

John Krogstie  
Andersen Consulting and  
The Norwegian University of Science and  
Technology

Arne Sølberg  
The Norwegian University of Science and  
Technology

# Information Systems Engineering:

Conceptual Modeling in a quality perspective

January 2, 2000

Information Systems Engineering:  
Conceptual Modeling in a quality perspective  
by  
John Krogstie, Andersen Consulting and NTNU  
and Arne Sølvsberg, NTNU

DRAFT January 2, 2000

*DRAFT January 2, 2000*

# Preface

This book presents an overview of the area of conceptual modeling, which is an important area within information systems engineering. It is intended for advanced teaching within information systems engineering, that should be based upon an introductory course on the subject. Such an introductory course can for instance be based on the book *Information Systems Engineering: An Introduction* by Arne Sølvsberg and David C. Kung (Springer 1993). The students should have had some experience in modeling for instance through using data modeling such as ER-modeling and process modeling such as DFD before embarking upon the course.

This book is based upon work originally presented in nine doctoral theses delivered at the information systems group at IDI, NTNU under the supervision of Professor Arne Sølvsberg. The theses are:

- HICONS: A general diagrammatic framework for hierarchical modeling [344] by Guttorm Sindre.
- A prototyping approach to validation of conceptual models in information systems engineering [236] by Odd Ivar Lindland.
- COMIS: A conceptual model for information systems [404] by Mingwei Yang.
- Explanation generation in information systems engineering [151] by Jon Atle Gulla.
- Executable conceptual models in information systems engineering [398] by Geir Willumsen.
- A configuration management approach for supporting cooperative information systems development [7] by Rudolf Andersen.
- Complexity reduction in information systems modeling [331] by Anne Helga Seltveit.
- Conceptual modeling for computerized information systems support in organizations [207] by John Krogstie.
- Conceptual modeling and composition of flexible workflow models [53] by Steinar Carlsen.

Material from the separate theses and accompanying publications is in this book reorganized according to a framework for quality of conceptual models, which have been developed primarily by Carlsen, Krogstie, Lindland, Sindre, and Sølvsberg in cooperation.

An outline of the book is presented in the end of the introduction chapter. The book covers a lot of subjects, many of which are only touched upon briefly. These are covered in more detail in the material referenced throughout the book. A course being based on the book would typically also include a set of up to date articles and other publications on specific areas of interest.

December 1999

John Krogstie  
Arne Sølvberg

DRAFT January 2, 2000

# Table of Contents

<b>Preface</b> .....	VII
<b>1. Introduction</b> .....	1
1.1 Organizations are Outcomes of Social Construction .....	2
1.2 Conceptual Modeling as Social Construction .....	5
1.3 Quality Dimensions of Conceptual Modeling .....	7
1.4 Outline of the Book .....	8
<b>2. Conceptual Modeling Languages</b> .....	11
2.1 Modeling as Hierarchical Abstraction .....	11
2.1.1 What Is a Hierarchy? .....	11
2.1.2 Four Standard Hierarchical Relations .....	17
2.1.3 Strengths and Weaknesses of Suggested Relations .....	20
2.2 Overview of Languages for Conceptual Modeling .....	23
2.2.1 An Overview of Modeling Perspectives .....	24
2.2.2 The Structural Perspective .....	25
2.2.3 The Functional Perspective .....	29
2.2.4 The Behavioral Perspective .....	32
2.2.5 The Rule Perspective .....	37
2.2.6 The Object Perspective .....	45
2.2.7 The Communication Perspective .....	50
2.2.8 The Actor and Role Perspective .....	57
2.3 Applying Several Modeling Perspectives .....	64
2.4 On the Expressiveness of CMLs .....	66
2.4.1 The Ontological Model of Information Systems .....	67
2.4.2 A Methodology Framework .....	68
2.4.3 The AMADEUS Metamodel .....	70
2.4.4 The GDR Metamodel .....	71
2.4.5 The ARIES Metamodel .....	73
2.4.6 A General Semantic Data model .....	74
2.4.7 A Brief Comparison .....	74
2.5 PPP – A Multi-perspective Modeling Approach .....	75
2.5.1 ONER – Structural and Object modeling .....	76

2.5.2	PPM - Functional, Behavioral, and Communicational Modeling . . . . .	77
2.5.3	DRL - Deontic Rule Language . . . . .	82
2.5.4	AML – Actor Modeling Language . . . . .	84
2.5.5	UