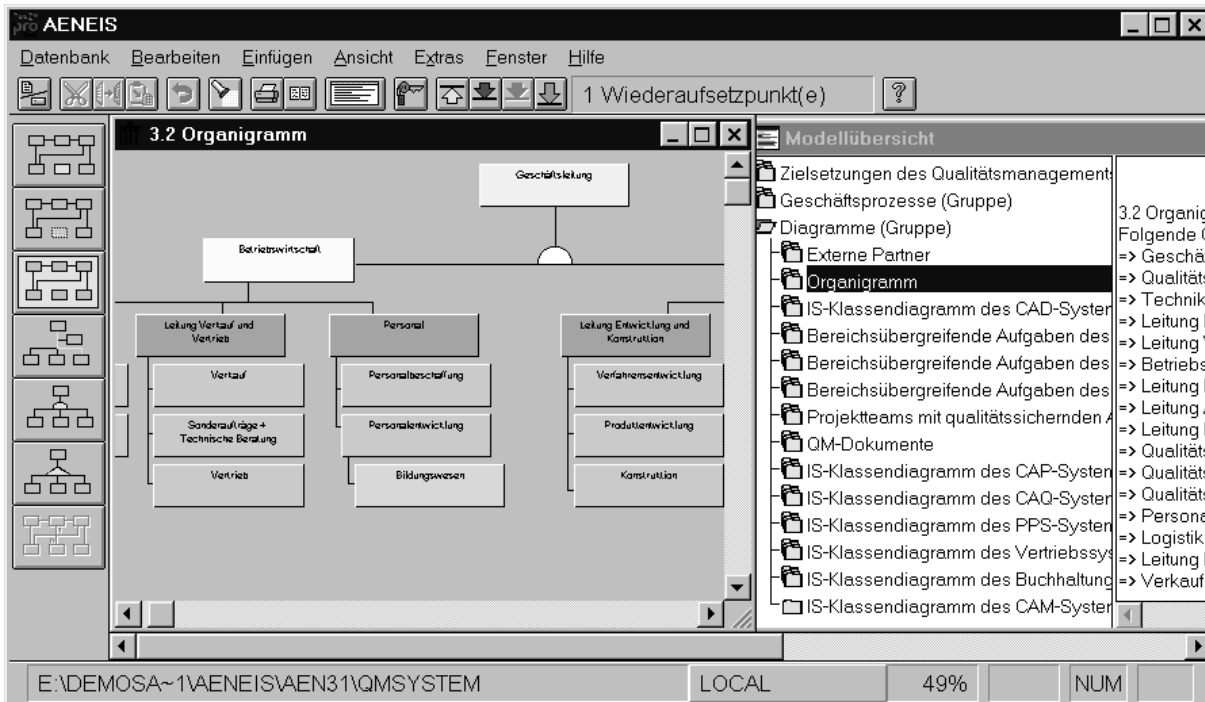


Figure B-8: Sample organization modeled with AENEIS

Roles are used as another entity for the modeling of an organization. They represent a bundle of tasks which can be carried out by more than one unit (including a person). Persons are role players. Similar to units, roles can be in relation with each other, namely in authority and inheritance relation.

Three other complex organizational applications with integrated functionality that have been evaluated in the GroupOrga market survey include:

- ❑ **ARIS-Toolset**
- ❑ **BONAPART**
- ❑ **Nautilus and CW-Kompaß**

While the aforementioned tools and applications have been examined only to a certain degree, these three applications and the VISIO tool from section B.2.1 have been subject to a thorough and detailed investigation in their modeling capabilities and internal data models. Sections B.3 and B.4 introduce the four compared applications.

B.3 Available Organizational Entities in the Examined Tools

While the next section B.4 goes into detail on each of the four examined business process reengineering applications and their capability to support organizational design, the following Table B-2 provides an overview of the organizational entities that are used in the applications'

organizational data models. The result of this comparison, which shows that ARIS-Toolset offers the richest set of organizational entities, has been one source for the definition of organizational entities in the GEIMM.

ARIS-Toolset	BONAPART	Nautilus	VISIO
<p>Type organizational unit</p> <p>An organizational unit can be defined as a <i>type</i> organizational unit. The user can define it as a unit, main unit or group.</p>			
<p>Organizational unit</p> <p>This entity allows for the representation of concrete organizational units of a firm. Organizational units are defined as the performers of tasks that lead to the realization of the organization's goals.</p>	<p>Organizational unity</p> <p>An organizational unity is a group that is lead by a leader. Example: Organizational unit.</p>	<p>Organizational unit</p> <p>Each task is carried out by a single employee or by a group of employees (organizational unit).</p>	<p>Stackable position</p> <p>Symbolizes the position of employees that all belong to the same organizational unit in an organizational chart.</p>
<p>Internal person</p> <p>A relation between a real person and a unit indicates that this person is employed in this unit. If the person is assigned to a position, this indicates the current position assignment. Moreover, persons can be assigned to functions they have to fulfill.</p>			
<p>External person</p> <p>External persons can also be assigned to organizational units.</p>			<p>Freelance employees</p> <p>Freelance employees can be displayed in the organizational chart, in order to integrate them into the organization structure.</p>
<p>Position</p> <p>The smallest identifiable organizational unit. Employees (persons) are assigned to it. Authorities of a position are defined in its description.</p>	<p>Position</p> <p>A position models the scope of duties of one or more persons. Positions are subordinated to organizational units and can be occupied by more than one person.</p>		<p>Position</p> <p>Serves to represent an employees position in an organizational chart.</p>
<p>Type employee</p> <p>An employee type is a typification of persons with similar characteristics. They can refer to similar rights and responsibilities. This entity resembles roles in other BPR applications.</p>	<p>Leader</p> <p>Leaders are those employees that direct organizational unities. Leaders are already assigned to units on an abstract level. The organizational chart automatically shows the correct leader type with the unit.</p>	<p>Employee</p> <p>Each task is carried out from one employee or from a grouping of employees (organizational units).</p>	<p>Organization head</p> <p>Highest management level in organizational chart.</p> <p>Manager</p> <p>Defines further, lower management levels in the organizational chart.</p> <p>Assistant</p> <p>Models assistants or secretaries.</p>

ARIS-Toolset	BONAPART	Nautilus	VISIO
Location Locations specify the place of organizational units, positions, resources, or hardware equipment. It can be a city, a region, a building, a room, or even a single desk.			
Group A group represents a grouping of employees (persons) which work together to solve a given task in a given time. This form is often used for project work.			Team Grouping of positions.
Organizational chart An organizational chart serves as a representation of a cluster of organizational relations on a higher abstraction level.			
Type organizational system The type organizational system represents several technological systems that bear the same or similar characteristics.			
Organizational system Integrated application programs (e.g. Lotus Notes) which bear organizational structures that are taken into consideration when being introduced in an organization.			

Table B-2: Overview over organizational entities

B.4 Detailed Information about Selected Tools of the Survey

B.4.1 ARIS-Toolset by IDS Prof. Scheer GmbH

The ARIS-Toolset has been developed by IDS Prof. Scheer GmbH, Saarbrücken, Germany. ARIS stands for "Architektur integrierter Informationssysteme" and the toolset realizes the ARIS methodology of Scheer (cp. [Scheer 1992] and [Scheer 1995]). On top of a graphical desktop, the software supports the modeling of business models, such as functional-, data-, organizational- and process-models. As modeling aids, the toolset provides best practice reference samples of models that can be copied into or compared to one's own models. The business models can be analyzed, simulated and optimized and new or modified models can

be generated automatically. The toolset also supports the design of individual, i.e. non-standard software packages.

ARIS represents the framework for a description of organizations and their processes. Its method comprises four views: an organizational view, a data view, a functional view, and a steering view. Each of these views is separated into three levels of description: technical concept, IT concept, and implementation (cp. [Scheer 1995], pp. 14ff. and Figure B-9).

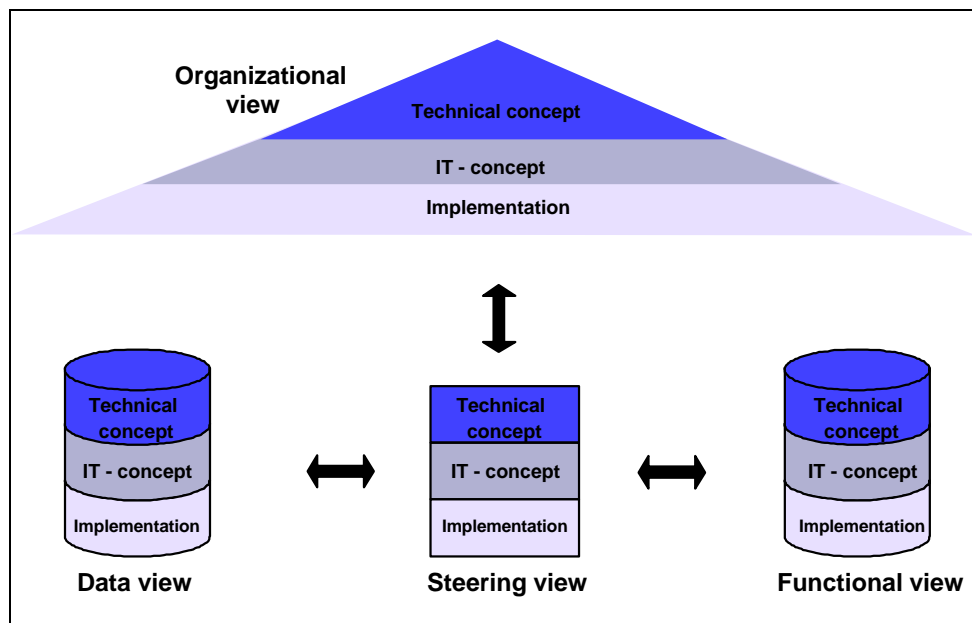


Figure B-9: Views and levels of the ARIS method

This separation into views and levels aims at a general description of organizational systems from the business problem down to the technical implementation. The technical concept (semantical model) describes the business-relevant aspects in a formalized descriptive language. Thereby the technical concept forms the base for a consistent implementation into information technology. When implementing a system, the descriptions of the technical concept are transferred into the terminology of data processing and IT. Technical concept and IT concept are coupled only loosely. At the implementation level the IT concept is transferred into hard- and software implementations.

For each view and level, various types of models may exist. Table B-3 shows which methods are used at the different levels of the organizational view.

	Technical concept	IT concept	Implementation
Organizational view	Organizational chart	Network topology	Network diagram
	Organiz. systems diagram	Locality diagram	Locality diagram (physical)

Table B-3: Description methods at the levels of the organizational view in ARIS

Any information about an organization, its process and information models is stored in a central database. The different views and levels of the ARIS method reduce the modeling complexity since a separation in distinguishable tasks can be obtained.

This investigation of the ARIS method and toolset concentrates on the technical concept and focuses on the organizational view. The remaining views will only be introduced briefly, for further information on the product evaluation refer to [Hoischen/Otto 1997] and to [Scheer 1992] and [Scheer 1995].

The functional view describes the functions (procedures) of an organization to be carried out. A function is an activity to be performed to a data object in order to reach one or more organizational goals. *Data* are information objects which are influenced and modified through activities. The data view uses the entity-relationship-model as a descriptive means. The original ER model has been extended by additional constructs, such as generalization, classification, aggregation, and grouping and is referred to as *extended ER model* (eERM). The steering view is the central component of the ARIS method. It describes the connections between the other three views and it controls the interplay of all objects that are involved in the processing of a business process. The *erweiterte ereignisgesteuerte Prozeßkette* (eEPK) is a notation for describing business processes in ARIS. It is a descriptive notation that combines events and functions into process descriptions which again use entities from the other ARIS views. Besides these notations, ARIS offers two other notations: the *Funktionszuordnungsdiagramm* and the *Vorgangskettendiagramm* (VKD).

In order to integrate more recent organizational concepts, such as lean management or lean production, the organizational view has been defined for ARIS (cp. [Scheer 1995], pp. 23ff.). In a functionally organized organization one organizational unit may be responsible for all products, for all regions, etc. In contrast, a product oriented organization aims at a particular product and the respective functions concerning this product are all carried out through the same organizational unit. Scheer notes that the first form has disadvantages in terms of a need for high coordination, while the second form shows drawbacks in terms of economies of scale (e.g. when purchasing raw materials). He provides hybrid structures for ARIS' organizational view instead ([Scheer 1995], pp. 26-30).

Based upon these considerations, the ARIS-Toolset uses two different reference models for structural organization, a functionally oriented organization and a process oriented organization (cp. [Scheer 1995], p. 30).

In ARIS, the organizational view describes the organizational entities and their relations. The organizational chart or the organizational systems diagram provide the descriptive means. Table B-4 elaborates the (German) entities of ARIS' organizational model.

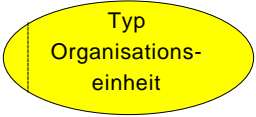

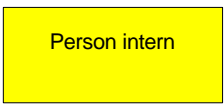
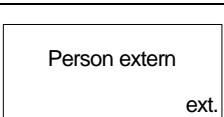
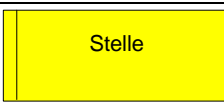
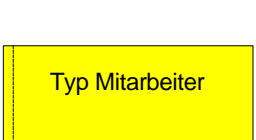
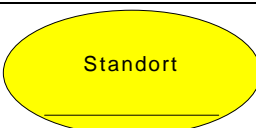
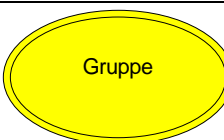
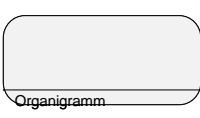
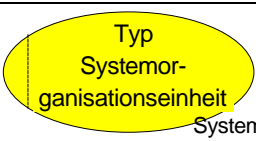

	<p>An organizational unit can be defined as a type organizational unit. The user can define it as a unit, main unit or group. Through this definition the organizational units gain the same characteristics, such as similar rights and responsibilities. Thus, all main units in an organization can be provided with the same organization rights.</p>
	<p>This entity allows for representation of concrete organizational units of a firm. Organizational units are defined as the performers of tasks that lead to the realization of the organization's goals.</p>
	<p>A relation between a natural person and an organizational unit indicates that this person is employed in this unit. If the person is assigned to a position, this indicates the current position assignment. Moreover, persons can be assigned to functions they have to fulfill.</p>
	<p>Similarly to internal persons, external persons can also be assigned to organizational units. An external person could be an external consultant who is not a permanent member of the organization.</p>
	<p>The smallest identifiable organizational unit. Employees (persons) are assigned to it. Authorities of a position are defined in its description.</p>
	<p>An employee type is a typification of persons with similar characteristics. They can refer to similar rights and responsibilities. In eEPKs this entity can be used to define that only a certain type of person is allowed to access certain information. High level managers are an example to define an employee type with certain rights. This entity resembles roles in other BPR applications.</p>
	<p>Locations specify the place of organizational units, positions, resources, or hardware equipment. It can be a city, a region, a building, a room, or even a single desk.</p>
	<p>A group represents a grouping of employees (persons) which work together to solve a given task in a given time. This form is often used for project work.</p>
	<p>An organizational chart serves as representation of a cluster of organizational relations on a higher abstraction level. Such a cluster describes a logical view onto a selection of entities and relations of a data model.</p>
	<p>The type organizational system represents several technological systems that bear the same or similar characteristics.</p>
	<p>Integrated application programs (e.g. Lotus Notes) which bear organizational structures that are taken into consideration when being introduced in an organization.</p>

Table B-4: Organizational entities of the ARIS organizational model

The graphical notation shown and explained in Table B-4 cannot be modified by the ARIS user. Each symbol has a number of attributes assigned, which cannot be modified or extended either. Attributes such as name, identifier, long description, definition, annotation, id, and

creator are available for all entities. All of the above entities can be related to other entities via edges. Between two objects more than one relationship may exist. Table B-5 lists the available relations in ARIS.

Type Organizational Unit	may be created from may be creating may be disciplinary ruled by may disciplinary rule may be superordinated by may be superordinate	Type Organizational Unit
Organizational unit	is disciplinary ruled by is superordinated by is creating is disciplinary ruling is superordinating is subordinating	Organizational unit
Internal/external person	substitutes for is substituted by	Internal/external person
Position	is disciplinary ruled by is superordinated by is superordinating is disciplinary ruling substitutes for is substituted by	Position
Type employee	not available	Type employee
Location	resides at comprises	Location
Group	cooperates with is subordinating is superordinating	Group

Table B-5: Relationship types in ARIS

ARIS supports more relations than those main one listed here. For instance, internal or external persons can substitute positions etc. For a complete list of relations refer to [Scheer 1995]. The entities group and position allow for an organization-wide cooperation across organizational borders.

Figure B-10 shows an organizational chart which has been modeled with ARIS-Toolset.

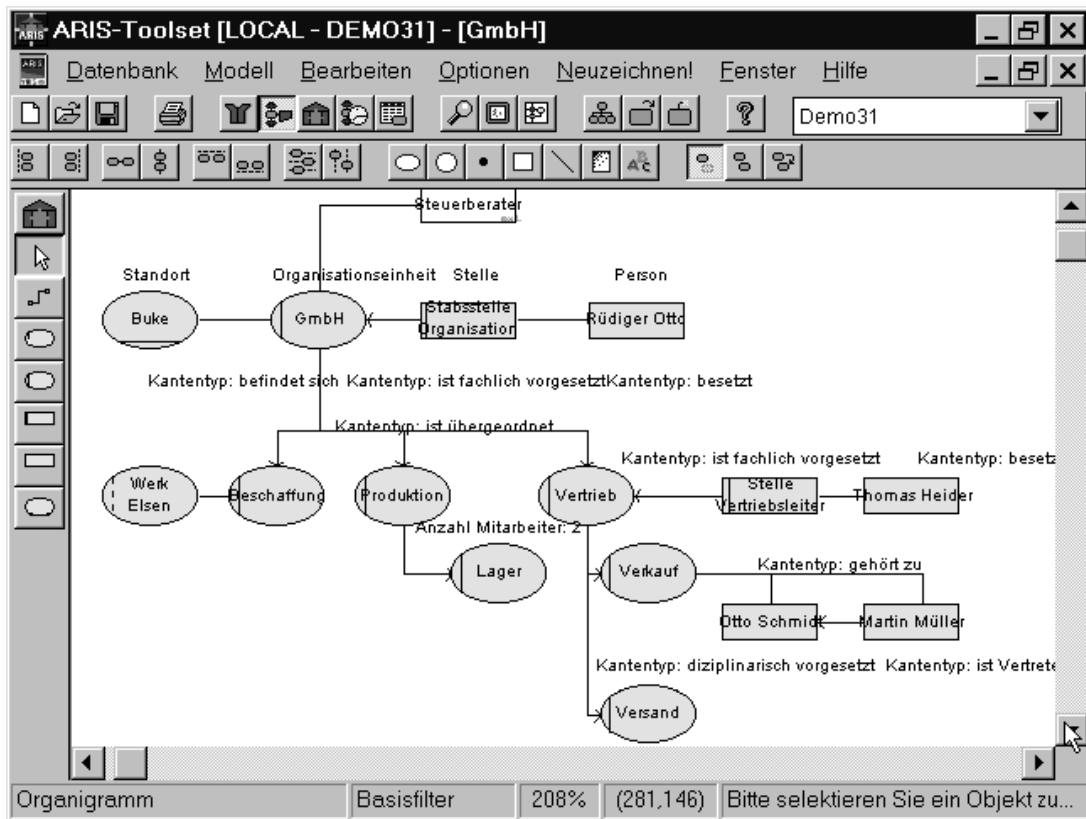


Figure B-10: Sample organization modeled with a German ARIS-Toolset V 3.1

Connecting organizational entities with functions or events requires that the structural organization be modeled in advance. The organizational entities created are managed in an object library for later use. Very complex organizational charts can be reduced in complexity by splitting off lower level charts, such as the organizational unit "Produktion" in Figure B-10.

ARIS-Toolset comprises four modules: the modeling module, the analysis module, the navigation modules, and the simulation module.

The modeling module is the core of ARIS-Toolset. It supports the comprehensive description of an organization in terms of its technical concept and IT realization. All four views, i.e. the organizational view as shown in Figure B-10, the data view, the functional view, and the steering view can be created with this module. All models created here can be documented as textual or as graphical descriptions. The analysis module helps to evaluate the models designed in the modeling module. Based on a reference model, an optimized model is computed and its result can be compared with other reference models. With the navigation module the designed models can be presented. Complex models of the four views can be grasped more easily when displayed in this module. Moreover, the logical relations between the sub-models can be visualized. Only with the simulation module it is possible to create test results that only come about in the daily use of the modeled processes and structures. The results can be statistically evaluated and displayed in tables or graphically.

ARIS-Toolset is not a workflow application in its own right but it provides interfaces to various workflow applications which is a great advantage since integration of organizational

design applications and workflow systems is strongly required: ARIS-MS-Project (Project management), ARIS-IBM-Flowmark (Workflow), ARIS-Multidesk Access (Workflow), ARIS-File Net (Workflow), ARIS-COSA (Workflow), ARIS-VISIO, and ARIS-Workflow.

B.4.2 BONAPART by UBIS GmbH

BONAPART 2.0 which has been developed by UBIS GmbH, Berlin, Germany is an organizational engineering platform with the primary goal to support the reorganization of organizational processes and structures. According to Bach, Brecht, and Österle ([1995], p. 60) it focuses on organizational modeling and provides the possibility for simulation. BONAPART is based on the *Kommunikationsstrukturanalyse* method (KSA) which was invented by Krallmann and his team. This method aims at the support of organizational modeling. For further information on KSA, refer to [Krallmann et al. 1989] and [Krallmann/Klotz 1994].

BONAPART supports three distinct areas of organizational modeling:

- ❑ Analysis and documentation of given circumstances
- ❑ Detection of weak points
- ❑ Development of a to-be situation

The authors point out, however, that no evaluation of the created models and solution in terms of to-be suggestions will be generated. It is up to the respective user to evaluate the available models and to judge upon their usefulness in a given situation.

The tool has a graphical user interface which can be manipulated by menu, by mouse, and by Drag&Drop operations.

BONAPART's concepts is to redesign an organization with the current situation as a starting point. In order to do this, the tool offers a general guideline for the modeling: as a first step the existing organizational structures and processes have to be modeled. This step may involve various employees in the stock-taking process, however, their involvement does not mean that they actually use the software, but that they are subject to questioning and observation. A first result of this process is a "basic model" which will later be refined into a "qualified model". The subsequent analysis and simulation examines the described organizational model statically and dynamically. Its result is an "optimized model" of the organization. In this phase, the analysis covers the static organization, while the simulation covers the dynamic aspects.

The general guidelines for the modeling in BONAPART cover two strategies:

- ❑ Top-down method
- ❑ Bottom-up method

In case of the top-down method, the structural aspects are modeled first, while the processes are covered subsequently. The bottom-up method puts processes first. This description will be done according to the top-down method and the focus will be on the structural modeling.

The first step in this top-down method defines abstract classes of organizational entities first and no concrete entities, such as "Unit Marketing", will be modeled. The aim is to describe a general model that comprises entities like units and project groups which relate to each other. The result is a consistent, user-defined terminology for the following modeling (cp. [Bach/Brecht/Österle 1995], p. 60). The structural organization in BONAPART is based on abstract functional task performers as described in Table B-6.

Functional task performer	Description	Example
Organizational unity	An organizational unity is any type of grouping that is lead by a leader.	Management, Unit, Project, Role, etc.
Leader	Leaders are those employees that direct organizational unities. Leaders are already assigned to units on an abstract level. The organizational chart automatically shows the correct leader type with the unit.	CEO, Unit leader, Project leader, etc.
Position	A position models the scope of duties of one or more persons. Positions are subordinated to organizational units and can be occupied by more than one person.	Clerk for ... , Typist, Trainee, etc.

Table B-6: Abstract organizational entities in the BONAPART meta-model

BONAPART generally distinguishes an abstract and a concrete modeling level. On the abstract level the functional task performers are described in form of system classes, while on the concrete level these classes are assembled in a concrete organizational chart. Only those entities that have already been defined on the abstract modeling level can be created on the concrete level. The scenarios that are defined on the abstract level thus present the basis for the creation of a concrete organizational chart, such as the organizational unity scenario, the leader scenario, and the position scenario as Figure B-11 shows.

Creation of a concrete organizational chart occurs by means of an organizational chart scenario. In order to do this, instances are created from the abstract classes, and the characteristics of the abstract classes are inherited into the instances. Only on this concrete level will the functional task performers be assigned real names, i.e. from the class organizational unit, concrete instances such as "Unit Marketing" or "Unit Sales" will be created.

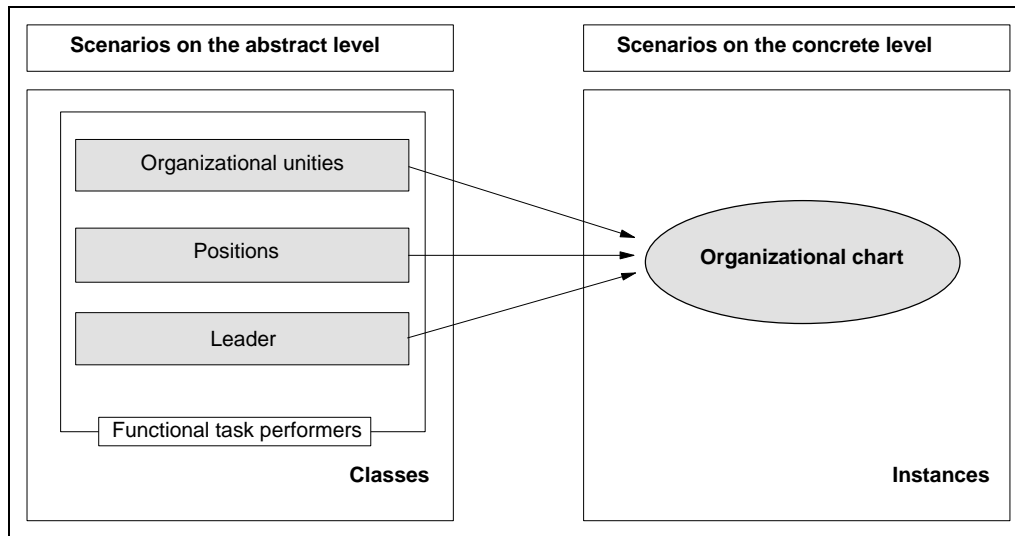


Figure B-11: Scenarios for structural organization in BONAPART

The employees of an organization have to be defined in the persons library of BONAPART. A person can be leader of several units and a position can be occupied by various persons. Moreover, a person can occupy more than one position. The library has a tabular form with each row representing one concrete employee and its attributes.

After all instances have been positioned and persons have been assigned, the organizational chart will be further specified through a hierarchy. This takes place by introducing relations into the model. BONAPART is able to predefine two types of relations:

- Is disciplinary ruling
- Is superordinating

Additionally, user-defined relations can be defined within the relation manager.

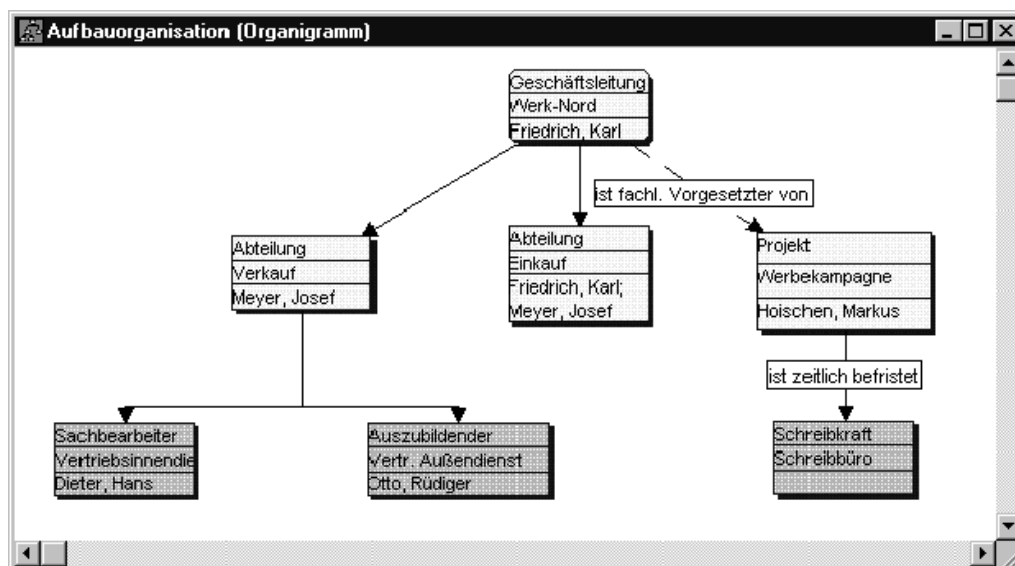


Figure B-12: Sample organization modeled with BONAPART

BONAPART also offers the function of defining position descriptions for the respective functional task performers. Such a position description is compiled from a list of activities and tasks that should be performed by the position to be described. In other words, each activity which is to be performed by a specific person is included in the position description.

Figure B-12 shows a fraction of a sample organizational chart scenario with a predefined and a user-defined relation type.

The analysis function in BONAPART supports the user to find weak points in both the structural and the procedural design. With it, static analyses of the organizational structures and consistency checks can be performed. Possible analyses of the structural organization primarily focus on an information aspect and not its layout. Three types of analysis are offered:

- ❑ Span of leadership
- ❑ Number of subordinated employees (per manager)
- ❑ Position descriptions

All results are in tabular form for which the data is read from the respective central database. *Span of leadership* indicates how many subordinated units and positions exist for a particular unit in question. *Number of subordinates* determines how many instances of person exist for a particular organizational unit. This also includes indirect subordinates. Finally, *position descriptions*, displays the descriptions of all employees' positions in tabular form. This is a helpful analysis if an organizer wants to obtain an overview of all activities and tasks which are performed in a certain unit or grouping.

The last step in BONAPART's general guideline for modeling is the creation of an optimized model through a stepwise refinement of the model in question. Recurrent analyses allow for the improvement of the model with every pass until a satisfactory result has been reached. The simulation module is only applicable to the process definitions in BONAPART. It only focuses on dynamic behaviors in an organization and the organizational structure is defined static in BONAPART.

UBIS offers an interface to the workflow management system InConcert. This interface transfers BONAPART's organizational charts, process models, activities, and information objects to the WfMS. Similarly, a transfer between Pavone Informationssysteme GmbH's WfMS ESPRESSO and BONAPART has been realized.

B.4.3 Nautilus by integraISA GmbH

Nautilus is a BPR tool developed by integraISA GmbH, Bielefeld, Germany. It offers the option to graphically describe both processes and organizational structures and to analyze the models afterwards. In contrast to the two products presented so far, ARIS-Toolset and BONAPART, Nautilus provides no simulation.

Nautilus is based on a process-oriented modeling method: The processes of an organization should be modeled first and only afterwards can the organizational structure be designed by assigning persons to process tasks. This supplementary assignment of persons to tasks, i.e. the subsequent design of organizational structures after the processes have been designed is not considered an option, but it is expressed as the optimal way of modeling (cp. Nautilus online help).

The modeling takes place according to the "Nautilus-method of modeling". Its idea is to separate technical, organizational, and technological aspects of an organization in order to reduce complexity in the modeling process. The core concept of modeling with Nautilus is to reduce any action in an organization to its main causality, i.e. to separate the main task to be performed from any organizational or technological aspects. What remains afterwards is a mere description of what has to be done (technical aspect), without saying where it has to be performed (organizational aspect), and with what technological means (technological aspect).

The intended result is to identify general tasks in an organization that can be reused and which are universally defined. Currently, such standardization of terminology in Nautilus has been done only for process description. Terminology in the area of structural organization has not yet been standardized according to the described Nautilus method.

Nautilus uses the standardized terminology in its process components (process, function, group of functions), which are combined in universally valid reference models. Currently, such reference models exist for warehousing and accounting.

Nautilus is a young product which so far mainly concentrates on processes. The process components *process*, *function*, and *group of functions* were the first entities to which attention was directed. Hence, modeling of structural organization is still in its infancy, however, Nautilus already displays interesting concepts for doing so.


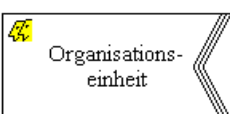
	<p>Employee</p> <p>Each task is carried out by one employee. Since this entity type has attributes and since information can be stored for it, by doing so instances of it are established.</p>
	<p>Organizational unit</p> <p>Each task is carried out by a group of employees, i.e. an organizational unit. Since this entity type has attributes and since information can be stored for it, by doing so instances of it are established.</p>

Table B-7: Organizational entities of the Nautilus organizational model

Nautilus' main resource for structural information is an organizational database which manages all specified organizational units and employees. These two entities are currently the only ones available (see Table B-7). Since the involvement of an employee in the organization's processes is difficult to determine, Nautilus offers information to examine this (cp. Figure B-13).

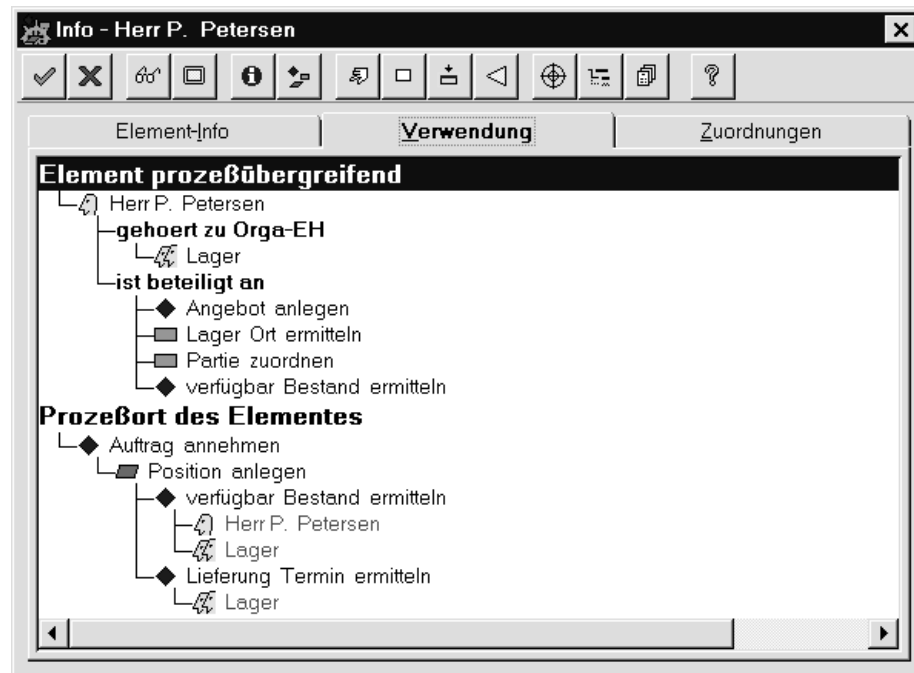


Figure B-13: Information about an instance of employee in Nautilus

Attributes of *employee* are title, first name, name, id, room, position, telephone, fax. Additionally such a specification can contain a description of the position obtained and a list of previous jobs and positions. Moreover, the user can find out which organizational unit the employee belongs to and which tasks the employee performs in the organization's processes. The employee's direct superior is also listed.

Organizational units and employees are defined via forms as text documents. In contrast to ARIS-Toolset and BONAPART, modeling in Nautilus does not take place by graphical means. Process visualization is generated from the textual process description. The textual description of organizational structure has as yet no graphical equivalent. According to the manufacturer's announcements, an organizational chart module will be implemented soon. At the moment, a report generator produces a report on the organizational structure and an integrated graphical drawing tool has to be used to visualize an organizational chart. From there, name changes for any organizational entity will be propagated from the textual description to the graphic and vice versa. This ensures consistency in the modeling process.

Further information regarding the organizational structure can be gained from a visualization of relations between employee or organizational unit entities with other entities, such as tasks, resources, media, etc. as shown in Figure B-14. On the left of Figure B-14 organizational entities are displayed, while on the right other, related entities are shown.

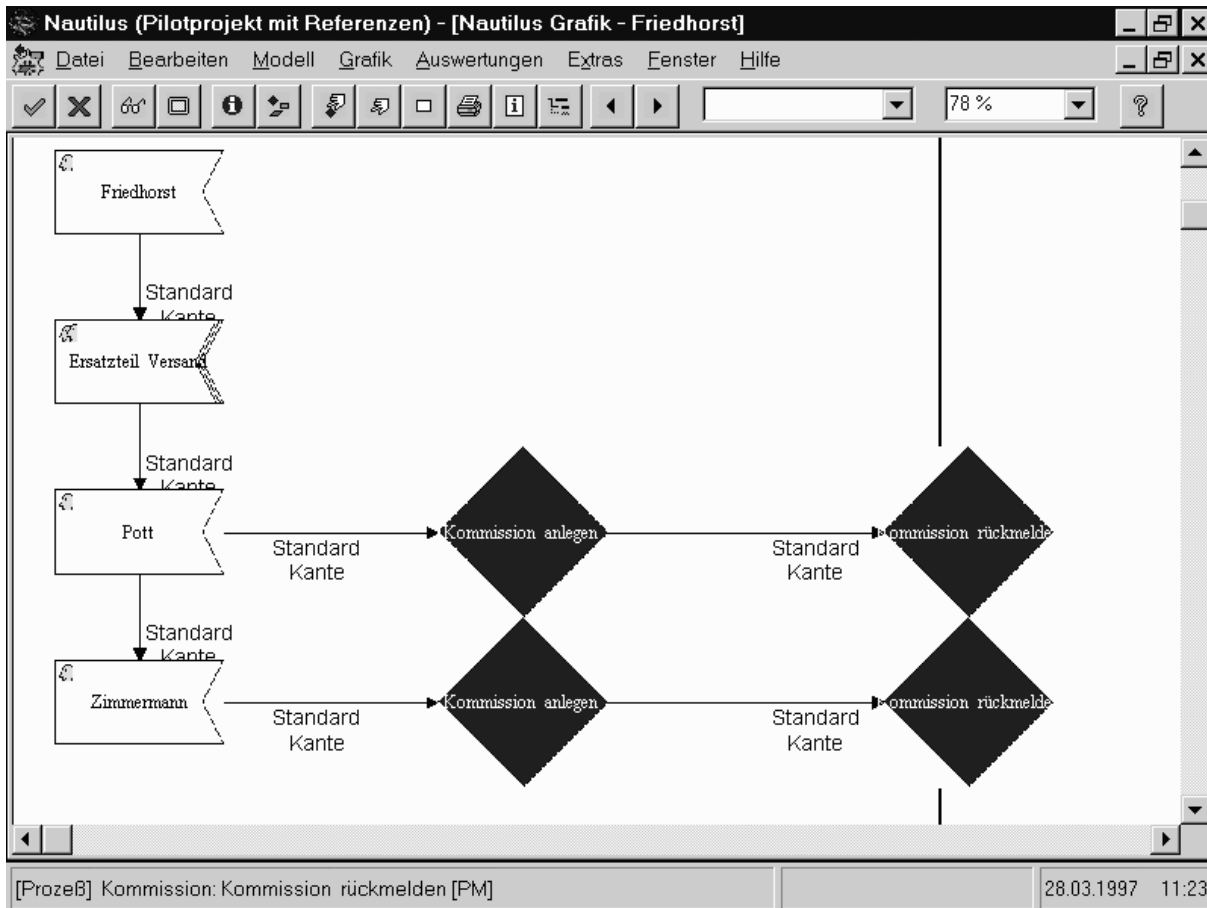


Figure B-14: Relationship of entities in Nautilus

B.4.4 VISIO by VISIO GmbH

VISIO 4.0 by VISIO GmbH Munich, Germany, is a tool to represent and visually describe organizations. According to the classification from above it falls into the category of tools for drawing and presentation purposes. The main difference between VISIO and the three other products is thus, that no analysis and simulation for processes or organizational structures is available. Hence, VISIO may be interesting for those users that are interested in documenting their organizational structures according to DIN EN ISO 9000 ff and to become ISO 9000-certified. The main emphasis for VISIO is on the traditional organizational tasks such as creating organizational and flow charts in order to describe structures and processes. Its main advantage lies in its very good diagramming capabilities with manifold features for creating an individualized design.

Such diagramming features do support the idea of BPR, however, VISIO cannot be positioned as a BPR tool. Most importantly, it does not provide any analysis and simulation features. A Gartner Group study (cp. chapter 3 and [Lindo 1996]) positions VISIO as a niche product. VISIO can cooperate with some full-size BRP products via interfaces, such as ARIS-Toolset. This ARIS-VISIO interface allows for the use of ARIS symbols in VISIO and to import comprehensive organizational charts that have been modeled in VISIO into ARIS-Toolset for

analysis and simulation. "VISIO Business Modeler", an add-on for VISIO, understands ARIS' *ereignisgesteuerte Prozessketten* (EPK) and it can thus design business processes for SAP R/3.

In order to design a concrete model in VISIO, the user has to decide on a template. In this case, the *organizational chart* template will be used as an example. The template contains the graphical objects, so called *shapes* which can be used in a diagram of this type. Shapes from different templates can be combined into one diagram if necessary. For instance, a resource *computer* can be assigned to a *person* in an organizational chart. However, no such assignment or any other relation which is modeled in such diagrams has any effect on programs or WfM applications—it only serves the purpose of visualization. Hence, in contrast to ARIS-Toolset, BONAPART, or Nautilus, a relation such as *employee X uses resource Y* has no consequence in any workflow processing.

The *organizational chart* template provides the following master shapes shown in Table B-8.








Master-Shape	Name	Description
	Organization head	Highest management level in organizational chart.
	Manager	Defines further, lower management levels in the organizational chart.
	Position	Serves to represent an employee's position in an organizational chart.
	Stackable position	Symbolizes the position of employees that all belong to the same organizational unit in an organizational chart.
	Assistant	Models assistants or secretaries.
	Freelance employees	Freelance employees can be displayed in the organizational chart, in order to integrate them into the organization structure.
	Team	Grouping of positions.

Table B-8: VISIO organizational chart master shapes

Drag&Drop actions position the shapes on a drawing pad and are treated as instances afterwards. Relations can be established between these instances and the user can create and position own shapes in the diagram. Shapes are characterized by attributes which can be edited and retrieved via pop-up windows. Besides predefined attributes such as *organizational unit* and *telephone*, additional attributes may be defined. Organizational charts can also be generated from external sources such as Excel tables (e.g. a human resource database). Each diagram created with VISIO can be exported into other application including Lotus Notes. In this particular case Notes' F/X interface allows for the bi-directional exchange of data on a field basis.

Figure B-15 shows a sample organization modeled with VISIO.

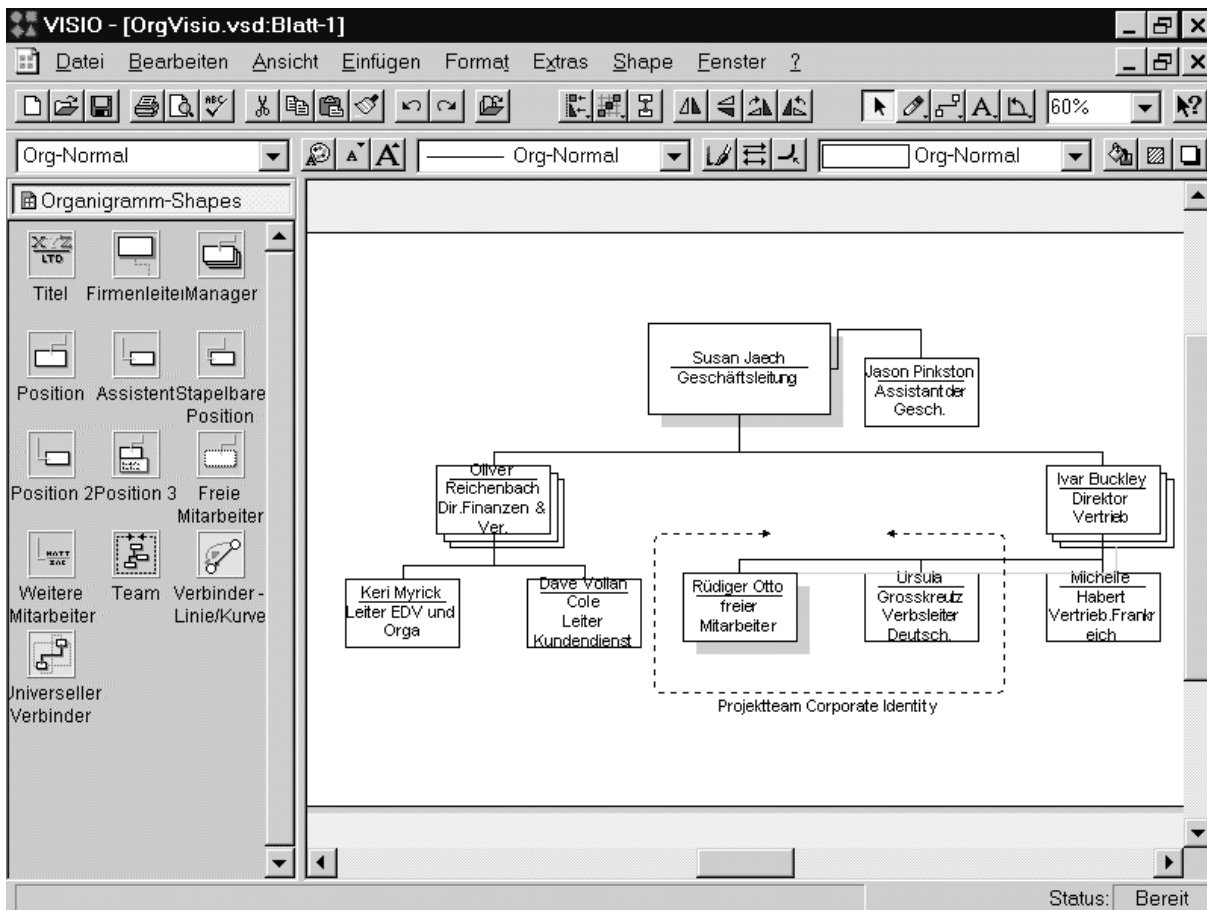


Figure B-15: Sample organization modeled in VISIO

B.5 Results of the Benefit Analysis applied in the Market Survey

Benefit analysis is a useful scoring technique to evaluate alternatives regarding software products, projects, or hardware systems (cp. [Domsch/Reinecke 1989], p. 151). The complexity of the alternatives often hinders the conclusion of an intuitive decision. The scoring method in turn, rates the products by means of criteria and weighting. The method is a decision *aid* in order to evaluate the software products according to the criteria. However, it must always be made clear that the benefit analysis is based on subjectively chosen criteria and weighting. Thus, it cannot present an optimized solution in a mathematical sense and it is only a supportive means in finding a solution which is still based on individual decisions.

The following table presents the results of the benefit analysis that has been conducted in the course of the GroupOrga project in order to evaluate the four BPR products ARIS-Toolset, BONAPART, Nautilus, and VISIO which were introduced above. Chapter 3 has verbally commented on these results and in [Hoischen/Otto 1997] further evaluation and discussion of this GroupOrga subproject is given. Since due consideration has been given to the terminology used in the criteria list, the result Table B-9 is reproduced in its original German version. This was done, since the use of English terms and criteria may falsify the results due to misunderstandings in translations and terminology.

Kriterienkatalog	Gewichtung Rubrik	Gewichtung Oberkriterium	Gewichtung Unterkriterium	Anteil des Unterkriteriums am Gesamtanzahlwert	ARIS-Toolset Version 3.1 Demo		BONAPART Version 2.0 Demo		Nautilus Version 1.2 beta		VISIO Version 4.0 Demo	
					Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen
1 Allgemeine Kriterien	10	10,00										
11 Visualisierung und Flexibilität der grafischen Aufbereitung	3	1,00										
111 Grafische Darstellung von Organisationsstrukturen		0,150	0,450	4	1,80	2	0,90	5	2,25	5	2,25	
112 Grafische Darstellung von Geschäftsprozessen		0,150	0,450	4	1,80	3	1,35	5	2,25	5	2,25	
113 Verwendung und Vorgabe standardisierter Symbole		0,150	0,450	6	2,70	1	0,45	6	2,70	6	2,70	
114 Individuelle Erstellung von Symbolen (Objekten)		0,150	0,450	0	0,00	3	1,35	5	2,25	6	2,70	
115 Verwendung von Objektbibliotheken		0,050	0,150	6	0,90	4	0,60	6	0,90	2	0,30	
116 Individuelle Gestaltung von Symbolen (Objekten)		0,100	0,300	0	0,00	2	0,60	4	1,20	6	1,80	
117 Transparente Darstellung der Gesamtunternehmung durch Hyperlinkstrukturen		0,100	0,300	4	1,20	4	1,20	5	1,50	1	0,30	
118 Layoutgenerierung bei selbst erstellten Grafiken		0,050	0,150	1	0,15	0	0,00	0	0,00	6	0,90	
119 Zoomfunktion		0,025	0,075	5	0,38	5	0,38	6	0,45	6	0,45	
1110 Rasterfunktion		0,025	0,075	6	0,45	6	0,45	6	0,45	6	0,45	
1111 Horizontale und vertikale Ausrichtung		0,025	0,075	6	0,45	6	0,45	0	0,00	6	0,45	
1112 Leichte Anpassung von bestehenden Grafiken		0,025	0,075	3	0,23	4	0,30	4	0,30	5	0,38	
12 Anwenderfreundlichkeit	3	1,00										
121 Integration eines interaktiven Lernprogramms		0,300	0,900	2	1,80	4	3,60	5	4,50	0	0,00	
122 Einführungsdemo (Video)		0,100	0,300	0	0,00	0	0,00	4	1,20	2	0,60	
123 Intuitive Menütechnik		0,100	0,300	6	1,80	5	1,50	6	1,80	6	1,80	
124 Verschiedene Spracheinstellungen möglich		0,150	0,450	6	2,70	0	0,00	0	0,00	0	0,00	
125 Hilfe über Menüleiste vorhanden		0,050	0,150	6	0,90	6	0,90	6	0,90	6	0,90	
126 Direkte Kontexthilfe verfügbar		0,100	0,300	3	0,90	3	0,90	5	1,50	6	1,80	
127 Drag & Drop-Unterstützung		0,050	0,150	2	0,30	2	0,30	6	0,90	6	0,90	
128 Undo-Funktion vorhanden		0,100	0,300	1	0,30	1	0,30	0	0,00	6	1,80	
129 Verschiedene Druckoptionen ermöglichen aussagekräftige Ausschnitte		0,050	0,150	4	0,60	4	0,60	2	0,30	1	0,15	
13 Systeminformationen	2,5	1,00										
131 Das Produkt ist auf verschiedenen Betriebssystemen einsatzfähig (Windows 95, Windows NT, OS/2 WARP)		0,300	0,750	6	4,50	3	2,25	3	2,25	3	2,25	
132 Das Produkt ist multiuserfähig		0,250	0,625	0	0,00	0	0,00	0	0,00	0	0,00	
133 Werkzeug verfügt über ein Online-Repository		0,250	0,625	3	1,88	2	1,25	4	2,50	0	0,00	
134 Das Produkt ist netzwerkfähig (Client-Server-Architektur)		0,200	0,500	6	3,00	6	3,00	0	0,00	6	3,00	
14 Schulung & Service	0,5	1,00										
141 Basisschulungen beim Kauf inbegriffen		0,300	0,150	0	0,00	0	0,00	0	0,00	0	0,00	
142 Unterstützung bei Installation, Implementierung und Releasewechsel		0,100	0,050	6	0,30	6	0,30	6	0,30	6	0,30	
143 Methoden- und Modellierungsseminar		0,300	0,150	6	0,90	6	0,90	6	0,90	6	0,90	
144 Hot-Line Service (z.B. Remote-Fehlerdiagnose)		0,300	0,150	6	0,90	6	0,90	6	0,90	6	0,90	
15 Rahmendaten der Software	0,5	1,00										
151 Miete		0,100	0,050	6	0,30	6	0,30	0	0,00	0	0,00	
152 Leasing		0,100	0,050	0	0,00	0	0,00	0	0,00	0	0,00	
153 Einzellizenz		0,400	0,200	6	1,20	6	1,20	6	1,20	6	1,20	
154 Sammellizenz		0,400	0,200	6	1,20	6	1,20	0	0,00	6	1,20	
16 Einsatzbereich des Produktes	0,5	1,00										
161 Kleine Unternehmen		0,100	0,050	1	0,05	1	0,05	1	0,05	6	0,30	
162 Mittelständische Unternehmen		0,300	0,150	3	0,45	3	0,45	3	0,45	4	0,60	
163 Große Unternehmen, Konzerne		0,600	0,300	5	1,50	5	1,50	5	1,50	2	0,60	
Nutzen der Rubrik GESAMT												
Erfüllungsgrad in der Rubrik												
Anteil am Gesamtnutzen des Werkzeugs												
					35,53	29,43	35,40	34,13				
					59,21%	49,04%	59,00%	56,88%				
					8,82%	9,40%	11,67%	32,11%				

Kriterienkatalog	Gewichtung Rubrik	Gewichtung Oberkriterium	Gewichtung Unterkriterium	Anteil des Unterkriteriums am Gesamtsummenwert	ARIS-Toolset Version 3.1 Demo		BONAPART Version 2.0 Demo		Nautilus Version 1.2 beta		VISIO Version 4.0 Demo	
					Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen
2 Methodik	15	15,00										
21 Methode		5	1,00									
211 Dem Tool unterliegt eine wissenschaftlich fundierte Methode			0,200	1,000	6	6,00	6	6,00	6	6,00	0	0,00
212 Dem Tool liegt ein Vorgehensmodell als roter Faden zugrunde			0,300	1,500	4	6,00	6	9,00	6	9,00	0	0,00
213 Kompatibilität der implementierten Methode mit der eigenen Vorgehensweise			0,300	1,500	6	9,00	6	9,00	6	9,00	0	0,00
214 Einheitliche Abfolge bei Erstellung der Modelle			0,200	1,000	6	6,00	6	6,00	6	6,00	6	6,00
22 Plausibilitätsprüfung		6	1,00									
221 Inkonsistenzen in den Modellen werden automatisch vom System erkannt			0,500	3,000	6	18,00	6	18,00	0	0,00	0	0,00
222 Nicht zulässige Verknüpfungen werden automatisch untersagt			0,250	1,500	6	9,00	6	9,00	6	9,00	0	0,00
223 Vom Anwender definierte Restriktionen in der Modellierung werden vom System verarbeitet			0,250	1,500	0	0,00	2	3,00	0	0,00	0	0,00
23 Unterstützte Phasen der organisatorischen Gestaltung		4	1,00									
231 Aufnahme des IST-Zustandes			0,100	0,400	6	2,40	6	2,40	6	2,40	6	2,40
232 Darstellung des IST-Zustandes			0,200	0,800	6	4,80	6	4,80	6	4,80	6	4,80
233 Analyse des IST-Zustandes			0,200	0,800	6	4,80	6	4,80	6	4,80	0	0,00
234 Erstellung eines SOLL-Konzeptes			0,300	1,200	6	7,20	6	7,20	6	7,20	6	7,20
235 Simulation von IST- und SOLL-Konzept			0,200	0,800	6	4,80	6	4,80	6	4,80	0	0,00
Nutzen der Rubrik GESAMT						78,00		84,00		63,00		20,40
Erfüllungsgrad in der Rubrik						86,67%		93,33%		70,00%		22,67%
Anteil am Gesamtnutzen des Werkzeugs						19,36%		26,82%		20,78%		19,20%

Kriterienkatalog	Gewichtung Rubrik	Gewichtung Oberkriterium	Gewichtung Unterkriterium	Anteil des Unterkriteriums am Gesamtnutzenwert	ARIS-Toolset Version 3.1 Demo		BONAPART Version 2.0 Demo		Nautilus Version 1.2 beta		VISIO Version 4.0 Demo	
					Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen
3 Kriterien der Aufbauorganisation	20	20,00										
<i>31 Organisatorische Objekte</i>		2,75	1,00									
311 Das Werkzeug bietet verschiedene aufbauorganisatorische Konstrukte			0,500	1,375	5	6,88	4	5,50	3	4,13	6	8,25
312 Organisatorische Objekte können nur nach definierten Regeln miteinander verknüpft werden			0,100	0,275	6	1,65	6	1,65	6	1,65	0	0,00
313 Benutzerdefinierte Entitäten können erstellt werden			0,400	1,100	0	0,00	6	6,60	6	6,60	6	6,60
<i>32 Kanten / Relationen</i>		2,25	1,00									
321 Systemdefinierte Kantentypen (Relationen) stehen dem Anwender zur Verfügung			0,200	0,450	6	2,70	3	1,35	4	1,80	0	0,00
322 Erstellung benutzerdefinierter Kantentypen (Relationen) möglich			0,300	0,675	0	0,00	6	4,05	6	4,05	0	0,00
323 Benutzerdefinierte Relationen werden vom System auf Konsistenz überprüft			0,300	0,675	0	0,00	6	4,05	0	0,00	0	0,00
324 Beschriftung der Kanten kann wahlweise ein- und ausgeblendet werden			0,200	0,450	6	2,70	6	2,70	0	0,00	0	0,00
<i>33 Organigrammerstellung & Benutzerfreundlichkeit</i>		3,25	1,00									
331 Automatische Generierung des Organigramms aus dem Prozeß heraus			0,250	0,813	0	0,00	0	0,00	0	0,00	0	0,00
332 Die Änderung der Datenbasis bewirkt einen dynamischen Abgleich des Organigramms			0,300	0,975	3	2,93	3	2,93	6	5,85	0	0,00
333 Änderungen im Organigramm bewirken eine dynamische Veränderung der Datenbasis			0,300	0,975	3	2,93	0	0,00	6	5,85	0	0,00
334 Entitäten organisatorischer Einheiten unterscheiden sich eindeutig aufgrund ihrer intuitiven grafischen Darstellung			0,150	0,488	2	0,98	1	0,49	6	2,93	4	1,95
<i>34 Beschreibung der im Modell verwendeten Entitäten (z.B. Person, Abteilung, Gruppe etc.)</i>		0,75	1,00									
341 Hinterlegung von Freitext für jede im Modell verwendete Entität möglich			0,300	0,225	4	0,90	1	0,23	6	1,35	6	1,35
342 Beschreibungen der jeweiligen Entitäten werden generiert			0,700	0,525	4	2,10	4	2,10	6	3,15	0	0,00
<i>35 Skillmanagement</i>		3,25	1,00									
351 Freitext Eingabe für Qualifikationen			0,150	0,488	3	1,46	1	0,49	6	2,93	2	0,98
352 Verknüpfung auf externe Dokumente um Qualifikation abzubilden			0,100	0,325	1	0,33	1	0,33	5	1,63	6	1,95
353 Abbildung der Qualifikation und Erfahrungen der Mitarbeiter			0,400	1,300	2	2,60	2	2,60	4	5,20	2	2,60
354 Transparenz der MA-Qualifikationen möglich			0,350	1,138	1	1,14	1	1,14	3	3,41	0	0,00
<i>36 Modelle</i>		1,75	1,00									
361 Referenzmodelle werden angeboten			0,300	0,525	6	3,15	0	0,00	0	0,00	0	0,00
362 Dem Werkzeug unterliegt ein vordefiniertes Metamodell (Datenmodell)			0,150	0,263	6	1,58	3	0,79	6	1,58	0	0,00
363 Benutzerdefinierte Anpassung des Metamodells			0,550	0,963	0	0,00	6	5,78	6	5,78	0	0,00

Kriterienkatalog	Gewichtung Rubrik	Gewichtung Oberkriterium	Gewichtung Unterkriterium	Anteil des Unterkriteriums am Gesamtanzwert	ARIS-Toolset Version 3.1 Demo		BONAPART Version 2.0 Demo		Nautilus Version 1.2 beta		VISIO Version 4.0 Demo	
					Teil- nutzen	Gewich- teter Teil- nutzen	Teil- nutzen	Gewich- teter Teil- nutzen	Teil- nutzen	Gewich- teter Teil- nutzen	Teil- nutzen	Gewich- teter Teil- nutzen
37 Ressourcenmanagement	0,75	1,00										
371 Ressourcen (Sachmittel) können Organisationseinheiten zugeordnet werden		0,200	0,150		3	0,45	3	0,45	6	0,90	2	0,30
372 Verfügbarkeitsprüfung von Mitarbeitern im Hinblick auf Einsetzbarkeit		0,600	0,450		3	1,35	3	1,35	5	2,25	0	0,00
373 Verfügbarkeitsprüfung von Sachmitteln im Hinblick auf Einsetzbarkeit		0,200	0,150		3	0,45	3	0,45	5	0,75	0	0,00
38 Rechtemanagement	3,5	1,00										
381 Mitarbeiter können aufgrund einer differenzierten Rechtsstruktur Organisationsgestaltung vornehmen		1,000	3,500		6	21,00	2	7,00	1	3,50	0	0,00
39 Aufgabenmanagement	0,5	1,00										
391 Aufgabenträger können nicht nur einzelne Personen, sondern auch Abteilungen, Rollen usw. sein		1,000	0,500		6	3,00	6	3,00	6	3,00	6	3,00
310 Analyse	1,25	1,00										
3101 Die Aufbauorganisation kann analysiert und in Berichtsform ausgewiesen werden		0,300	0,375		4	1,50	3	1,13	6	2,25	0	0,00
3102 Die Aufbauorganisation kann aufgrund verschiedener Kriterien ausgewertet werden.		0,150	0,188		6	1,13	2	0,38	6	1,13	0	0,00
3103 Es können Gestaltungsempfehlungen im Hinblick auf prozeßorientierte Aufbaustrukturen abgeleitet werden		0,300	0,375		5	1,88	4	1,50	5	1,88	0	0,00
3104 Die durch die Analyse gewonnenen Ergebnisse werden in Tabellenform ausgegeben		0,100	0,125		6	0,75	4	0,50	4	0,50	0	0,00
3105 Die Analyseergebnisse werden zusätzlich grafisch dargestellt		0,150	0,188		4	0,75	4	0,75	4	0,75	0	0,00
Nutzen der Rubrik GESAMT						66,25		59,25		74,76		26,98
Erfüllungsgrad in der Rubrik						55,21%		49,38%		62,30%		22,48%
Anteil am Gesamtnutzen des Werkzeugs						16,44%		18,92%		24,66%		25,38%

Kriterienkatalog	Gewichtung Rubrik	Gewichtung Oberkriterium	Gewichtung Unterkriterium	Anteil des Unterkriteriums am Gesamtanzahlwert	ARIS-Toolset Version 3.1 Demo		BONAPART Version 2.0 Demo		Nautilus Version 1.2 beta		VISIO Version 4.0 Demo	
					Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen
4 Kriterien der Ablauforganisation	20	20,00										
41 Kanten / Relationen	3	1,00										
411 Systemdefinierte Kantentypen (Relationen) stehen dem Anwender zur Verfügung		0,250	0,750		6	4,50	3	2,25	4	3,00	0	0,00
412 Erstellung benutzerdefinierter Kantentypen (Relationen) möglich		0,400	1,200		0	0,00	6	7,20	6	7,20	0	0,00
413 Benutzerdefinierte Relationen werden vom System auf Konsistenz überprüft		0,300	0,900		0	0,00	6	5,40	0	0,00	0	0,00
414 Beschriftung der Kanten kann wahlweise ein- und ausgeblendet werden		0,050	0,150		6	0,90	6	0,90	0	0,00	0	0,00
42 Rechtemanagement	5	1,00										
421 MA können Rechte zugeordnet werden, die es ihnen ermöglichen Prozesse innerhalb ihres Kompetenzbereichs eigenständig zu ändern		1,000	5,000		6	30,00	2	10,00	1	5,00	0	0,00
43 Modelle	3	1,00										
431 Referenzmodelle werden angeboten		0,300	0,900		6	5,40	0	0,00	4	3,60	0	0,00
432 Das Metamodell der Prozesse ist vorgegeben		0,200	0,600		6	3,60	3	1,80	6	3,60	0	0,00
433 Benutzerdefinierte Anpassung des Metamodells		0,500	1,500		0	0,00	6	9,00	6	9,00	0	0,00
44 Benutzerfreundlichkeit / Flexibilität	5	1,00										
441 Intuitive Symbole für Prozesse, Teilprozesse, Sachmittel etc. werden verwendet		0,200	1,000		3	3,00	4	4,00	6	6,00	5	5,00
442 Ad hoc Änderungen sind ohne großen Aufwand schnell integrierbar		0,200	1,000		2	2,00	2	2,00	6	6,00	5	5,00
443 Grafische Aufbereitung der Ablaufstruktur		0,100	0,500		5	2,50	5	2,50	5	2,50	6	3,00
444 Unterschiedliche Beschreibungsmethoden für Prozesse (Prozesse aus verschiedenen Sichten anzeigen lassen)		0,050	0,250		6	1,50	1	0,25	5	1,25	1	0,25
445 Einmal modellierte Standardprozesse werden in einer Strukturbibliothek verwaltet		0,200	1,000		6	6,00	1	1,00	6	6,00	1	1,00
446 Automatische Dokumentation des modellierten Prozesses		0,100	0,500		6	3,00	3	1,50	6	3,00	0	0,00
447 Automatische Generierung der Ablaufstruktur aus dem Prozeß heraus		0,150	0,750		0	0,00	0	0,00	6	4,50	0	0,00
45 Analyse	2,5	1,00										
451 Analytierte Prozesse können aus verschiedenen Sichten ausgewertet werden		0,200	0,500		4	2,00	3	1,50	6	3,00	0	0,00
452 Analytierte Prozesse können anhand von Protokollen dokumentiert werden		0,100	0,250		6	1,50	6	1,50	5	1,25	0	0,00
453 Es können Gestaltungsempfehlungen im Hinblick auf eine bessere Prozeß-Struktur abgeleitet werden		0,350	0,875		5	4,38	4	3,50	5	4,38	0	0,00
454 Modellierte individuelle Prozesse werden vom System mit Referenzprozessen verglichen		0,350	0,875		3	2,63	0	0,00	1	0,88	0	0,00
46 Simulation	1,5	1,00										
461 Simulationsmöglichkeiten		0,300	0,450		5	2,25	6	2,70	0	0,00	0	0,00
462 Simulationsergebnisse werden grafisch aufbereitet		0,200	0,300		2	0,60	4	1,20	0	0,00	0	0,00
463 Vergleich von verschiedenen Simulationsläufen möglich		0,300	0,450		3	1,35	0	0,00	0	0,00	0	0,00
464 Historie der einzelnen Simulationsläufen möglich		0,200	0,300		3	0,90	1	0,30	0	0,00	0	0,00
Nutzen der Rubrik GESAMT						78,00		58,50		70,15		14,25
Erfüllungsgrad in der Rubrik						65,00%		48,75%		58,46%		11,88%
Anteil am Gesamtnutzen des Werkzeugs						19,36%		18,68%		23,14%		13,41%

Kriterienkatalog

	Gewichtung Rubrik	Gewichtung Oberkriterium	Gewichtung Unterkriterium	Anteil des Unterkriteriums am Gesamtsummenwert	ARIS-Toolset Version 3.1 Demo		BONAPART Version 2.0 Demo		Nautilus Version 1.2 beta		VISIO Version 4.0 Demo	
					Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen
5 Zusammenspiel mit operativen Umgebungen und Potentiale	25	25,00										
<i>51 Workflowanbindung</i>		12,5	1,00									
511 Build-Time-Komponente um Prozesse in Workflow Management Systemen (WFMS) zur Laufzeit zu bringen			0,200	2,500	6	15,00	6	15,00	6	15,00	0	0,00
512 Schnittstelle(n) zu Run-Timeversionen von WFMS			0,150	1,875	4	7,50	2	3,75	0	0,00	0	0,00
513 Schnittstelle(n) zu WFMS erfüllt die Kriterien der Workflow Management Coalition (WfMC)			0,050	0,625	3	1,88	6	3,75	0	0,00	0	0,00
514 Mit dem Werkzeug modellierte Strukturen und Prozesse können 1:1 in ein WFMS übertragen werden			0,300	3,750	6	22,50	3	11,25	0	0,00	0	0,00
515 Dynamischer Datenaustausch zwischen BPR-Werkzeug und Workflow-Tool			0,300	3,750	6	22,50	3	11,25	0	0,00	0	0,00
<i>52 Unterstützungspotential zur Integration und Auswahl von Software</i>		12,5	1,00									
521 Referenzmodelle			0,100	1,250	5	6,25	0	0,00	3	3,75	0	0,00
522 Vergleich der unternehmensspezifischen Modelle mit Referenzmodellen			0,200	2,500	2	5,00	0	0,00	1	2,50	0	0,00
523 Änderungen der Referenzmodelle bewirken Änderungen in der Software ("lernende Software")			0,300	3,750	0	0,00	0	0,00	0	0,00	0	0,00
524 Darstellbarkeit der Änderungen zwischen beiden Modellen			0,200	2,500	0	0,00	0	0,00	0	0,00	0	0,00
525 Anforderungen an Individualprogrammierung werden aufgezeigt			0,200	2,500	6	15,00	0	0,00	0	0,00	0	0,00
<i>Nutzen der Rubrik GESAMT</i>						95,63		45,00		21,25		0,00
<i>Erfüllungsgrad in der Rubrik</i>						63,75%		30,00%		14,17%		0,00%
<i>Anteil am Gesamtnutzen des Werkzeugs</i>						23,73%		14,37%		7,01%		0,00%
6 Ergänzende Features	5	5,00										
<i>61 Projektmanagement</i>		2	1,00									
611 Inhaltliche Dokumentation des Projektes			0,250	0,500	3	1,50	3	1,50	6	3,00	0	0,00
612 Unternehmensinterne Projektgruppenbildung möglich			0,250	0,500	6	3,00	4	2,00	6	3,00	3	1,50
613 Unternehmensübergreifende Projektgruppenbildung möglich			0,250	0,500	4	2,00	3	1,50	1	0,50	1	0,50
614 Abfrage von Informationen zu abgeschlossenen Projekten möglich			0,250	0,500	4	2,00	1	0,50	5	2,50	3	1,50
<i>62 Prozeßkostenrechnung</i>		1	1,00									
621			1,000	1,000	4	4,00	5	5,00	3	3,00	1	1,00
<i>63 Integration anderer Anwendungen</i>		2	1,00									
631 Integration funktionsneutraler Programme (z.B. Excel, Word, Lotus 1-2-3, Lotus Ami Pro etc.) Schnittstelle zu Grafiktools			0,00	0,00		400		400		400		400
632 Direkter Internetzugang möglich			0,100	0,200	3	0,60	3	2,40	0	0,00	0	0,00
634 Anbindung an ein Mailingsystem			0,100	0,200	3	0,60	0	0,00	6	1,20	0	0,00
<i>Nutzen der Rubrik GESAMT</i>						20,10		16,90		18,40		0
<i>Erfüllungsgrad in der Rubrik</i>						67,00%		56,33%		61,33%		28,33%
<i>Anteil am Gesamtnutzen des Werkzeugs</i>						4,99%		5,40%		6,07%		8,00%

Kriterienkatalog	Gewichtung Rubrik	Gewichtung Oberkriterium	Gewichtung Unterkriterium	Anteil des Unterkriteriums am Gesamtnutzenwert	ARIS-Toolset Version 3.1 Demo		BONAPART Version 2.0 Demo		Nautilus Version 1.2 beta		VISIO Version 4.0 Demo	
					Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen	Teilnutzen	Gewichteter Teilnutzen
7 Kriterien für Datenmanagement	5	5,00										
71 Organisationsdatenbank		3,5	1,00									
711 Dem Werkzeug ist eine eigene Organisations-Datenbank unterlegt			0,250	0,875	6	5,25	0	0,00	0	0,00	0	0,00
712 Der Zugriff auf die Organisationsdaten erfolgt extern über eine Schnittstelle z.B. ODBC			0,150	0,525	6	3,15	6	3,15	6	3,15	0	0,00
713 Verwendung der Organisationsdatenbank in anderen Applikationen ist möglich			0,600	2,100	6	12,60	6	12,60	6	12,60	0	0,00
72 Datenschutz und -sicherung		1,5	1,00									
721 Berechtigungskontrolle und Zugriffsschutz auf Serverebene			0,200	0,300	6	1,80	6	1,80	0	0,00	6	1,80
722 Berechtigungskontrolle und Zugriffsschutz auf Benutzerebene			0,200	0,300	6	1,80	4	1,20	6	1,80	0	0,00
723 Berechtigungskontrolle und Zugriffsschutz auf Dateiebene			0,200	0,300	5	1,50	3	0,90	0	0,00	0	0,00
724 Protokollierung unbefugter Zugriffe			0,100	0,150	4	0,60	0	0,00	0	0,00	0	0,00
725 Datensicherungsprotokolle vorhanden			0,250	0,375	6	2,25	0	0,00	6	2,25	0	0,00
726 Kopierschutz			0,050	0,075	6	0,45	6	0,45	6	0,45	3	0,23
<i>Nutzen der Rubrik GESAMT</i>						29,40		20,10		20,25		2,03
<i>Erfüllungsgrad in der Rubrik</i>						98,00%		67,00%		67,50%		6,75%
<i>Anteil am Gesamtnutzen des Werkzeugs</i>						7,30%		6,3%		6,68%		1,91%
	100,00	100,00		100,00								
GESAMTNUTZEN						402,90		313,18		303,21		106,28
RANGFOLGE						1		2		3		4
<i>Gesamterfüllungsgrad gemessen am Gesamtnutzen</i>						67,15%		52,20%		50,54%		17,71%

Table B-9: Criteria of the benefit analysis and valuation

Chapter C

Introduction to the X.500 Standard Recommendations

This chapter is an introduction to the X.500 standard recommendations of the ITU-T and to LDAP. It aims to describe them in sufficient detail to allow for an understanding of references made to X.500 and LDAP in the GroupOrga project. It is not meant to be a complete reference for the implementation of an X.500- or LDAP-based solution, but should give an impression of structure and content of the recommendations as well as an introduction to its concepts. This chapter's content mainly falls back upon a documentation of a subproject on X.500 and LDAP within the GroupOrga research ([Heinz/Ott 1997]).

C.1 The Structure of the X.500 Standards

What is often called "X.500" or "The X.500 Standard" is actually a set of recommendations developed jointly by the ISO (International Standards Organization) and the former CCITT (International Telegraph and Telephone Consultative Committee), now called ITU-T (International Telecommunication Union-Telecommunication Standardization Bureau). In the "Joint ISO/CCITT working group on Directories", ISO is represented primarily by representatives from leading computer vendors and ITU-T by national, public and private telecommunication service providers. The studies commenced in 1984 and resulted in the first version of the X.500 Standard in 1988. In the following years, extensions were developed concerning primarily replication and security issues. In 1993, the current version of the recommendations was released which presents the base for GroupOrga developments and which is meant whenever this chapter refers to the *X.500 standard*. Often, the 1988 version of the standards is still referred to as *The Standard*, and the latest version as the *1993 extension* or the *1993 edition*.

The 1993 extension of the X.500 standard comprises the following recommendations:

- ❑ X.500 - The Directory: Overview of Concepts, Models and Services (ISO 9594-1)
- ❑ X.501 - The Directory: Models (ISO 9594-2)
- ❑ X.509 - The Directory: Authentication Framework (ISO 9594-8)
- ❑ X.511 - The Directory: Abstract Service Definition (ISO 9594-3)
- ❑ X.518 - The Directory: Procedures for Distributed Operations (ISO 9594-4)
- ❑ X.519 - The Directory: Protocol Specifications (ISO 9594-5)
- ❑ X.520 - The Directory: Selected Attribute Types (ISO 9594-6)
- ❑ X.521 - The Directory: Selected Object Classes (ISO 9594-7)
- ❑ X.525 - The Directory: Replication (ISO 9594-9)

The following sections outline the contents of the recommendations that are relevant in the context of the GroupOrga project. Since the different standards strongly interact, they are not presented one after another but are introduced along the lines of relevant topics.

C.2 The Directory Model

The X.500 standard defines a directory as a repository of information. This repository is called *directory information base* (DIB). The DIB stores information about the various entities that exist in the real world and are relevant for directory users. They access the directory via *directory user agents* (DUA) connecting them to server components called *directory service agents* (DSA). The entirety of all directory service agents holds and manages all directory information, thus the whole directory information base. Figure C-1 highlights this encapsulation and visualizes the interaction between DIB, DSA and DUA.

The directory information base, and therefore the directory service agents, are structured hierarchically. The hierarchical representation of the directory information is the *directory information tree*. Structure and content of the DIB are further explained in section C.3.

The standard also deals with the problems of ensuring consistency of the DIB structure and its contents by specifying a set of rules called the directory schema. Section C.7 gives an overview of the directory schema.

Furthermore, the X.500 standard defines a communication protocol for interaction between directory user agents and directory service agents called the directory access protocol. It also establishes protocols for the communication between different directory service agents. Three protocols are distinguished: the *directory service protocol* (DSP), the *directory information shadowing protocol* (DISP) and the *directory operational binding management* (DOP). Section C.5 will discuss these protocols in more detail.

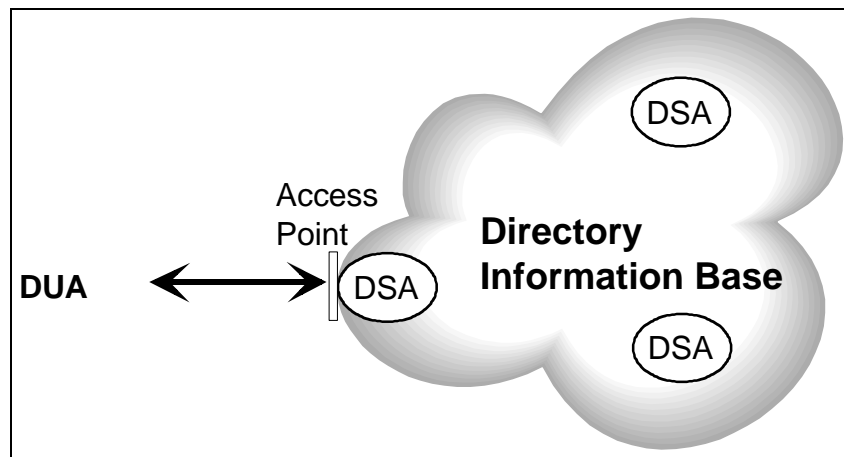


Figure C-1: Directory structure

C.3 The Information Model

The information model, described in X.501, defines how directory information is organized. It is divided into several sub models each of which provides a particular view on the directory. The *directory user information model*, which was already introduced in the 1988 standards, describes the user's prospect. It defines the way information is named and identified. The second building block, the *directory operational and administrative information model*, describes the administrator's view of a directory. It defines the concepts required to administer a local directory.

A third model, the *directory administrative authority model*, defines how the responsibility for administering specific parts of the directory can be shared between different authorities. It deals with methods of splitting up the directory information into several parts. This last model described in the standard is the *DSA information model*. It presents the view of a directory service agent and deals with the management of the distributed directory information. Each DSA holds a piece of the directory and communicates with other DSAs to handle user requests which cannot be resolved autonomously. The DSA information model specifies the information items that DSAs need to communicate with each other.

In the following sections the first two sub-models are explained in more detail, whereas the latter two are briefly outlined in section C.4.

C.3.1 Directory User Information Model

The directory user information model defines the structure of the information within the directory information base. It comprises directory entries, their attributes, the directory information tree, name conventions and rules for matching different entries with each other.

C.3.1.1 Directory Entries

The smallest information unit stored in a directory is called an entry. The standard distinguishes three different types of entries: object entries, alias entries and subentries.

Object entries

Object entries represent objects of the real world, that are of interest for a directory, e.g. persons, groups of persons or devices. A real world entity does not necessarily need to be represented by exactly one object; it may also be described by several different objects. Thus, persons may be represented as users of the organization's information systems with attributes such as *user name*, *password* and *mail address*, and additionally as employees described with attributes such as *office location* and *phone number*, *position* and *salary*. Vice versa, one object can represent more than one real world entity. The object "Project Group X.500 Implementation" may for example represent all members of this group.

Alias entries

However, each object entry represents exactly one object of interest, being the primary collection of information about that object in the DIB. In addition to the object entry, there may be an arbitrary number of alias entries for a real world entity. These aliases have a different identifier and only hold a pointer to the object entry they refer to, i.e. its name.

Subentries

Subentries are collections of object and alias entries. They serve administrative purposes and are explained in section C.4.1 in more detail.

C.3.1.2 Attributes

Directory entries are characterized by their attributes. An attribute contains an attribute type and one or more attribute values that comply to a specified syntax. Some attribute types are predefined in the standard recommendation X.520. Nevertheless, the standard allows administrators to define custom attribute types if necessary. The definition of a new attribute type must contain a unique identifier and a syntax definition for its values. Furthermore, it has to be defined whether attributes of the new type are single- or multi-valued.

Attribute types can be structured hierarchically using generic types at the top of the hierarchy that may be gradually refined towards more specific types. Therefore, attribute values can be accessed via their specific type (direct reference) or by a superordinated type (indirect reference). An example of a hierarchical attribute type is the office phone number, that may be a subtype of the company's central phone number.

C.3.1.3 Object Classes

Objects that share certain characteristics, i.e. have common attributes, are aggregated in an object class. The object class defines the attributes that an entry of that class must contain.

There are three different kinds of object classes defined in the standard:

- ❑ *Abstract object classes* are used to inherit characteristics to other object classes, e.g. the particular object class *Top*. An entry should not belong only to abstract object classes.
- ❑ *Structured object classes* are used to represent entries in the DIT. The structured object class of an entry is one of its most important characteristics. The membership of an entry to a structured object class should be permanent.
- ❑ *Auxiliary object classes* may be used to define flexible attributes for an entry. The membership of an entry to auxiliary object classes may change frequently.

The standard defines a couple of object classes that may be used to model an organization directory according to its recommendation X.521. As with attribute types, the standard provides means for administrators to create customized object classes. The specification of a new object class must contain a unique object identifier, the inheritance tree, and a list of all mandatory and optional attributes. All classes are organized in an inheritance hierarchy, where subclasses inherit all attributes of their superclasses. The particular object class *Top* is defined as the root class of all other standard or custom object classes.

As examples of customized object classes the following object classes have been added to extend the DIT structure defined in [ISO/IEC 1993b] in order to match the standard with the GroupOrga GEIMM. The semantics of the contained attribute types are defined in the comparison in section 5.3.4. Technical details such as ASN.1 syntax and matching rules are not further specified. Figure C-2 to Figure C-6 show the definitions of such custom object classes.

```

organizationalPosition OBJECT-CLASS ::= {
  SUBCLASS OF      { top }
  MUST CONTAIN    { commonName }
  MAY CONTAIN     {description |
                  requiredSkills |
                  connectedResponsibilities |
                  owner }
ID                id-oc-organizationalPosition }

```

Figure C-2: Custom object class organizationalPosition

```

softwareAgent OBJECT-CLASS ::= {
  SUBCLASS OF      { top }
  MUST CONTAIN    { commonName }
  MAY CONTAIN     {description |
                  executionTime|
                  commandLineParameter}
ID                id-oc-softwareAgent }

```

Figure C-3: Custom object class softwareAgent

```

substitutionRule OBJECT-CLASS ::= {
  SUBCLASS OF    { top }
  MUST CONTAIN  { commonName }
  MAY CONTAIN   {description |
                validTime|
                rule}
  ID            id-oc-substitutionRule }
    
```

Figure C-4: Custom object class substitutionRule

```

informationObjectReference OBJECT-CLASS ::= {
  SUBCLASS OF    { applicationProcess }
  MUST CONTAIN  { commonName }
  MAY CONTAIN   { description |
                location |
                accessMethod |
                commandLineParameter}
  ID            id-oc-informationObjectReference }
    
```

Figure C-5: Custom object class informationObjectReference

```

embeddedInformationObject OBJECT-CLASS ::= {
  SUBCLASS OF    { applicationProcess }
  MUST CONTAIN  { commonName }
  MAY CONTAIN   { description |
                informationObject}
  ID            id-oc-embeddedInformationObject }
    
```

Figure C-6: Custom object class embeddedInformationObject

C.3.1.4 Directory Information Tree

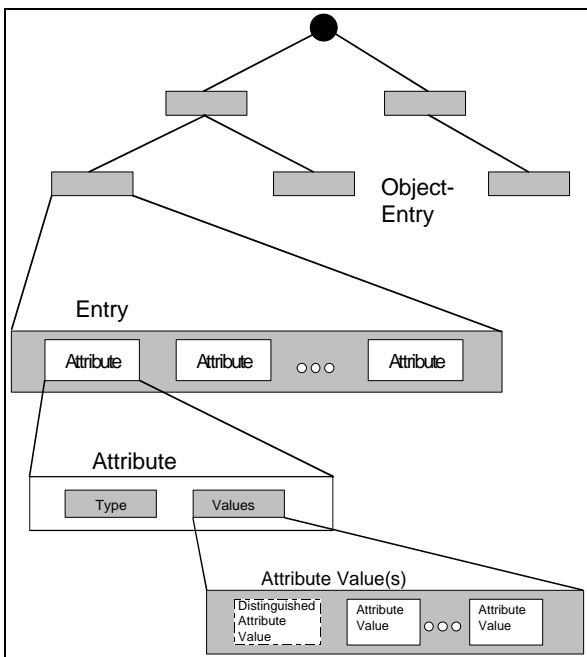


Figure C-7: Structure of entities in the DIT

All directory entries are organized in a hierarchy. The logical structure of the DIB is referred to as the *directory information tree* (DIT). Each entry of the DIB has a corresponding node in the DIT. All nodes comprise one immediate superior and one or more immediate subordinates. The root of the DIT has no real world counterpart, it exists only for completeness purposes. Entries at the first level, i.e. the direct subordinates of the root, usually represent countries or multinational organizations, followed by national organizations and their departments. The leaf entries represent people or devices. Figure C-7 shows hierarchy and attributes in the DIT.

C.3.1.5 Names

Each directory entry must have a unique *relative distinguished name* (RDN), that uniquely distinguishes the entry from its peers, i.e. those entries that have the same immediate superiors. The RDN can be set up either by a single attribute, like {Organization = University of Paderborn}, or as well by a whole set of attributes like {Organization = University of Paderborn, Location = Meschede}.

The entries are globally identified by their *distinguished name* (DN) which contains the RDN of its superior entries and its own (e.g. {Country = DE, (Organization = University of Paderborn, Location = Meschede)}).

C.3.1.6 Matching Rules

The directory aims to solve queries from its users by applying so-called matching rules. When a user searches for a particular entry, the system traces the DIT, beginning at the root and following the path specified in the DN. When it finds an alias entry, it replaces the alias name by the name the alias is pointing to (de-referencing). It then starts the search again at the top, this time using the real name. This way of searching an entry by tracing the DIT is called *name resolution*.

Matching rules are used for name resolution as well as search and compare operations. They solve problems such as *is "Tom" equal "TOM"*, *is "(05251) 600" the same as "05251-600"* or *is "5" smaller or equal than "10"*.

X.520 defines a large set of matching rules. In general, there are four types of matching rules that are applicable for most attribute types:

- *Present*: checks the presence of an attribute of a specified type irrespective of its value.
- *Equality*: checks if an entry has an attribute of which the type and value are equal to the input value.
- *Substrings*: checks if an entry has an attribute of the specified type of which the value contains the input value.
- *Ordering*: checks if an entry has an attribute of the specified type of which the value is e.g. greater than, equal to, or less than the input value.

Furthermore, the standard allows for the definition of new matching rules and specifies requirements that have to be met by these rules. Custom matching rules may be especially useful when defining new attribute types.

C.3.2 Directory Operational and Administrative Information Model

This part of the information model deals with data that is of particular importance for administrators, whereas it may not be relevant for users of the directory.

Directory operational attributes

The standard distinguishes between two types of attributes: user attributes and directory operational attributes. Whereas the user attributes hold information relevant for the users of the directory, i.e. all data that describes the entries like name, location and so on, the directory operational attributes have administrative purposes and thus are part of the directory operational and administrative information model. These attributes include for instance access control information or time stamps indicating prior modifications of an entry. By default, users do not see these attributes when browsing directory entries, however, the information can be made available to administrators.

Subtrees

The DIT can be divided into *subtrees*. A subtree is a collection of entries that, after being grouped together, can be administered as a single unit. This is useful when entries belonging to a subtree contain *collective attributes* that hold the same value such as the postal address of all employees working in one department. A subtree definition consists of:

- The *base* specifying the root node of the subtree
- The *chop* defining the scope of the subtree
- The *specification filter* allowing to select particular entries out of the chop as members of the subtree

A sample DIT is displayed in Figure C-8. The subtree enclosed by the shaded rectangle is defined by the base "Company X", the chop "1 level" and the filter "All, but department Z".

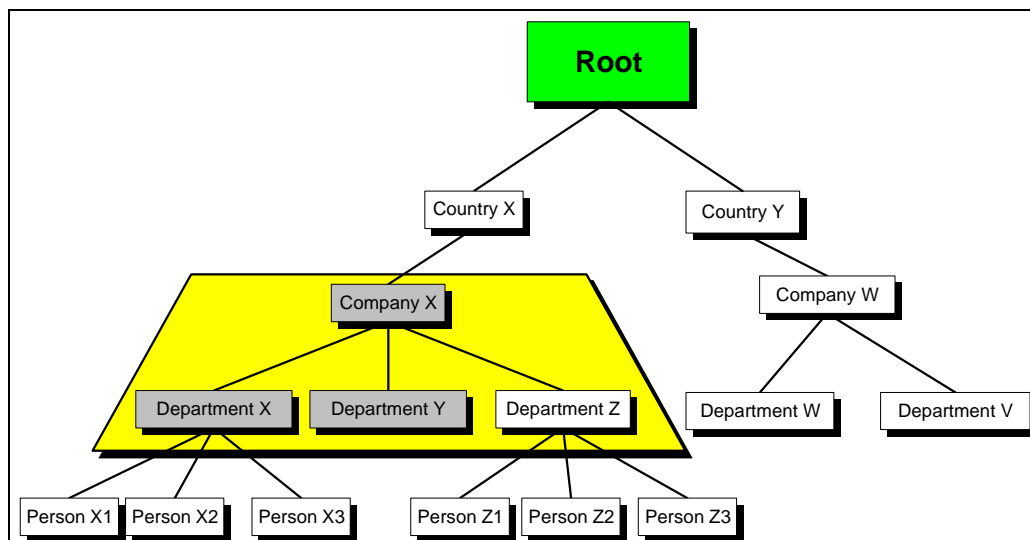


Figure C-8: Sample DIT subtree specification

C.4 Distribution of the Directory

Concerning the distribution of directory information, the standard follows two principles:

- ❑ The directory is presented to the user as a single unit
- ❑ Internally, the directory information is distributed over many different autonomous DSAs

These principles are explained in two models, the directory administrative authority model and the DSA information model. A further important aspect of distribution is replication of directory information. These issues are discussed in this section.

C.4.1 Directory Administrative Authority Model

The directory administrative authority model pays particular attention to the distributed administration of the directory. It defines means for building collections of entries that can be administered autonomously by different authorities.

Specific administrative authorities

Authorities have to fulfill different tasks for administrating their part of the DIT:

- ❑ *Naming administration*, i.e. defining the entries' names and name structure
- ❑ *Subschema administration*, i.e. defining the subschema, for example the custom definition of attribute types
- ❑ *Security administration*, e.g. the access control management
- ❑ *Collective attribute administration*, i.e. defining attributes shared between all entries belonging to an area, such as the central telephone number or the address of a company.

Administrative areas and administrative points

The DIT is divided into separate *autonomous administrative areas* (AAA) each of which is administered by exactly one administrative authority. Every AAA starts at an *autonomous administrative point* (AAP) that is represented by an administrative entry in the DIT and it ends either at a leaf entry or at the next AAP. Nested within an AAA, one or more so called *inner administrative areas* (IAA) may exist, each starting at an *inner administrative point* (IAP) within the wrapping AAA and ending at the end of the AAA.

Each AAA consists of three layers of *specific administrative areas*:

- ❑ Subschema administration area
- ❑ Access control administration area
- ❑ Collective attribute administration area

The different areas represent the tasks an administrative authority has to provide.

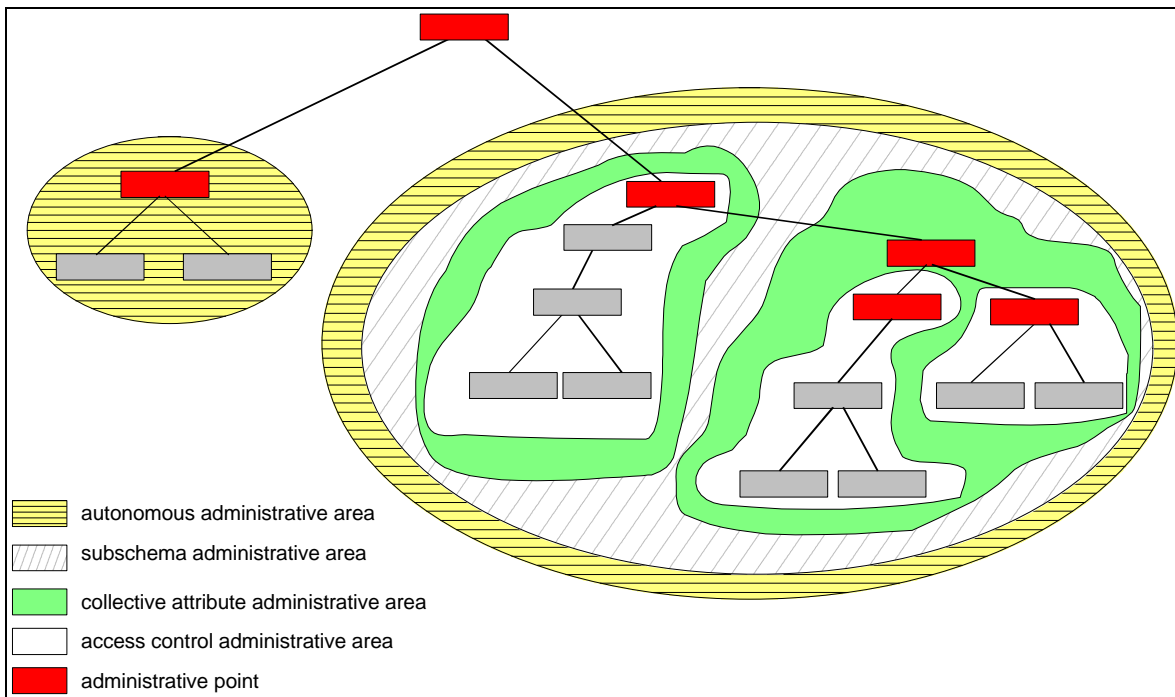


Figure C-9: Administrative areas

Figure C-9 shows how the different specific administrative areas are embedded in autonomous administrative areas.

Similar to AAAs, inner administrative areas and specific administrative areas start at a specific entry in the DIT. For IAAs this is an inner administrative point, and for SAAs it is called *specific administrative point*. All administrative points are distinguished by the *administrativeRole* attribute assigned to their corresponding entry. This attribute defines the type of the administrative area by holding one of the following values:

- `autonomousArea`,
- `accessControlSpecificArea`,
- `accessControlInnerArea`,
- `subschemaAdminSpecificArea`,
- `collectiveAttributeSpecificArea`
- `collectiveAttributeInnerArea`

Subentries

For administration purposes, the standard defines a special type of entry, the so called subentry. Subentries are usually immediate subordinates of administrative entries and hold information about the administrative area beneath its administrative entry. Subentries themselves have no subordinates. Aside from attributes containing the name of the subentry

and the specification of the subtree it masters, each subentry contains the details about the specific administrative area. This might be the complete subschema for the area, the whole set of collective attributes or the access control information, depending on what type of area it masters. Details about content and format of this information can be found in section C.7.6.

C.4.2 DSA Information Model

The models of the directory discussed so far consider the directory one unit. With a concept of autonomous authorities, the last and most detailed view of the directory breaks down its structure to actual application processes that administer directory information. These application processes are called *directory system agents* (DSA). They provide the interface, which are the access points for directory users (represented by directory user agents, DUAs). In order to answer user requests different DSAs have to cooperate. The DSAs communicate with each other as well as with DUAs via OSI directory protocols that are described in section C.5. Similar to the directory entries, the DSAs are hierarchically organized, i.e. each DSA has exactly one superior DSA and an arbitrary number of subsequent DSAs (this does not apply to the highest level).

Each DSA holds a part of the directory information base (DIB), its *DIB fragment*. All DIB fragments are disjoint, i.e. they cover the whole DIB and no entry is contained in two or more DIB fragments. A fragment contains one or more disjoint subtrees, the *naming contexts*. All entries of these naming contexts have a common part of their distinguished names which is the distinguished name of the root entry of the subtree. This DN is the *context prefix* of the naming context. An example of such a DSA hierarchy including different naming contexts is shown in Figure C-10.

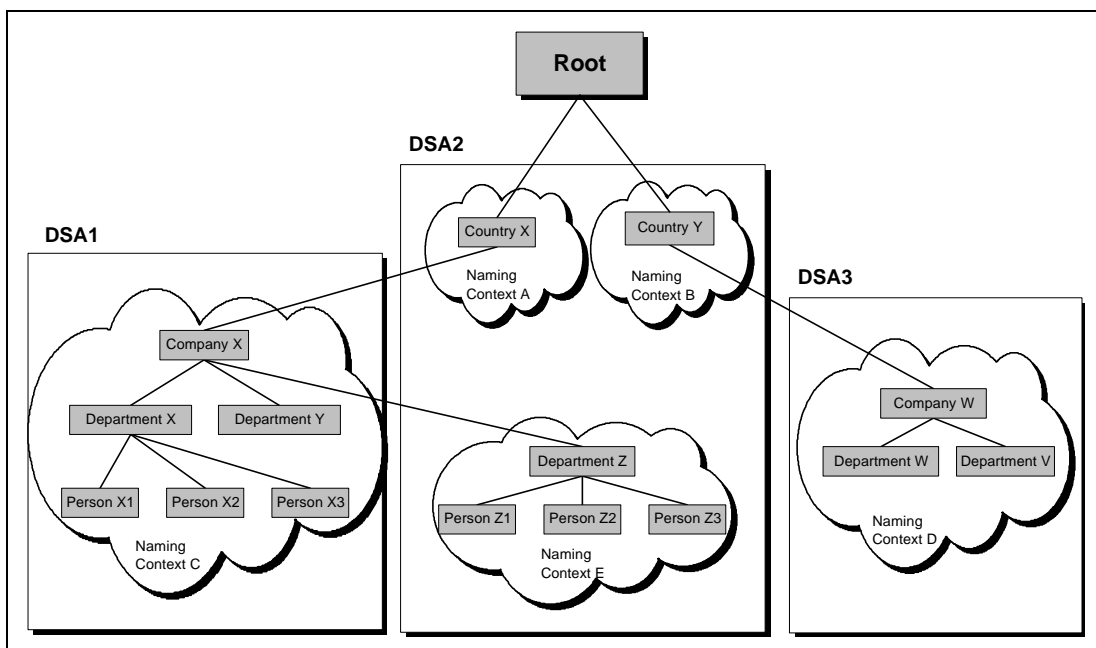


Figure C-10: DSA structure

The DSA information model describes the directory from a DSA's point of view. It deals with information that DSAs need to have to serve requests in distributed environments. The following two concepts describe the content of this information, called *knowledge* and the way it is stored in the DIT in form of *DSA specific entries* and *DSA operational attributes*.

Knowledge

When a user requests information that is not held by the DSA the user is currently connected to (referred to as the *home DSA*), the request has to be forwarded seamlessly to the appropriate DSA. In order to achieve this, all DSAs have to know which part of the DIB is mastered by which DSA. This information is referred to as *knowledge* that consists of *knowledge references*. Knowledge information can be separated into *master knowledge* and *shadow knowledge*. Master knowledge contains the access point of the master DSA for a naming context, i.e. it helps to find the DSA that holds the entries belonging to an organization or an organizational unit for example. Shadow knowledge references point to DSAs that hold replicated directory information. Details about the replication of directory information can be found in section C.4.4.

Knowledge references are classified into several reference types depending on the information they hold. These knowledge reference types can be found in Table C-1.

Reference Type	Description
superior reference	Contains the access point of any DSA that holds a naming context that is located further up the tree than the context held by the current DSA.
immediate superior reference	Contains the context prefix of the immediate superior naming context as well as the access point of the DSA that masters that naming context.
subordinate references	Contain the context prefix of an immediate subordinate naming context as well as the access point of the DSA that masters that naming context. All immediate subordinates are represented by a subordinate reference, and thus each DSA contains complete information about its subordinates.
non-specific subordinate references (optional)	Contain only the access point of a DSA that holds any subordinate naming context. The context prefix is not stored (maybe because it is not known).
cross references (optional)	Contain any content prefix and the access point of the appropriate DSA. Cross references can improve performance of name resolution.
supplier references	Are only held by shadow consumer DSAs. Contain the access point of the shadow supplier DSA and additional information about the shadowing agreement.
consumer references	Are only held by shadow supplier DSAs. Contain the access point of the shadow consumer DSA and additional information about the shadowing agreement.

Table C-1: Knowledge reference types

Using this knowledge, a DSA can pass a request further up the DIT to its immediate superior or further down to the correct subordinate DSA. Details about the name resolution in a distributed environment can be found below.

DSA specific entries (DSEs)

The DSA information model defines a new type of directory entries to store knowledge information: *DSA specific entries*. DSA specific entries consist of attributes containing knowledge information and optional directory entries holding user and operational attributes. DSEs that only store knowledge information are for example used for subordinate references, whereas DSEs, also containing operational and user attributes, are used to represent actual directory entries.

The reason for the invention of a new entry type is the necessity to store information about one single entry in two or more locations when regarding the physical representation of the DIT within DSAs. This is for example helpful when the information about an object is stored in a different DSA. In this case both DSEs, the one containing the reference information and the one containing the actual entry, carry the same DIT name, but the name of the directory entry itself is still unambiguous.

The collection of entries a DSA masters is referred to as its DSA information tree. Each DSA information tree starts at the root of the DIT. That means that each DSA has to store the chain of DSEs beginning at the top and reaching until the first DSE that is actually mastered by the DSA. Figure C-11 gives an overview about the structure of a DSE and the parts that are relevant for the different directory information models.

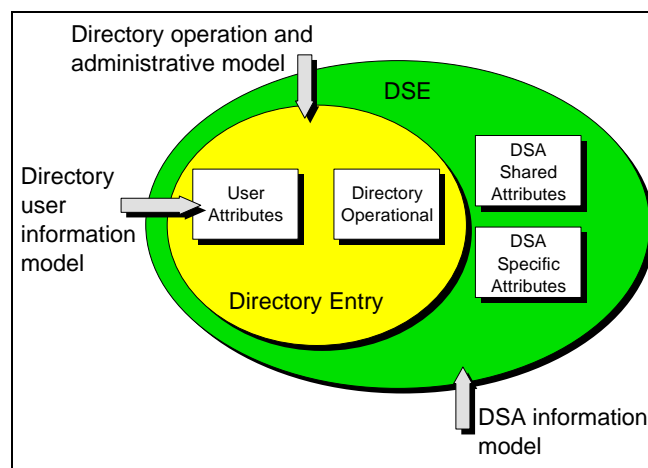


Figure C-11: DSEs and the different directory models

The knowledge attributes contained in a DSE are distinguished in DSA shared attributes and DSA specific attributes. The DSA shared attributes of a DSE contain the same values, irrespective of which DSA the DSE is stored on. An example for a DSA shared attribute is the information about the DSA that masters the corresponding entry. This information is not dependent on the location of the DSE. The values of the DSA specific attributes of an DSE

may vary depending on the DSA that stores the DSE. An example is the shadow consumer information, because entries may be shadowed over several levels, and thus different copies of an entry may be shadowed to different consumers. Figure C-15 in section C.4.4 shows an example for this scenario.

The way knowledge information is represented by attributes and attribute values is not described here. Information on this can be found in [ISO/IEC 1993a] (chapter 18) and [Chadwick 1994] (chapter 9).

C.4.3 Managing User Requests in the Distributed Directory

In a distributed directory, a user may request information about an entry that is not stored in the DSA with which the user is connected (referred to as the home DSA). For this scenario the standard defines two possible ways the home DSA can proceed: *chaining* and *referral*.

Chaining

Chaining is the more user friendly way to handle the request. The home DSA searches its knowledge information for another DSA that might be able to answer the request, and forwards it to this DSA. The next DSA proceeds in the same way and so the request is resolved by a chain of DSAs. Each of them aims to supply the requested information and passes it back to the predecessor in the chain. Hence, after the request has been answered completely, the result is not passed to the user directly, but it will follow the chain back to the originator. In Figure C-12, DSA A chains a request to DSA B, which can answer it. It sends the result back to DSA A, which sends it to the DUA. If the home DSA was able to solve the request partially it would add the interim result.

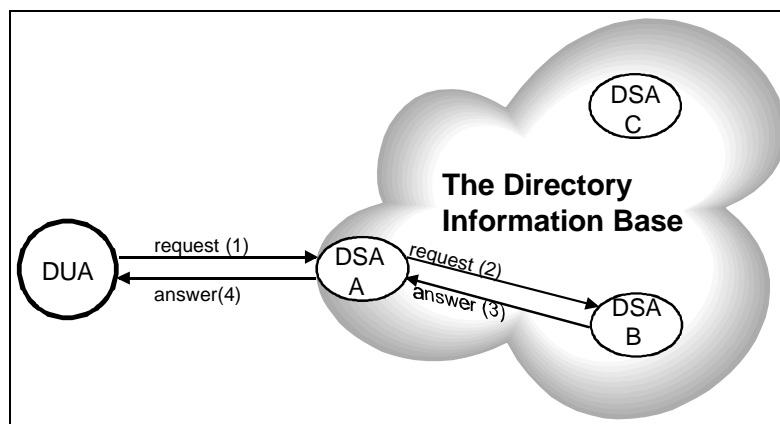


Figure C-12: Chaining

Referral

In case a home DSA is not configured to allow chaining or not able to chain a request, it refers the user to a DSA that might then be able to solve the request directly. Figure C-13 shows an example for such a referral. In order to obtain the result, the user has to connect to this other DSA and repeat the request. This approach is called *referral*. Referrals destroy the users'

imagination of a single directory that stores all information as they reveal the location of a specific entry. Users are confronted with different units that hold fragments of the DIT. This might have disadvantages such as an additional configuration effort for the client since another access point has to be contacted. However, there are also benefits (cp. [Chadwick 1994]). The performance might improve since the user connects directly to the relevant DSA and furthermore under security aspects it may be preferred to connect directly to a DSA instead of sending the requests via several other DSAs.

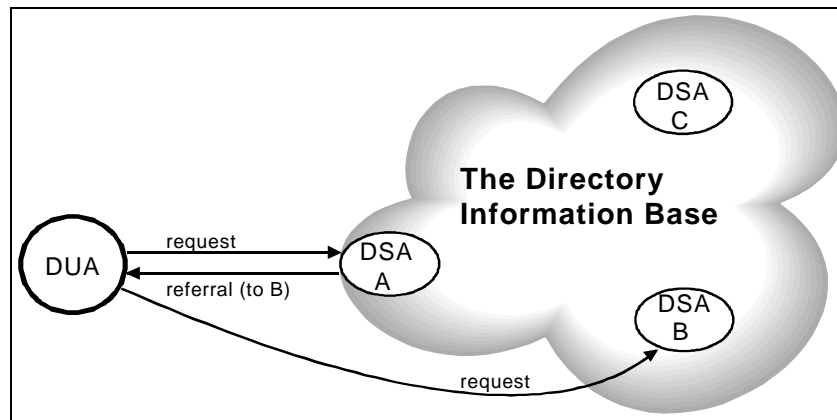


Figure C-13: Referral

Distributed name resolution

Distributed name resolution is the method DSAs use to find a path to the entry the user requested information about. It makes use of the knowledge information described above such as superior, subordinate and cross references.

Firstly, the DSA compares the distinguished name of the requested entry with its naming context that is either the beginning of the DN or not. When the requested DN starts with the naming context, the entry is either held by the DSA itself or by one of its subordinates. If the entry is not part of the DSA information tree, the request is chained to the subordinate whose naming context is part of the DN of the requested entry. If the naming context is not part of the DN, the entry cannot be found further down the DIT. In this case, the DSA chains the request to a superior DSA. The next DSA proceeds the same way until the master DSA of the requested entry is found.

In the example in Figure C-14 a user connected to DSA 1 requests information about Person X1 in Company X. DSA 1 cannot answer this request and passes it on to DSA 2, using a superior reference. From there, it is passed on by using subordinate references to DSA 3 and finally to DSA 4. DSA 4 answers the request and the information follows the same way back. If DSA 1 had stored a cross reference to DSA 4, it would have passed the request directly.

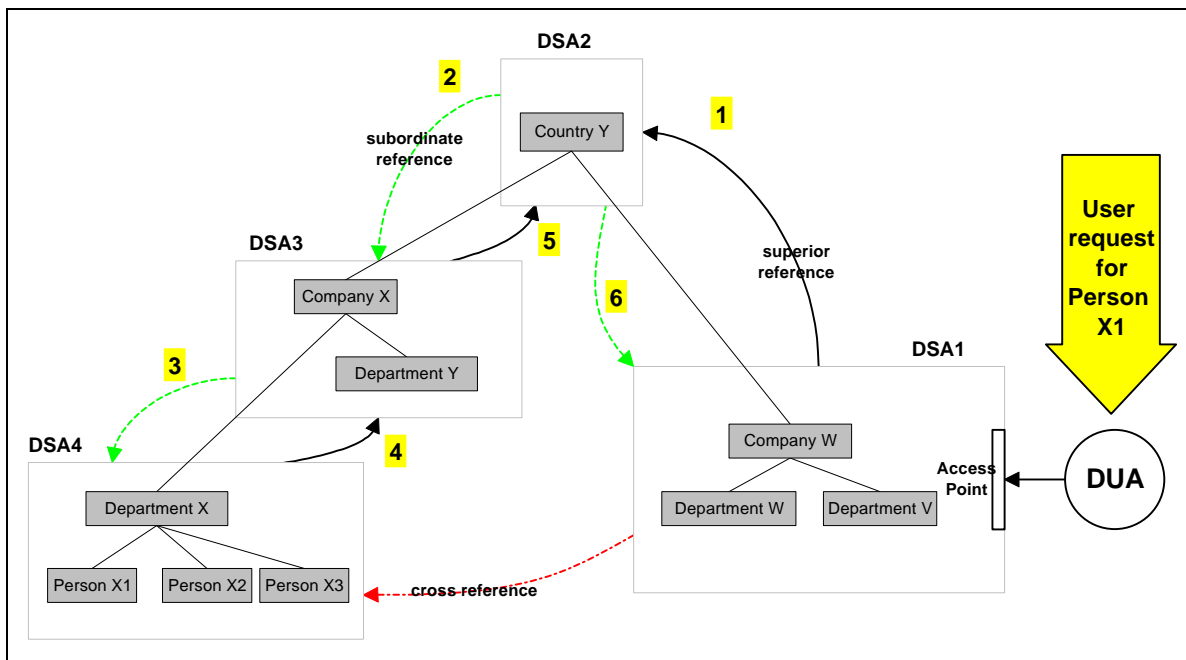


Figure C-14: Distributed name resolution

Since each DSA knows both the access point of a superior DSA and the access points of all immediate subordinates, all requests can be solved that way, either by returning the requested information or by returning an error when the entry is not part of the DIT.

C.4.4 Replication of Directory Information

Another important aspect of distribution is replication of information. The concept of replication is to store several copies of the same information redundantly at different locations. The information system ensures consistency of the different copies in specified time intervals.

The replication mechanism described in recommendation X.525 is referred to as *shadowing*. The shadowing concept defines one of the DSAs that stores a copy of the replicated information as the *master DSA*. All manipulations, i.e. deletions or modifications of the replicated objects are allowed on the master DSA's copy only. The modifications are then sent to the other DSAs, the *shadow DSAs* following a specified schedule. The standard does not define how frequently these updates have to be performed.

To set up a shadowing relation between two DSAs, a *shadowing agreement* must be defined. This agreement fixes the terms of the shadowing process like the scope of shadowed information, the update schedules, and security issues. The two shadowing parties are referred to as *shadow supplier DSA* providing the information and *shadow consumer DSA* holding the shadow copies. One DSA might be shadow supplier for one part of the DIT and shadow consumer for another part. Furthermore, information can be shadowed over several levels, so that a DSA that holds shadow copies might operate as shadow supplier for these copies in a relation with another DSA. This is referred to as *secondary shadowing*, whereas the

replication between the master DSA and a consumer is referred to as *primary shadowing*. The concept of shadowing is illustrated in Figure C-15.

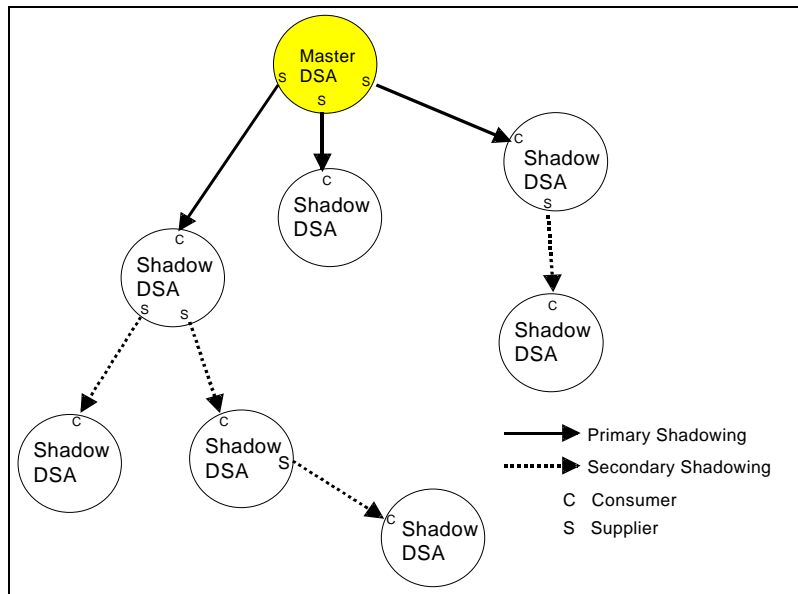


Figure C-15: Shadowing

Shadowing agreements are transported via the directory operational binding management protocol, whereas for updates of shadowed information the directory information shadowing protocol is used. Both protocols are described in sections C.5.3 and C.5.4 respectively.

Further details about the directory replication, such as implementation rules, listings of different attributes that hold shadowing information and detailed contents of a shadowing agreement, can be found in [ISO/IEC 1993d] and [Chadwick 1994] (chapter 6).

C.5 The Directory Protocols

The different components of an X.500 directory, the directory service agents (DSA) and the directory user agents (DUA) communicate through a set of protocols. These protocols provide different levels of services. This section describes the different protocols and provides an overview of their functionality.

The protocols to be discussed are tightly embedded into the OSI layer architecture and operate as application layer protocols that provide the communication medium for the DSAs that are application processes in OSI terms. The X.500 protocols make use of other OSI services like

- ❑ the remote operations service element (ROSE)
- ❑ the reliable transfer service element (RTSE)
- ❑ the association control service element (ACSE)
- ❑ the OSI presentation layer
- ❑ the OSI low layer services

This list gives an impression of the integration of an X.500 implementation in the OSI environment. The services are not described further here, details can be found in [ISO/IEC 1993e] (section 6.7).

Figure C-16 gives an overview of the different directory protocols described in the following.

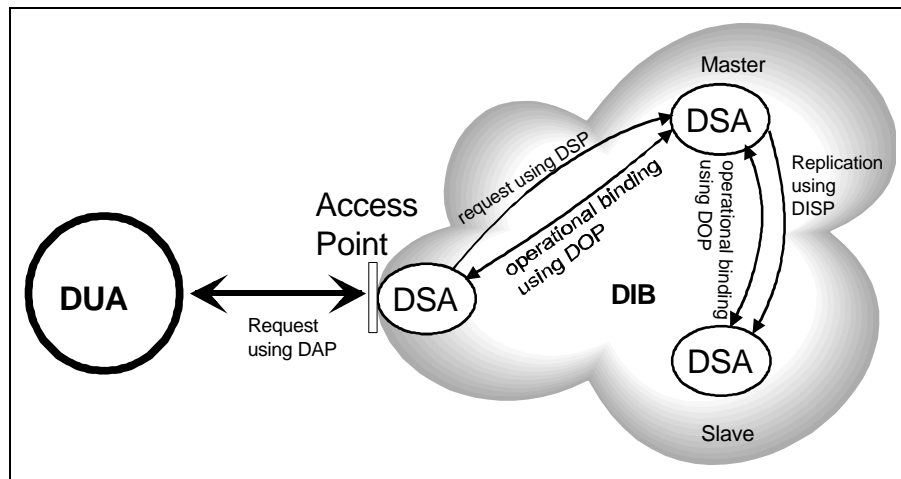


Figure C-16: Overview of the directory protocols

C.5.1 The Directory Access Protocol - DAP

The communication between a DUA and a DSA is specified in the *directory access protocol* (DAP). The DAP combines the *directory abstract service* (see section C.6) with other OSI protocols that manage the communication between the different OSI layers.

It consists of the operations that are used to access or modify directory information. Since these operations are crucial for the directory service, they are defined in a separate recommendation (X.511). This chapter discusses them separately in section C.6.

C.5.2 The Directory System Protocol - DSP

The *directory system protocol* (DSP) is used to pass DAP requests from one DSA to another. Therefore, the DSP has to be very similar to the DAP. Basically, all DAP operations are also part of the DSP, extended with additional parameters required for chained operations. These parameters include information about the current state of the request as well as about security issues. The result and the errors returned in the DSP are very much identical with those of the DAP.

C.5.3 The Directory Operational Binding Protocol - DOP

The setup and the maintenance of the inter-operation between two DSAs are defined in the *directory operational binding protocol* (DOP). An operational binding defines the terms of a cooperation between two DSAs. Cooperation here means not the connection of the DSAs for a short period of time to exchange information, but the general agreement to share information or jointly serve user requests.

The operations defined in the DOP are described in Table C-2.

Operation	Description
dSAOperationalBindingManagementBind	is used to start a connection between two DSAs
dSAOperationalBindingManagementUnBind	is used to terminate a connection between two DSAs
establishOperationalBinding	transfers the parameters of the operational binding between the two involved DSAs and initiates the relationship
modifyOperationalBinding	is used to update the binding parameters
terminateOperationalBinding	is used to terminate the binding relationship

Table C-2: Operations of the DOP

The standard distinguishes between two types of operational bindings, the *hierarchical operational bindings* and the *shadow operational binding*.

A hierarchical operational binding (HOB) is established between two DSAs that are located one below the other in the DIT. They exchange knowledge information about the part of the DIT they master, and optionally the superior DSA may transfer information about access control, schema management, and collaborative attributes if both DSAs are member of the same organization.

A shadow operational binding (SOB) is established between two DSAs that have a shadowing agreement with each other (see section C.4.4). The SOB specifies the terms of this agreement.

C.5.4 The Directory Information Shadowing Protocol - DISP

Two DSAs that have established a shadow agreement exchange the replicated information via the *directory information shadowing protocol* (DISP). The DISP manages the initiation and verification of the update cycle as well as the actual transfer of shadow information from the supplier to the consumer. The operations of the DISP are described in Table C-3.

Operation	Description
dSAShadowBind	is used to establish an update procedure
dSAShadowUnbind	is used to terminate an update procedure
requestShadowUpdate	is used by the consumer to request updated information
coordinateShadowUpdate	is used by the supplier to initiate an update
updateShadow	is used by the supplier to transfer updated information to the consumer

Table C-3: Operations of the DISP

The shadowing procedure can be initiated by either the supplier or the consumer DSA by using the *dSAShadowBind* operation. If the consumer started the operation, it uses the *requestShadowUpdate* operation to tell the supplier to start an update procedure. When the supplier itself wants to update the information it uses the *coordinateShadowUpdate* operation to inform the consumer about the start of an update process. Both operations carry parameters

that inform the other party about the update strategy and that ensure proper data flow. There are three types of update strategies:

- ❑ *Incremental update* sends only the data that has changed since the last update
- ❑ *Total update* sends the whole shadow information
- ❑ *NoChanges* tells the consumer that there is no new information available

Additionally, the two DSAs exchange a time stamp for each update procedure to unambiguously identify the sent packages. Both the `coordinateShadowUpdate` operation and the `requestShadowUpdate` operation include the time stamp of the package that was transferred during the last update procedure and so it is ensured that no packages gets lost and no packages are sent twice.

C.6 The Directory Services

The standard recommendation X.511 deals with the services that the directory as a whole provides to its users. The services are the access language to the directory. All possible operations like retrieving and modifying information are defined in here.

The *directory services* can be split up into read operations, search operations and modify operations. Additionally, the connection from a DUA to a DSA has to be established through a `bind` operation and terminated by an `unbind` operation. This section describes these services, yet leaving out the detailed syntax of the service commands.

C.6.1 Bind Operations

When two directory agents, either two DSAs or one DUA and one DSA, want to communicate with each other, the originator of this communication has to perform a *bind* operation. Within this operation, the requester transmits the information necessary for authentication (credentials) including the authorization level he wants to use. Section C.8 has details about the authentication process. Additionally, a list of possible version numbers of the directory services (currently always version 1) is transmitted.

When the request is successful, the receiver returns his credentials and the version number. If the request is denied, the receiver returns either a security error or a service error. A security error occurs when either the authentication level indicated by the requester is insufficient or the credentials are invalid, e.g. the password is wrong. A service error indicates that the receiver is not able to provide any of the service versions offered by the requester. In this case a list of versions supported by the receiver is returned.

After finishing a working session with the directory, the requester has to perform an `unbind` operation that closes the connection between the two parties. The *unbind* operation has no

parameters and cannot fail. After the unbind operation, there is no communication possible in either direction, i.e. there is no feedback from the receiver.

C.6.2 Read Operations

In this category the *read* operation itself and the *compare* operation are aggregated. By using the read operation users can query the attributes of an entry whose *purported name* is known to them. A purported name has the same syntax as a distinguished name but has not proven yet to be the distinguished name of an object. Alternatively, the purported name may identify an alias entry of the requested object. Furthermore, users specify the information they are looking for as an *entry information selection* and may ask for their modify rights concerning the returned entries. The common argument parameter set is also attached to the request.

The result of a read operation comprises the requested information as *entry information* and additionally an indication about whether the requester may modify the entry or not. The common result parameters are transmitted along with the system's answer. The parameters, results and errors of the read operation are listed in Table C-4.

Read		
Parameters	object selection modifyRightsRequest commonArguments	the DN of the entry the requested information (EntryInformationSelection) requests information about the user's access level
Results	entry modifyRights commonResults	the DN of the entry (if alias was entered) the user's access rights (add, remove, rename, move)
Errors (only one)	attributeError nameError serviceError referral abandoned securityError	

Table C-4: Read operation

The second operation of the read family is the *compare* operation, that compares a given entry to an attribute value of a specified entry. The result is *1* if both values are equal and *0* if not. This operation is used primarily to check passwords because in this case it is important not to return the stored value but only the result of the comparison. The result also includes the name of the object. The parameters, results and errors of the compare operation are listed in Table C-5.

Compare		
Parameters	name purported common arguments	the DN of the entry the assumed value
Results	name matched fromEntry matchedSubtype commonResults	the DN of the entry (if alias was entered) result, 0 for false, 1 for true indicated if result comes from entry or copy if the comparison succeeded and the result was found in a subtype, the name of the subtype is indicated here
Errors (only one)	attributeError nameError serviceError referral abandoned securityError	

Table C-5: Compare operation

Abandon

Any operation can be canceled by using the abandon operation. The only parameter indicates the operation to be abandoned and the only error reports that the operation failed. There is no result if the operation was successful. The parameters, results and errors of the abandon operation are listed in Table C-6.

Abandon		
Parameters	invokeID	ID of operation to be abandoned
Results	no result	
Errors (only one)	abandonFailed	

Table C-6: Abandon operation

C.6.3 Search Operations

The search operation itself and the list operation can be found in this group. The list operation provides the user with the immediate subordinates of a given entry. Its arguments are the object whose subordinates are requested and optionally a paged result request that indicates that the result shall be divided into several pages.

The result contains the distinguished name of the object in case the user entered an alias name of it and a collection of information about the subordinates, among them the distinguished name and flags indicating whether they are alias entries or not and whether the information comes from a copy or from the original entry. When the home DSA had to chain the request to

another DSA that attached a digital signature to its answer, this information must be provided separately to the user so as not to destroy the signature. It is contained in the *uncorrelatedListInfo* argument of the result. Should the DAS fail to complete an operation, the reason is provided in the *partialOutcomeQualifier* argument. Examples are that a size limit was exceeded, chaining would be required but could not be performed or further results will be provided on following pages. The parameters, results and errors of the list operation are listed in Table C-7.

List		
Parameters	object pagedResults commonArguments	starting point specifies output format
Results	listInfo (list of) name subordinates (list of) rdn aliasEntry fromEntry partialOutcomeQualifier commonResults uncorrelatedListInfo	of entry whose subentries are requested of subordinate flag that indicates alias or object entry or from copy information about incomplete answer if necessary signed answers from remote DSAs
Errors (only one)	nameError serviceError referral abandoned securityError	

Table C-7: List operation

The purpose of the search operation itself is to provide the user with a powerful means to find entries that match certain criteria. Therefore the user defines a base object from where the search is to begin and the scope of the search. The scope may be

- ❑ Only the base object itself (*baseObject*)
- ❑ All immediate subordinates of the base object (*oneLevel*)
- ❑ The whole subtree starting at the base object (*wholeSubtree*)

The user can further refine the search by specifying a filter that defines details about the requested objects, and a selection that defines which information from the entries the user is interested in. The result can even be made clearer by using the parameter *matchedValuesOnly* that tells the directory service to return only the values of an attribute that match the filter. Otherwise, if one value matches the filter all values are returned. Furthermore, the user may

specify some parameters concerning alias handling. The parameters, results and errors of the search operation are listed in Table C-8.

Search		
Parameters	baseObject subset (one of) baseObject oneLevel wholeSubtree filter searchAliases selection pagedResults matchedValuesOnly extendedFilter commonArguments	starting point for search scope single or combined filter item if set to true aliases are de-referenced specifies which information is searched (<i>EntryInformationSelection</i>) specifies output format indicates that only those values shall be returned that match search used for compatibility issues to X.500 (88)
Results	searchInfo (list of) name entries partialOutcomeFilter commonResults	distinguished name of the result entry requested information (<i>EntryInformation</i>) information about incomplete answer if necessary
Errors (only one)	attributeError nameError serviceError referral abandoned securityError	

Table C-8: Search operation

C.6.4 Modify Operations

The first of the modify operations is the add entry operation that allows users to add a leaf entry to the DIT. The name and position in the DIT is defined by the distinguished name of the new entry that contains the DN of its immediate superior. Furthermore, the user defines a set of user attribute types and their values which will be contained in the new entry and which have to conform to the directory subschema. The structural object class the entry should belong to is also defined. The DSA automatically adds further operational attributes to the entry. Finally users determine which DSA is to hold the new entry. There is no feedback from

the directory when the operation is successful. The parameters, results and errors of the addEntry operation are listed in Table C-9.

AddEntry		
Parameters	object entry targetSystem commonArguments	distinguished name of new entry attribute set DSA that holds the entry
Results	NULL	a result is returned, but does not contain any information
Errors (only one)	attributeError nameError serviceError referral securityError updateError	

Table C-9: addEntry Operation

The removeEntry operation allows users to delete a leaf entry from the DIT. They just provide the DN of the entry that shall be deleted. Again, no feedback is given when the operation succeeded. The parameters, results and errors of the removeEntry operation are listed in Table C-10.

RemoveEntry		
Parameters	object commonArguments	distinguished name of entry
Results	NULL	a result is returned, but does not contain any information
Errors (only one)	attributeError nameError serviceError referral securityError updateError	

Table C-10: removeEntry Operation

The modifyEntry operation may be used to add or remove whole attributes and to add or remove attribute values of existing attribute types. In order to achieve this, the user specifies

the entry he wants to modify and a set of the changes listed above. To replace an attribute value, a removeValue and an addValue may be performed within one operation. It is not possible to modify the distinguished name of an entry by using the modifyEntry operation. The parameters, results and errors of the modifyEntry operation are listed in Table C-11.

ModifyEntry		
Parameters	object changes (one of) addAttribute removeAttribute addValues removeValues commonArguments	distinguished name of entry list of changes to be performed attribute type and value to be added attribute type to be removed attribute type and value to be added attribute type and value to be removed
Results	NULL	a result is returned, but does not contain any information
Errors (only one)	attributeError nameError serviceError referral securityError updateError	

Table C-11: modifyEntry Operation

The modification of the distinguished name of an entry can have such an enormous effect on the whole DIT structure that the standard defines an own operation for this. The modifyDN operation is able to modify the DN of a leaf entry and of a non leaf entry which implies modifying the DN of all of its subordinate entries. It is furthermore able to move a leaf entry or a whole subtree to another location within the DIT. The parameters, results and errors of the modifyDN operation are listed in Table C-12.

ModifyDN		
Parameters	object newRDN deleteOldRDN newSuperior commonArguments	distinguished name of entry the new relative distinguished name of the entry true, if attributes values that built former RDN are to be deleted specifies the new superior entry of the entry or subtree
Results	NULL	a result is returned, but contains no inform.
Errors (only one)	nameError serviceError referral securityError updateError	

Table C-12: modifyDN Operation

C.7 The Directory Schema

Another very important part of the standard recommendations is the specification of the directory schema. The directory schema comprises a set of rules that ensure the proper function of the directory services. They encompass the definition of new object classes and attribute type as well as naming entries and positioning them in the DIT.

Since the control over the directory information is distributed over several independent authorities the administration of the schema is distributed as well and so called subschemas control the autonomous administrative areas. The directory schema governs entries as well as user and collective attributes, whereas subentries and operational attributes are governed by the directory system schema (discussed in section C.7.6).

Figure C-17 (adapted from [ISO/IEC 1993a], p. 32) gives an overview of the directory schema, showing the schema elements on the left hand side and the directory components which they apply on the right hand side.

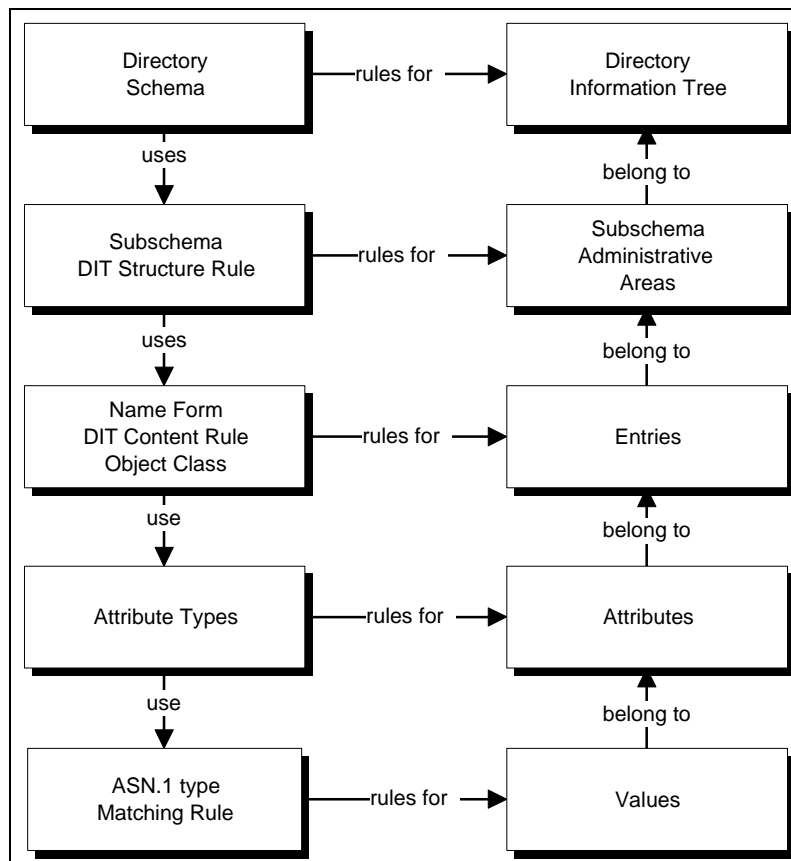


Figure C-17: Overview of the directory schema

ASN.1

The schema definition and all other components of X.500 such as object classes and attribute types are defined using ASN.1. ASN.1 means *abstract syntax notation one* and is a language created especially to define data types in a standardized way. The following sections describe and explain the directory schema more compactly and omit the ASN.1 syntax. Chapter 12 of [ISO/IEC 1993a] explains ASN.1 in greater detail.

C.7.1 Object Class Definitions

The standard allows and supports the definition of new object classes, completing those defined in [ISO/IEC 1993b]. To ensure compatibility between all classes and to enforce a uniform structure of all objects belonging to the same class, the standard defines rules for the creation of new classes. These rules include the position in the class hierarchy, the presence of an identifier, the list of attribute types that must be or can be assigned to objects and other administrative issues.

According to X.501, the definition of an object class must contain the following information:

- The name of the class from which the new class is derived, i.e. its superior class
- A unique identifier of the object class that will be stored in the *objectClass* attribute of all entries belonging to that class

- ❑ The object class type specified as either "abstract", "structural" or "auxiliary"
- ❑ A list of attributes that are mandatory for an entry belonging to the class
- ❑ A list of the optional attributes

C.7.2 Attribute Type Definitions

Analogously to object classes, it is also supported to define additional attribute types to those defined in [ISO/IEC 1993f]. Besides the specification of an identifier and the position in the attribute type hierarchy, the used syntax must be indicated. A syntax is defined via an ASN.1 data type, like *Integer* or *DirectoryString*. The syntax also contains size limits for the attribute values. Most of the remaining parts of the definition have administrative purposes.

An attribute type definition contains the following information according to [ISO/IEC 1993a] (chapter 12.4):

- ❑ A unique identifier
- ❑ The position in the attribute hierarchy by naming its superior attribute type
- ❑ The syntax of the attribute type (e.g. as an ASN.1 data type)
- ❑ The matching rules that will be applied by search queries (optional)
- ❑ Single- or multi-valued attribute type
- ❑ *Operational* or *user* attribute type
- ❑ *Collective* attribute type marker (optional)
- ❑ Prohibition of user modification (optional)
- ❑ The application of the attribute type (only for operational attributes), that is either directory operational attribute, DSA-shared operational attribute, or DSA-specific operational attribute

C.7.3 Matching Rule Definitions

Apart from a selection of attribute types, the standard defines a collection of matching rules in [ISO/IEC 1993f]. Examples are *CaseIgnoreMatch* that compares two strings and returns true if they are equal disregarding the case, or *IntegerOrderingMatch* that compares the presented value with an attribute value and returns true if the attribute value is less than the presented value. Matching rules similar to those can be defined by users of the directory for specific purposes and they may often only be applicable for particular custom attribute types.

The X.501 recommendation describes the definition of matching rules. They must include:

- ❑ A unique identifier for the matching rule
- ❑ The syntax of an assertion of the matching rule, i.e. of the value the user entered (ASN.1 data type)
- ❑ The types of matches supported by the rule, e.g. "true", "false" and "undefined"
- ❑ The rules for evaluating assertions against the values stored in the DIB

C.7.4 DIT Structure Definitions

This part of the directory schema deals with the position of an entry in the DIT (DIT structure rule) and defines rules for building an entry's relative distinguished name (name form).

Name Form

A name form specifies the list of attributes that are used to build the RDN of an entry that belongs to the object class the uses this name form. The definition of a name form requires:

- ❑ A unique identifier
- ❑ The object class it names
- ❑ A list of the mandatory attributes that are used to build the RDN for the respective object class
- ❑ A list of the optional attributes that can be used to build the RDN

DIT structure rule

Each name form and thus each entry of the DIT is governed by a particular DIT structure rule (specified in the entry's *governingStructureRule* attribute), that specifies the name form used by the entry. Furthermore the entry's structure rule defines the available superior structure rules and thus the available object classes for superior entries. A DIT structure rule definition includes:

- ❑ A unique integer identifier
- ❑ The name form governed by the rule
- ❑ A set of allowed superior structure rules (optional)

C.7.5 DIT Content Rule Definitions

In order to control the contents of an entry in addition to those defined within the entry's object class, DIT content rules are built. At the most, one DIT content rule may be assigned to an entry via its object class and determines auxiliary object classes, mandatory and optional attributes that are additionally assigned to the entry as well as a set of optional attributes that are precluded from the entry. A DIT content rule definition includes:

- ❑ The structural object class it governs
- ❑ The auxiliary object classes
- ❑ Mandatory attributes
- ❑ Optional attributes that are additionally allowed
- ❑ Optional attributes that are permitted

C.7.6 The Directory System Schema

In contrast to the directory schema that controls directory user information, the directory system schema deals with directory operational information represented by operational attributes and subentries. Thus the directory system schema contains definitions of attributes assigned to subentries and operational attribute definitions. Analogous to the directory schema, the directory system schema is also distributed, which means that each administrative authority is responsible for defining the rules applied within its part of the DIT.

Subentries

Subentries, as discussed in section C.4.1, are an administrative means to define characteristics of an autonomous part of the DIT, like access control or collective attributes. The *subentry* object class is defined as a subclass of *top* and is named by its *commonName* attribute. The structure of the subtree administered by the subentry is defined by the *subtreeSpecification* operational attribute using the syntax described in section C.3.2 (using "base", "chop" and "specification filter"). Additionally the standard defines two auxiliary object classes that occur as additional values in the subentry's *objectClass* attribute and specify what kind of specific administrative area the subentry controls. These object classes are:

- ❑ *accessControlSubentry* containing an *prescriptiveACI* attribute
- ❑ *collectiveAttributeSubentry* that contains all the collective attributes in the members of the subtree

Each entry of a subtree may contain the *collectiveExclusions* operational attribute whose values specify collective attributes to be excluded from the respective entry.

Operational attributes

The directory system schema furthermore defines the following operational attributes that are used to administer directory entries and keep track of their modifications.

- ❑ *administrativeRole* is assigned to an administrative entry and indicates what types of subentries may be subordinated to that administrative entry
- ❑ *createTimestamp* holds the creation time of an entry
- ❑ *modifyTimestamp* holds the time an entry was last modified

- *creatorsName* holds the distinguished name of the person who created an entry
- *modifiersName* holds the distinguished name of the person who last modified an entry

C.7.7 The Directory Schema Administration

The administration of the different subschemas used by administrative authorities is supported by the standard. A special subentry exists for each subschema specific administrative area that is member of the auxiliary object class "subschema" and stores the complete subschema applied at this administrative area. That helps the DSA to allow only updates that are consistent with the schema and it furthermore provides an easy means to make the area's subschema available to the directory users. The latter is of particular importance in the context of access of users of foreign domains. As they might not be aware of elements of the local subschema, e.g. customized attribute types and object classes, there might be difficulties in properly accessing and reading directory entries. Storing the subschema in the subschema subentry (and thus implicitly in each entry), enables them to extend their system in order to recondition the consistency between the two domains.

The operational attributes assigned to a subschema subentry (i.e. a subentry that is a member of the auxiliary object class "subschema") are called subschema policy attributes. They are, as mentioned above, available in each entry belonging to the subschema specific administrative area for reading, but may only be modified via the subschema subentry. Each attribute type is multi-valued with each value representing a schema element (e.g. a DIT structure rule or an attribute type definition) which basically consists of:

- The ASN.1 definition
- A natural language name and description
- A Boolean "obsolete" marker that indicates that this element is out of use—These markers help authorities to administer the constant change of the schema. When rules are modified, the old versions can be marked as obsolete instead of deleted immediately in order to have a reference for the entries still depending on that rule until they are changed.
- An information part that specifies the respective schema element in more detail (optional)

The subschema policy attributes are:

- *dITStructureRules* contains the DIT structure rules applied in the subtree
- *dITContentRules* indicates the DIT content rules applied in the subtree
- *matchingRules* lists the matching rules including an information part that indicates the attribute syntax governed by the respective rule

- ❑ *attributeTypes* stores all attribute types. Its information part includes all information required to specify the attribute type
- ❑ *objectClasses* holds all used object classes. Its information part includes all information required to specify the object class
- ❑ *nameForms* contains the name forms including their definitions

Additionally the directory system schema defines two more attribute types that are assigned to each entry of the DIB:

- ❑ *governingStructure Rule* lists the DIT structure rule that governs the entry
- ❑ *structuralObjectClass* that indicates the structural object class on the entry

Finally the *matching RuleUse* attribute type is assigned to each matching rule definition and indicates the attribute types governed by the matching rule.

C.8 Authentication

It is absolutely crucial for a directory, as for other information systems, to ensure that users logging in are correctly authenticated, i.e. they have to prove that they really are the person they claim to be. Network operating systems for example typically prompt users for a password to ensure their identity. The X.509 standard recommendation defines the authentication framework applied at X.500 directory services. It defines two levels of authentication called simple and strong authorization respectively.

C.8.1 Simple Authentication

The simple authorization basically uses the distinguished user name together with an optional password to identify the user. The DSA compares the provided password with a local copy stored in the *userPassword* attribute of the user's entry. In the least secure scenario, the name and password data is sent unprotected. This makes it quite easy for third parties to intercept the information during the transmission and then to access the directory unauthorized on the user's behalf.

A first step towards secure authorization is protection of personal information. To protect passwords, one-way functions are used. A one-way function is a mathematical algorithm that transforms a clear password into a protected one without being able to reverse the operation. Both parties involved in the authentication process need to hold the same one-way function in order to cooperate properly. The requesting unit transmits the protected password together with the clear user name to the receiver, who then applies its one-way function to the stored copy of the password and compares the result with the received information. If they are identical, the authentication was successful.

A more advanced version of this method is to use additional parameters, like a time stamp and/or a random number for the one-way function. The parameters then have to be transmitted in the clear together with the protected password. The time stamp prevents third parties from intercepting the sent information and using it at a later stage, because the time stamp will then have expired. A random number improves the method even more because a second login trial, even when the time stamp is still valid, is denied because the random number is the same and this is interpreted as an illegal login by the DSA.

The standard defines a last improvement of the protected password method by using a second one-way-function along with a second set of input parameters like time stamp and random numbers. The extra encoding of the information is in order to make the inversion of the one-way function even more difficult.

Yet, there is still insecurity remaining with a scenario where personal information is transmitted to a wrong recipient by mistake. This person then can use this authentication information as long as the time stamp is valid to access the directory on the first user's behalf.

C.8.2 Strong Authentication

Due to the shortcomings of the simple authentication in terms of security, the standard defines a second model, the strong authentication. It uses certification as well as encryption and digital signatures based on the public key cryptographic system.

The public key method is a permutable, asymmetric encryption algorithm. This means that there is a pair of two different keys used from the sender and the recipient (asymmetric) and both keys can either be used to encipher a message or to decipher the message (permutable). One part of the pair is published in the directory and stored within the user's entry (public key) and the other part is kept by the user (private key).

Digital signatures

Digital signatures are used to make it impossible for users to send messages or requests to the directory on other user's behalf. To achieve this, the user encrypts a message with a private key (it is signed) before submitting it to the directory. The DSA attempts to decrypt the message with the public key of the user and if successful, can be sure that the message is authentic. The method is actually slightly more complex and powerful. At first the message is reduced by using a hash algorithm and then encrypted with the private key. This result is sent along with the clear version of the message. The receiver uses the same hash algorithm to reduce the clear message and decrypts the signed part. If both results are identical, the systems knows that it's authentic, but furthermore it can be sure that the clear version of the message was not modified during the transmission. This method identifies a user by his private key and thus can only work reliably if private keys are kept secretly.

Certificates

All concepts described above assume that the users trust the directory, e.g. in trusting the authentication of the public key stored in the directory. To improve the confidence, so-called certification authorities (CAs) are established that are trusted by users to create certificates on their behalf. Certificates are certified public keys that are produced by signing personal information of the users. This information includes

- ❑ the distinguished name of the user
- ❑ the public key of the user
- ❑ the distinguished name of the certification authority
- ❑ a unique identifier (optional)
- ❑ a time stamp that indicates the expiration time of the certificate
- ❑ the identifier of the encryption algorithm used for signing data

The receiver of a signed or encrypted message can now check the certified public key (certificate) of the sender by validating the digital signature of the CA attached to the key. If the receiving party is able to decrypt the certificate, they can be sure that the public key, which is either received along with the message or which is stored in the directory, is authentic. A prerequisite for this concept to work is that both parties share a common certification authority and that the CAs publish their public keys.

The second prerequisite is easy to meet, because CAs, that are often organizations that provide security services or IT departments within large companies, make their public keys available for their users. However the first prerequisite is harder, two parties (usually directory service agents or directory user agents) that are located in different companies or even countries often may trust different certification authorities. To solve that problem, the standard defines so-called certification paths. Certification paths are built by CAs exchanging their own certificates. This process, called cross certification, enables parties to safely authenticate each other when there is a closed path of CAs between their own CAs.

Authentication Procedures

Additional to the general principle of authentication, the standard specifies three different scenarios for two parties authenticating each other. The scenarios differ in terms of their complexity and reliability.

One-way Authentication

The most simple scenario requires only that a message by the sender is encrypted in the way described above.

Table C-13 lists the different message parts.

Message part	Benefit
name of recipient	the message is intended for the recipient
name of sender	the sender is really who she claims to be
time stamp, random number	the message was not sent before
digital signature of sender	it is in the original state
identifier and parameters of algorithm, certification path	

Table C-13: Message

Two-way (mutual) Authentication

In this scenario the recipient of the initial message sends an authentication back to the sender to ensure that the message actually arrived at the intended recipient.

Three-way Authentication

The three-way authentication is a variant of the mutual authorization. The difference is that the sender does not transmit a time stamp but only a random number. The recipient includes this number to his signed reply and so the initiator can be sure that this is indeed the answer to his request. In a third step, the sender signs the recipient's random number and transmits it back and so ensures that the message isn't replayed.

C.9 Authorization - Access Control

The standard also takes care of controlling the access to directory information, i.e. both user and operational information. This is done by defining a framework called access control scheme that includes:

- ❑ The specification of *access control information* (ACI), i.e. defining which user may access which information
- ❑ The enforcement of the access rights defined by the ACI
- ❑ The maintenance of the ACI

The definition and implementation of a specific access control scheme is left to the administrative authorities that control the access control specific areas (discussed in section C.4.1). An access control specific area is characterized by using a single access control scheme that is defined in the *accessControlScheme* operational attribute of the area's administrative entry.

The standard defines two possible, specific access control schemes, the *basic access control scheme* and the *simplified access control scheme* which is a subset of the first.

C.9.1 The Basic Access Control Model

The basic access control model describes the scope of the model itself, the objects involved in an access control decision and the different permission categories.

It deals with controlling information contained in the DIB, i.e. user as well as operational information. However, the basic access control model does not manage an DUA's access to a DSA application.

The objects described in the model are:

- ❑ *protected item*, i.e. the directory object to be accessed—Protected items may be aggregated to collections in order to make the administration easier.
- ❑ Users accessing the directory, called *requesters*, and the user classes they may belong to
- ❑ Different *permission categories* required for the directory operations
- ❑ The decision algorithm the system uses to evaluate a user access, called *access control decision function (ACDF)*

Thus it is possible to assign different permission categories for different users or user groups to each protected item. This allows administrators to define a very detailed access control system for the area for which they are responsible.

Protected Item

The directory is split into several layers and to access an inner layer the user has first to access the outer layers. These layers, called protected items are:

- ❑ *attribute value*, e.g. a single value of a multi-valued attribute
- ❑ *self value*, the distinguished name of the current user (e.g. for allowing users to add themselves to mailing lists)
- ❑ *all attribute values*, all values of an attribute or an attribute collection
- ❑ *attribute type*, excluding the attribute values
- ❑ *all user attribute types*, all user attributes belonging to an entry (there is no collection for operational attributes)
- ❑ *all user attribute types and values*, includes attribute values
- ❑ *entry*, access to the entry must be granted explicitly to access attribute types and values

Figure C-18 illustrates the different levels of protected items. To access attribute value "Value 2", users explicitly need to have access rights to "Entry 2", "Attribute Y" and "Value Y".

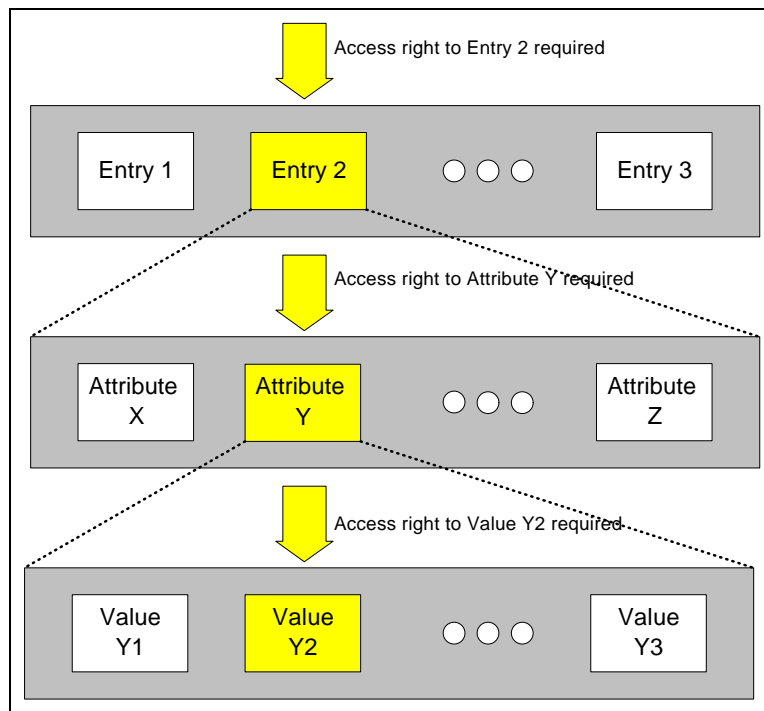


Figure C-18: Protected items

User classes

User classes are the entities that can be granted or denied access to the protected items. Apart from defining users by name in the ACI, it is a strong requirement to an access control model to be able to aggregate people in some way or to know concepts that abstract from the usage of real names in the ACI. In this way, the access control information is made independent from changes in the organization structure e.g. from users changing departments or leaving the company. The standard therefore defines the following user classes:

- ❑ *name* is the distinguished name of a person or application
- ❑ *this entry* specifies the distinguished user name which is identical to the distinguished name of the entry he wants to access
- ❑ *user group* as the distinguished name of an entry representing a group of users. Access is granted to all distinguished names that are included in the *member* attribute
- ❑ *subtree* defined by the distinguished name of the subtree's base, a chop definition is optional. This is especially useful for a definition of all members of a department or a whole organization
- ❑ *all users* indicates public access to an object

Permission Categories

The standard gives access to a protected item in different permission categories. They are listed in Table C-14 along with the protected items they are valid for.

Permission Category	Description	Protected Item
read	allows for the reading of an item after its name was provided	all
browse	allows for the reading of an item without providing its name	entry
compare	allows for the usage of the attributes and values in the compare operation	attributes and values
filterMatch	allows for the usage of a filter by the search operation	attributes and values
add	allows for the creation of a new item	all
remove	allows for the deletion of an item	all
modify	allows for a modification of an item	entry
rename	allows for a change in the RDN of an entry	entry
discloseOnError	allows users to return the name of an item in case of an error	all
export	allows users to remove a subtree in order to move it to another location	entry
import	allows for the pasting of a subtree that was removed from another location	entry
returnDN	allows a user to return the DN of an item during an operation	entry

Table C-14: Permission categories

The access control definition function (ACDF)

Access control information (ACI), i.e. the specification which user classes shall be granted which permission category, can be stored in different locations, either distributed in the *EntryACI* attribute of each entry or rather central in the *prescriptiveACI* attribute of the subentry mastering the access control inner area. Where the ACI is stored will depend on its scope, i.e. whether it is valid for the whole subtree or just for one entry.

The decision whether requested access is finally granted to a protected item or not is taken by the *access control definition function* (ACDF). It considers all ACI items valid for this protected item and finally grants or denies access. The ACDF furthermore uses the requester's DN and her authentication level as input. This is illustrated in Figure C-19.

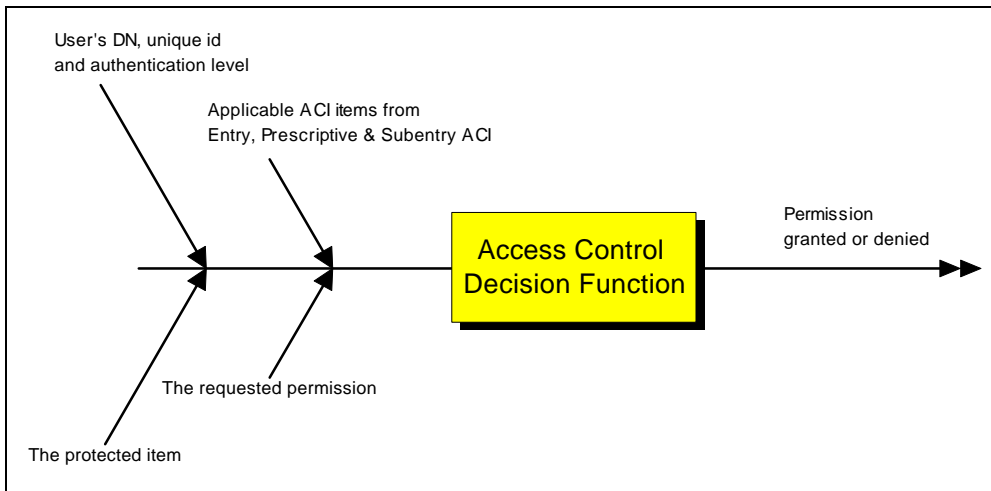


Figure C-19: The access control decision function

C.9.2 The Simplified Access Control Scheme

The simplified access control scheme facilitates the decision process about granting access to a user or not. By not concerning ACI that is stored within single entries (in their EntryACI attribute) and not concerning access control inner areas, it reduces the number of evaluations of the ACDF. The only ACI relevant for the simplified access control scheme is the one stored in the prescriptiveACI attribute of the subentry of the access control specific area the requested item belongs to.

C.10 The Object Classes and Attribute Types Specified in the Recommendations

Standard recommendation X.520 lists some basic attribute types and matching rules that may be used to characterize the objects stored in the directory. Recommendation X.521 specifies selected object classes. Table C-15 summarizes the defined object classes as well as the attributes assigned to them.

Object Class	Attributes	Remark
country	countryName* description searchGuide	
locality	description searchGuide LocaleAttributeSet seeAlso	one of localityName, state or provinceName must be present
organization	organizationName* OrganizationalAttributeSet	
organizationalUnit	organizationalUnitName OrganizationAttributeSet	represents subdivisions of organizations

Object Class	Attributes	Remark
person	commonName* surname* description telephoneNumber userPassword seeAlso	generically defines people entries
organizationalPerson	LocaleAttributeSet PostalAttributeSet TelecommunicationAttributeSet organizationalUnitName title	represents people associated with an organization
organizationalRole	commonName* description LocaleAttributeSet organizationalUnitName PostalAttributeSet preferredDeliveryMethod roleOccupant seeAlso TelecommunicationAttributeSet	represents positions or roles within an organization
groupOfNames	commonName* member* description organizationName organizationalUnitName owner seeAlso businessCategory	represents an unordered set of individual objects or other groups of names
groupOfUniqueNames	commonName* presentationAddress* description localityName organizationName organizationalUnitName owner seeAlso businessCategory	represents an unordered set of individual objects or other groups of names whose integrity can be assured
residentialPerson	LocaleAttributeSet PostalAttributeSet	represents persons in the residential environment

Object Class	Attributes	Remark
	preferredDeliveryMethod TelecommunicationAttributeSet businessCategory	
applicationProcess	commonName* description localityName organizationalUnitName seeAlso	represents elements that perform the information processing for a particular application in open systems (ISO 7498)
applicationEntity	commonName* presentationAddress* description localityName organizationName organizationalUnitName seeAlso supportedApplicationContext	represents the aspect of an applicationProcess that are relevant for OSI
das	knowledgeInformation	
device	commonName* description localityName organizationName organizationalUnitName owner seeAlso serialNumber	
strongAuthenticationUser	userCertificate*	
certificationAuthority	cACertificate* certificateRevocationList* authorityRevocationList* crossCertificationPair	

Table C-15: Selected object classes

A complete list of all attribute types defined in X.520 is shown in Table C-16.

Categorization	Attribute Type
System attribute types	knowledgeInformation
Labeling attribute types	name commonName surname givenName

Categorization	Attribute Type
	initials generationQualifier uniqueIdentifier dnQualifier serialNumber
Geographical attribute types	countryName localityName collectiveLocalityName stateOrProvinceName collectiveStateOrProvinceName streetAddress collectiveStreetAddress houseIdentifier
Organizational attribute types	organizationName collectiveOrganizationName organizationalUnitName collectiveOrganizationalUnitName title
Explanatory attribute types	description searchGuide enhancedSearchGuide businessCategory
Postal addressing attribute types	postalAddress collectivePostalAddress postalCode collectivePostalCode postOfficeBox collectivePostOfficeBox physicalDeliveryOfficeName collectivePhysicalDeliveryOfficeName
Telecommunications addressing attribute types	telephoneNumber collectiveTelephoneNumber telexNumber collectiveTelexNumber teletexTerminalIdentifier collectiveTeletexTerminalIdentifier facsimileTelephoneNumber collectiveFacsimileTelephoneNumber x121Address

Categorization	Attribute Type
	internationalISDNNumber collectiveInternationalISDNNumber registeredAddress destinationIndicator
Preferences attribute types	preferredDeliveryMethod
OSI Application attribute types	presentatonAddress supportedApplicationContext protocollInformation
Relational attribute types	distinguishedName member uniqueMember owner roleOccupant seeAlso

Table C-16: Selected attribute types defined in X.520

These predefined object classes and attribute types may be extended by the users, but custom additions will not necessarily be understood in the distributed environment.

The Standard [ISO/IEC 1993b] additionally suggests a DIT structure by defining a set of structure rules. This structure, shown in Figure C-20 is used as basis for the comparison to the infrastructure model in section 5.3.4.

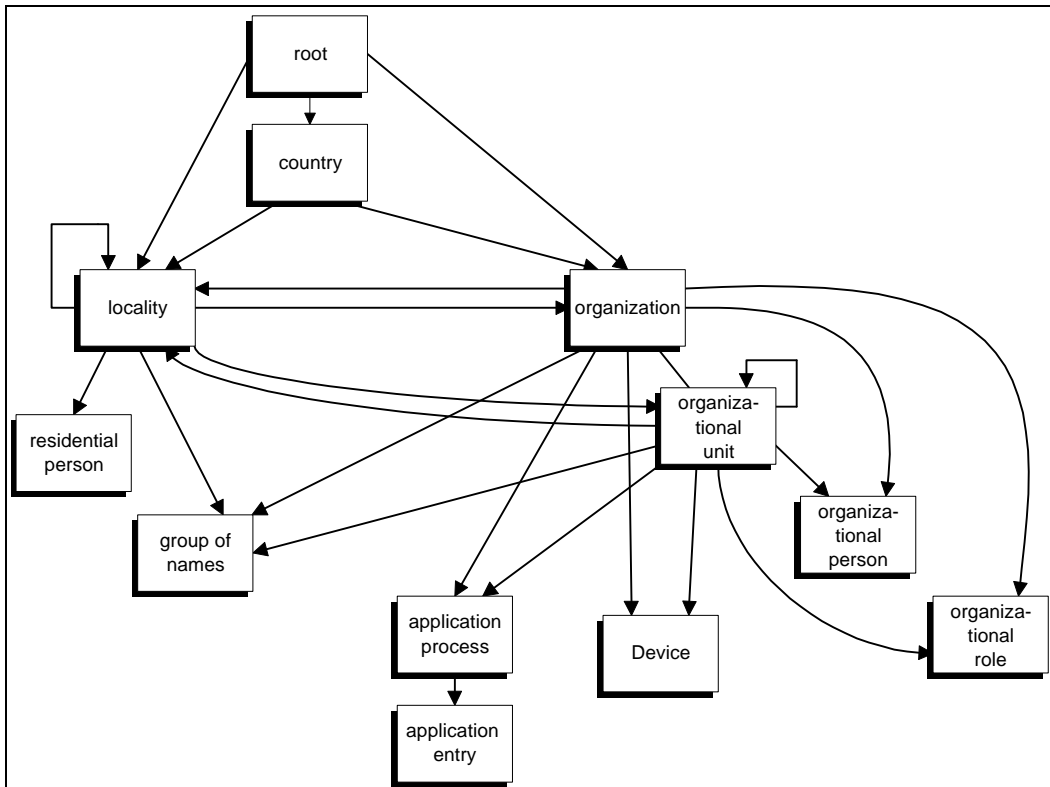


Figure C-20: DIT structure suggested in X.521

C.11 X.500 Summary

This section summarizes the content of the X.500 standard recommendations as to their relevance for the GroupOrga project. The most important principle of X.500 is to give users the impression of a single, homogeneous directory that provides them with information about people, groups and other objects such as hardware or software applications, regardless of their location within the directory. However, from an administrative point of view, directory services have to be easily maintainable, allow for distributed directories and have to be manageable by independent authorities. To fulfill these requirements the X.500 standards offer a set of well developed models covering a wide range of related issues.

The following list summarizes the most important issues covered by the X.500 Standard:

Building blocks of an X.500 directory service

- ❑ On the server side, directory system agents control all directory information.
- ❑ On the client side, directory user agents are the user interface to access any directory information.

Representation of objects and their structuring

- ❑ Real world objects are represented by object entries and characterized by attributes.
- ❑ The overall structure of the directory information is a hierarchical tree (DIT).

Customization

- ❑ Object Classes and Attribute Types can be specified individually.
- ❑ Matching and consistency rules can be modified individually.

Data security

- ❑ An authentication framework ensures the identification of directory users.
- ❑ The authorization concepts also apply for administration.

Directory Distribution

- ❑ The concept of administrative areas allows the distribution of different functions such as access control or schema management.
- ❑ Directory information can be replicated in order to improve performance and reduce costs.

Communication

- ❑ OSI-based protocols are used by the directory components (DUA and DSA) to access the directory, forward user requests or replicate directory information.

- ❑ Users can retrieve directory information by using predefined operations offered by directory user agents.

C.12 The Lightweight Directory Access Protocol (LDAP)

C.12.1 Origination of LDAP

Many companies that were searching for a platform for an enterprise wide directory service and evaluated X.500 in the 1988 version complained about its complexity. Some of the most frequently stated complaints are:

- ❑ *The X.500 demand for a complete OSI protocol stack.* TCP/IP has become the standard communication protocol instead of OSI and although there is a concept to build an X.500 environment on top of TCP/IP this does not meet wide acceptance.
- ❑ *The complexity of implementing both DUAs and DSAs.* Due to the powerful authorization and authentication concepts and the comprehensive operations defined in the directory access protocol (DAP), even the development of an X.500 client requires significant programming efforts
- ❑ *The lack of an API in the standard definitions.* Even though third party products are available they are still to complex (cp. [Kille 1996] on LDAP).
- ❑ *The complex ASN.1 encoding of names and attributes.* This complex encoding mechanism is even applied to simple data elements (cp. [Kille 1996] on LDAP).

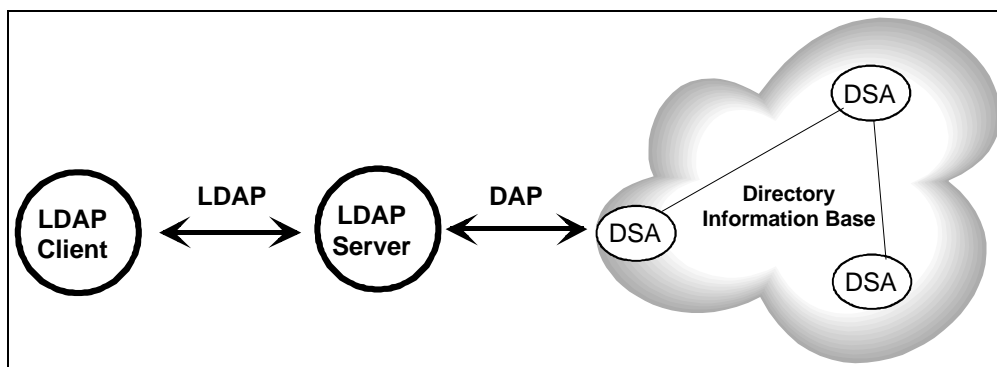


Figure C-21: LDAP Architecture

These problems initiated the development of LDAP (Lightweight Directory Access Protocol). Starting in 1989, the LDAP approach was developed by the Workgroup for Directory Services for Open System (OSI-DS) of the IETF (Internet Engineering Task Force). Originally, it was designed to replace the X.500 DAP in order to make directory clients less complex. An LDAP client communicates to an LDAP server that may use the DAP to access X.500 directory information. This architecture is illustrated in Figure C-21.

C.12.2 Comparison between LDAP and X.500

X.500 in its 1993 version and LDAP have many concepts in common, because both are based on the 1988 version of X.500. The most important common concepts (according to [Kille 1996]) are hierarchical names consisting of typed components (RDNs) to identify entries, object classes and attribute types for structuring information. For accessing and managing the directory contents, both use common operations. These are: read, compare, search, add, delete, modify entry, modify rdn.

The most important simplifications of LDAP in comparison to X.500 and especially the X.500 DAP are that it "does not use the OSI upper layers stack: a client simply makes a TCP connection to an LDAP server" and that "LDAP uses a string representation for all attribute types, values and distinguished names" unlike the ASN.1 encoding X.500 uses [Kille 1996, chapter 2]. These aspects dramatically simplify the development of applications that access directory information and make it interesting for the GroupOrga project.

However, LDAP has some limitations concerning functionality. The most important ones are:

- ❑ LDAP only supports anonymous access or simple authentication with the password carried in clear. It does not support strong authentication.
- ❑ LDAP lacks read and list operations (a workaround is to use the search operation)
- ❑ LDAP v2 servers (sLDAP, standalone LDAP) cannot communicate with each other. Due to this, chaining is not allowed. Furthermore, referrals to the clients are forbidden; thus the scope of a request is reduced to the directory information residing on the home LDAP server.

Despite these limitations, LDAP has become the preferred protocol for directory access (especially via the Internet) of most leading software vendors. IBM, Lotus, Netscape and Microsoft announced LDAP support in their groupware systems. Some vendors even see LDAP as the successor of X.500 as standard for complete directory services. Proprietary extensions do occur and the next version of LDAP could be another step in this direction.

C.12.3 The future of LDAP - Version 3

This section describes the most important enhancements LDAP version 3 will provide. Kille [1996] gives details.

The most important architectural enhancement in LDAP version 3 is that an LDAP environment does not depend on an X.500 directory service anymore. LDAP servers are now capable of providing directory services on their own. However, according to Kille, one of the authors of the LDAP specification, this does not mean that "the light and simple LDAP is now free of the complex and heavy X.500 baggage", because "LDAP relies on X.500 for much of its specifications and for the service definitions" ([Kille 1996], chapter "LDAP evolution").

Furthermore, an LDAP v3 servers can return a referral to another server in order to answer a request for information that it cannot master itself. To manage communication to multiple servers, additional information is available for LDAP clients. For example a server provides a list of the naming contexts it maintains, a list of alternative servers and a list of supported extensions. Additionally, LDAP v3 servers support the extension of the standard attribute types and object classes defined in X520 and X.521.

Other improvements concern security issues. The bind operation for example supports password protection and digital signatures compatible with the X.509 specifications. Moreover, by supporting the SSL (secure socket layer) protocol, LDAP v3 is capable of encrypting the transferred information.

In conclusion it can be said that LDAP is becoming more and more important due to the commitment of the industry. On account of this factor and the improvements of architecture and functionality in version 3, LDAP actually has the potential to become an important platform for directory services in the future.

Chapter D

The Entity Relationship Model and Extensions

The purpose of this chapter is to describe and critically evaluate the basic entity relationship (ER) model. In addition, the extended entity relationship (EER) model is examined in this chapter. Concepts from both data modeling techniques have been selected for the modeling of the GEIMM according to its contribution to conceptual data modeling.

D.1 Entity Relationship Model

The graphical entity relationship (ER) model was published by Chen [1976] and has been the basis for many other models. It is reviewed in this section.

D.1.1 Description

The basic ER model is a fundamental view of data. The ER model was one of the first conceptual data models to be developed in a graphical form. It made a very significant contribution by proposing a fundamental abstraction mechanism which divides the description of problem domains into the various entities (things or objects, whether real or abstract) and the relationships (associations) between them. The problem domain that is modeled is restricted to those entities and relationships that information (data) is kept about. In the ER model, entities are shown in rectangles and relationships are shown in diamonds, with lines connecting entities to relationships and vice versa. Names given to the entities and relationships are shown within their respective rectangles and diamonds. The ER model has a very strong intuitive appeal and is very widely used.

An additional feature provided in the ER model is cardinality constraints. Cardinality constraints show limitations to the extent to which an entity may or must be associated with other entities at the other end of a relationship. A common kind of cardinality constraint is the maximum number of entities with which an entity may be associated. This is usually shown as

either one (1) or many (n), meaning more than one. This means that an entity (instance) may be associated with either "at most 1 instance" or "up to n instances" of entities of the type at the other end of the relationship. This is normally shown by placing the 1 or n near the entity rectangle at the other end of the relationship. Some versions show this with a *crows foot notation* on the line for many and a normal line for one.

Figure D-1 shows a simplified version of the meta-model for ER models according to [Sinz 1996] (p. 133). The meta-model shown is considered simplified since it does not cover any of the model's aspects discussed in the following, such existence dependence and weak entities. The attributes shown will be introduced in section D.2.

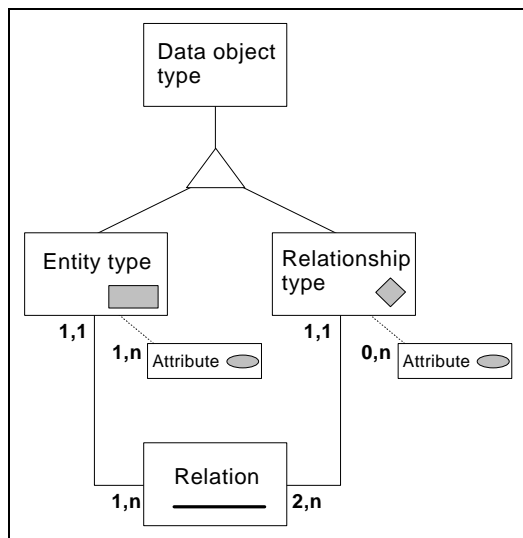


Figure D-1: Meta-model of the EER model

Figure D-1 shows that each relation connects one entity type with one relationship type. A relationship type has at least two relations to entity types. Each entity type has assigned at least one attribute and attributes may also be assigned to relationship types. Since various forms of ER models exist, Figure D-1 shows a general ER model with any cardinality possible for relations and for attributes of entity types and relationship types. The next section shows a variant of the ER model with restrictions on the cardinality of relations. For a description of further variants, refer to [Sinz 1990].

An entity can sometimes be characterized as being dependent on another entity to exist. Such a dependent entity is called a weak entity (cp. [Chen 1976]). The idea of a weak entity is concerned with information relevance rather than physical existence; i.e. when information is no longer kept about the entity on which a weak entity is dependent, it will no longer be necessary to keep information about the weak entity itself. Therefore it is a characteristic of the artificial information system world rather than of the real world that it models. A weak entity is expressed in the ER model with a double rectangle. Existence dependence is also expressed through the relationship via which the weak entity is associated with the entity on which it is dependent. Existence dependency via a relationship is expressed on the ER diagram with an arrow from the relationship diamond toward the existence dependent entity. Note that this also expresses the idea of mandatory participation in the relationship on the part of the existence dependent entity.

Chen also defined roles in relationships. An entity's role in a relationship is "the function that it performs in the relationship" (cp. [Chen 1976]). In ER diagrams, the role name is often omitted if it is otherwise implied in the context, but the role name may be explicitly added along the line connecting the entity to the relationship in which it plays that role.

The structure of an ER model is often strongly reflected in the structure of a database or set of files in a system design or implementation. The attributes are reflected in the data elements in data structures and the fields in reports and forms.

D.1.2 Evaluation

For all of its strengths, the ER model suffers from some important disadvantages. This section evaluates the basic ER model against the necessities for modeling an organizational data model.

The most serious deficiency of the *basic* ER model is in its assessment against semantic concepts. Most importantly, the basic ER model does not have sufficient richness of constructs to adequately support complex problem domains. It does not directly support aggregation, although the relationship construct is sometimes used to aggregate entities by giving the relationship a name like "component of" or "part of" which implies aggregation. Even so, it does not support aggregation of relationships. Clearly if it does not provide support for aggregation, then it does not support overlapping aggregates or multiple levels of granularity either.

As for other semantic criteria, the basic ER model performs well. In particular, the model is very minimal in semantic content which makes it easy to learn and use for simple kinds of modeling requirements.

The basic ER model also performs well for syntactic criteria. It has good one-to-one correspondence with the model's semantics, the use of lines to link entities and the use of relationship diamonds along the lines, which visually suggest the continuation of the relationship, is good. The two main syntactic difficulties are sometimes understanding the cardinality constraints in the way that they are expressed (by placing the number at the opposite end of the relationship from the entity that is constrained) and understanding which way the relationship name is to be read (which can be clarified by including role names).

As to the model's relationship to other areas (e.g. organizational modeling as in the case of GroupOrga), while the model may be partitioned, there is little guidance for doing so and no natural groupings. This has been the subject of extensive work since the original ER paper was published in 1976. The model also impressively used existing terminology, particularly as it broke new ground.

D.2 Extended entity relationship models

The entity relationship model is often extended to provide additional semantic constructs which improve its ability to describe complex situations. One of the best known, the extended entity relationship (EER) model of the Logical Relational Design Methodology (LRDM) was presented in [Teorey/Yang/Fry 1986] and is described here.

D.2.1 Description

Teorey, Yang, and Fry extended the basic ER model notation with attributes (following the work of others). Chen described attributes, but did not incorporate them into ER diagrams. Attributes are items of information (data) which describe or identify entities. The attributes in an EER model show which kinds of information are relevant to the various entities and relationships supported by an information system. In the EER model, the attributes are shown as horizontal ovals, connected by a line to the entity or relationship which they describe or identify. Identifier attributes are distinguished from descriptor attributes by underlining the name of the identifier.

Teorey, Yang, and Fry also extended the ER model by characterizing the degree of the relationships. They distinguished between unary, binary, and ternary relationships according to the number of entity types participating in the relationship (1, 2, or 3). Chen also allows relationships of degree larger than 2, but did not distinguish between them. Teorey, Yang, and Fry use a triangle, rather than a diamond to represent a ternary relationship, connecting each of the three entity types participating in it to different corners. Entity types in unary or binary relationships are connected to opposite corners of the diamond representing the relationship.

Like the ER model, Teorey, Yang, and Fry provide a notation for denoting the maximum cardinality. They use the term *connectivity* to denote a maximum cardinality constraint. A maximum cardinality of many (n) is shown by shading the portion of the relationship symbol to which the entity is connected. A maximum cardinality of 1 is shown by not shading the same portion.

Teorey, Yang, and Fry further extended the ER model by providing additional cardinality constraint information on the relationships. Sometimes, cardinality is further augmented, (or maybe combined with) a minimum number of associations, usually a choice between zero and one. In other words, the relationship is optional (minimum = 0) or mandatory (minimum = 1). They use the term *membership class* to denote the minimum cardinality. A minimum cardinality of 0 (optional) is shown with a small circle drawn through the line connecting the entity to the relationship. A minimum cardinality of 1 (mandatory) simply uses a plain line without a circle through it. In other conceptual data models, various other notations exist for minimum cardinality and sometimes it is combined with maximum cardinality in a single notation.

So far, the extensions discussed have only been modifications to the ideas already present in the ER model. The EER model also added two new ideas: subset hierarchies and generalization hierarchies. Both subset and generalization hierarchies express the "IS-A" or generalization relationship. The difference is that a subset hierarchy allows overlapping subsets while a generalization hierarchy does not. Thus, the EER model's generalization hierarchy divides an object type into disjointed or exclusive specialization, while a subset hierarchy divides it into non-disjointed or non-exclusive specialization. In the EER model, a

subset hierarchy is represented by a bold arrow from the specialized entity type to the more general entity type. A generalization hierarchy is represented by bold arrows to an elongated hexagon with the name of the partitioning attribute inside it. The hexagon in turn has a bold arrow connecting it to the more general entity type.

To summarize, the EER model of Teorey, Yang, and Fry adds notation for descriptor and identifier attributes, adds a different notation for ternary relationships (unary and binary relationships use Chen's diamond notation), uses a different notation for maximum cardinality, adds a notation for minimum cardinality, and adds two new (but similar to each other) notations for non-exclusive (subset hierarchies) and exclusive (generalization hierarchies) generalization relationships, the latter including information about the partitioning attribute.

D.2.2 Evaluation

From the point of view of semantic richness, the major contribution of the EER model over the basic ER model was the addition of support for generalization through the subset and generalization hierarchies. It additionally supports the useful idea of mandatory or optional membership in a relationship, which the basic ER model did not. It also provides a way to represent attributes and to distinguish identifier (key) attributes from descriptor attributes.

Although the authors do mention "aggregation among entities" and describe it as a special case of a binary relationship, which can be treated like a binary relationship, the EER model does not support aggregation of objects into complex objects. In addition, this does not support aggregation of relationships, the same limitation of such use for the basic ER model.

The introduction of unary relationships (degree = 1) is considered a superfluous distinction. How could there be only one party in a relationship?

As for the syntax, first, the EER model may be displayed graphically. A large problem is that the correspondence of syntax to semantics is rather uneven. The semantic problem with unary relationships is intensified because both binary and unary relationships are graphically represented with a diamond. On the other hand, a triangle is used for a ternary relationship. This could be confusing, but at least makes the degree of the relationship clear (in this case).

The graphical constructs used are very easily distinguishable from each other and thus easy to understand and learn. Subset and generalization hierarchies are easily distinguishable from ordinary relationships. Their notations are similar, which is good, since their concepts are related, but still easily distinguishable. It is hard to say what visual form would naturally respond to the idea of generalization, but the wide arrows used seem reasonable and in conjunction with placement of subtypes below supertypes gives good results. The direction of the arrow toward the supertype also seems to be a good convention.

As for the model's relationship to and usefulness in other areas, the introduction of visualization of attributes greatly improved this point. With the attributes being displayed, it is now much clearer what certain entities are to express and how they fit into the overall model. Like the ER model, no guidance is given for partitioning. The EER model also introduced some conflicts with established terminology, particularly with its introduction of the "unary relationship" and the terms "connectivity" and "membership class" in place of cardinality constraints.

Chapter E

A Continuum of Organizational Design Users

The GroupOrga project involves any employee in the organizational design process and section 5.2.5 has introduced how the varying types of users in an organization can be supported by different types of organization design applications. This chapter explores in those issues previously discussed and investigates further the GroupOrga scale presented in Table 5-1.

Form and intensity of an employee's involvement in the design process differs according to the tasks that have to be performed within the organization. This context will be examined closely in the following sections. A scale will be presented which explains the varying requirements of different user types in an organization (see Table E-1). Such a scale has already been shown in [Ott/Nastansky 1998a] (p. 568) and was later refined in [Ott/Huth 1998b] and [Ott/Huth/Nastansky 1998]. The impact of different types of users in this scale will be examined here. Moreover, and in more depth than in section 5.2.5, this chapter will address the requirements of the different users and will present technological solutions for their requirements.

Section E.1 commences with the requirement-profile of full-time organizational designers. Consequently, the requirements of those users, who are less strongly involved in the organizational design process, will be described respectively. Sections E.2 to E.5 focus on users that regularly modify the organizational structure, that occasionally adapt the model, that administer their personal data, and on users that have read-only access. The concluding section E.6 compares the various user classes and comments on their appearance under certain circumstances, such as different organizational sizes and types.

inspected later, frequency of use of organizational design applications is rather low for this user class of "intensive changes" (cp. "Frequency of use" in Table E-1).

The typical tasks of organizers of this user class are far-reaching changes in the global organizational structure, for instance initiated by a change in the core processes of the organization. If an organization is newly established, it may be necessary to complete a new design from scratch. Concrete examples of design tasks are, for instance, to move whole sub-units within the organization, or to flatten organizational structures (that is, to reduce hierarchy levels). In the case of the modeling of a virtual organization, an organizer of this type may have to integrate the various individual organizational (sub-)models into one. Hence, the typical focus of modeling for members of this user class is the hierarchical organizational model. An organizer of this class will most likely not modify bottom-level workgroup membership, particular role assignments, or individual job descriptions.

From this scenario it can be concluded that an organizer of this class intensively uses the organizational design applications and tools, which results in very specific and detailed requirements for the tools. This intensity of use is an indicator of the profundity of required changes to the organizational model from this class.

E.2 The User Class "Regular Changes"

When considering management levels, the members of the user class "regular changes" may be classified into the tactical, mid-level management (see Table E-1). Members of this class will deal with the design of single units in the organizational structure and with their subordinated units. Hence, the sub-model of all subordinated units (i.e. the unit tree) is the focus of this design and responsibility. The task of employees in this class is a regular departmental design and planning which may spread across a department's or unit's borders. Tactical management makes medium-term decisions and the literature points to a planning term of one to five years. Hence, organizational changes which are undertaken by members of this class can still be considered intentional planning. The degree of differentiation and detail in division of labor increases in comparison to strategic management levels. Since modification and planning is mainly concentrating on single units rather than on complete organizational structures, the intensity of use of organizational design applications will decrease, while the frequency of use will increase.

The tasks of units or managers in middle management include the creation of new (sub-)units when the scope of responsibility is increased, or the amalgamation of several units into one in the course of BPR projects. Other tasks may include a flattening of organizational sub-structures, the definition of new positions, or the integration of new employees into the structure. Moreover, managers of this level may have to assign authorizations and competencies to their employees, such as access rights to information and resources or rights

in the organizational design process itself. These examples explain why Table E-1 indicates an increasing frequency of use of organizational design tools for this user class.

Depending on the overall structure of an organization, managers of middle-management may lead a division, a functional unit, or even a complete partner organization within a virtual organization. Whether these tasks have to be classified top-level or bottom-level design (cp. section 5.2.2) is difficult to say. In the case of a division manager, it would rather be top-level design, while the tasks of a unit manager may already be classified as bottom-level design.

The distinction between this user class and the following is rather fluid and also differs according to the organizational culture. Some tasks may be subsumed to this class, as well as to the next one of "occasional adaptations". However, other characteristics clearly distinguish the two classes as shown in the next section.

E.3 The User Class "Occasional Adaptations"

Those members of an organization, which belong to the user class "occasional adaptations" (see Table E-1) most likely belong to the lower-level management, often entitled *operative management*. The leadership of a manager of this class directly addresses the members of a single unit which in turn has generally no further subordinated units. The number of subordinated employees, that is, the span of control varies immensely. The literature mentions a span of control between three and ten employees in *normal* situations, while other tasks may require a span of control of ninety staff.

The average span of control influences the number of managerial employees who have responsibility over others. In case of larger control spans their number decreases while with a smaller span of control the number increases, respectively. Accordingly, the number of hierarchical levels will decrease or increase, which results in either flatter or steeper organizational structures. Thus, the number of employees in the operative management level is considerably higher than in those management levels already discussed. Table E-1 shows this connection in terms of the *relative share of employees of the organization*.

In units of this organizational level short-term decisions will be taken and the planning period will most likely not exceed the one-year marker. Tasks in this class are characterized by long-term planning, but they are also spontaneous and reactive to current organizational circumstances and requirements. Division of labor is strongly differentiated and detailed.

According to this scenario, organizational design activities of this user class belong to the category of bottom-level design (cp. section 5.2.2 and [Ott/Nastansky 1997b], p. 5). Changes made to the organizational model do not affect other organizational entities (units, workgroups), but are restricted to the own organizational unit and take place within the own organizational borders.

The intensity of use of organizational design application decreases, since only the individual organizational entities are modified. Similarly, the frequency of use of such design tools generally decreases for the same reasons. In specific cases, for instance in organizations which strongly focus on workgroup structures, frequency of use of organizational modeling tools may increase, since workgroups do cross hierarchical borders and frequent remodeling may be necessary.

A clear distinction against the category "regular changes" can be made when considering internal organizational borders: while users of the class "regular changes" will design organizational structures across such borders (i.e. spanning more than one organizational unit), this is not the case in the class of "occasional adaptations". In contrast, the design of workgroups has to be assigned to both categories as Table E-1 indicates. The workgroup concept as such explains why: a workgroup does not belong to a particular organizational level and it does not belong to a specific unit—it is thus spanning organizational borders which classifies the workgroup design as tactical management task. On the other hand, the way that tasks in workgroups are accomplished (spontaneously, flexibly) would categorize the design of workgroups into the operative management.

For the design of organizational units and hierarchies, the design tasks can be classified into the respective user classes. In other words, the tactical management level can perform a general organizational design, while the operative management level performs the detailed and unit-specific modeling. This is impossible for workgroups. Due to their flexibility, the design of workgroups cannot separately be assigned to several management levels or user classes. The responsibility for a particular workgroup design must be on one organizational level, however, workgroups exist on all three levels discussed so far. The highest degree of design activities for workgroups will be found in operative and tactical management, which is why the workgroup icon in Table E-1 has been positioned in-between the two categories.

Examples of concrete tasks of employees in the "occasional adaptations" class are to modify authorizations and rights of specific employees, to define new roles, to assign roles to employees, and to create, modify, or abolish positions. The assignment of concrete employees to organizational positions is another task in this category, as well as the creation and formation of workgroups.

E.4 The User Class "Administration of One's Own Data"

This category comprises employees with no specific managerial responsibilities. Although these employees have no specific organizational design order per se, the GroupOrga concepts also includes them in the organizational design process. It has been shown that the complexity of the design tasks can be reduced if everybody is individually involved in the modeling process. This requirement can be reached, if users of this class administer their personal organizational data independently. This may, for instance, include address data, telephone-

and fax-number, email-address, etc. In the ideal case, the employee also updates this data in times of unavailability, such as before a business trip. Another example is the unavailability of an employee due to vacations, work at another location, or illness, which results in the fact that tasks in organizational workflows cannot be performed. In these cases the employee would have defined a delegation or substitution regulation himself which allows for workflows not to be halted and hindered. Such active substitution rules need not to be identified by managers only, but can also be keyed in and modified by the affected employees themselves.

Another aspect of "administration of one's own data" are skills, qualifications, and knowledge of employees. Such skills comprise seminars attended, education, certificates, language skills, knowledge about programming languages, etc. If these skills are stored and managed in an organizational enterprise knowledge base, the employees can be assigned to tasks according to their skills. Moreover, an active exchange of knowledge between employees and a task assignment in workflow systems according to skills and knowledge is practicable.

The amount of data that has to be covered by this user class seems to be tremendous for the present, but in general this information will rarely change. In addition, the changes and modifications are carried out by a large number of employees, which significantly reduces the workload of the individual worker. Thus, a low frequency of use of organizational design applications can be assumed, going along with low intensity of use, since only personal data will be affected. Another reason for low intensity of use is the fact that modifications do not affect entities beyond one's own personal data, such as unit, role, position, or workgroup entities.

While the former three classes of user types implicitly described a subset of organizational members according to their managerial level, this user class mainly addresses those at the base of an organization, but in general it comprises all organizational employees from top to bottom who administer their personal data. Table E-1 indicates this with a high relative share of employees of the organization.

E.5 The User Class "Read Access Only"

So far, every involvement in the organizational design included modification and active design. This last user class describes those accesses to the organizational EKB which are read-only. Any type of organizational information may be necessary and requested in various scenarios: information about organizational units and hierarchies, workgroup composition, membership in units or workgroups, skills of employees, and so on. Such information may be requested from persons internal and external to the organization.

When reading information in the EKB, one cannot make a valid assumption about the frequency of use. Up to here, the frequency of use was considered for active design activities—

this user class performs lookups only and no modifications. Such read-only access does not correspond with the management level and it does also not directly depend on an employee's involvement in organizational design activities elsewhere. Hence, the frequency of use can be considered constant across all organizational levels and may be added to those uses which are due to active design. The intensity of use can also hardly be compared to the other user classes, since no active modeling activities are performed. In the case of "read access only" users, based on the previously given definition for *intensity of use* the intensity would be zero.

In the course of this chapter all considerations are focused on internal design and information retrieval. In addition to this scenario, external partners might also have a need to lookup organizational information, to find the correct person to turn to, or to identify a particular workgroup manager. Examples are an organization's customers or members of other parts of a virtual organization in order to designate roles, workgroups, or persons to be included into an intra-organizational workflow. These examples show that it is possible for a restricted group of external users to gain read-access to the EKB. However, in most cases, their read access would cover only a fraction of the organization's model, and many organizations may even completely restrict access for externals.

E.6 General Considerations about the User Classes

The statements made in the previous sections may vary according to organizational size, culture, and type. Organizations to which the concepts apply need a minimum size in order to show the characteristics discussed, such as different management levels and clearly distinguished tasks. However, as Table 3-1 in section 3.1 describes, medium-sized organizations are also covered by GroupOrga concepts. For smaller organizations some classes of the scale of user types may be not applicable, specifically those of mid-level management ("regular changes"). A similar context exists for flat, decentralized, or virtual organizations, since in these forms the design activities may be more strongly polarized. Moreover, this type of organization requires in general less organizational design.

Another example of differing organizational types are those that are strongly structured, conservative, and mostly hierarchical. Such organizations may have a large share of manufacturing workers, which implies that the category "administration of one's own data" drops out completely. On the contrary, organizations that focus mainly or only on workgroups have no traditional units and thus no hierarchical structure. However, the statements about the different types of user classes still hold true, since there will be users who intensively use the design tools and others who do not. In the case of very large organizations or trusts, one will be able to identify even more managerial levels which implies that the categories of Table E-1 may be refined. The continuum described here presents a basic form, covering the most likely and common user classes.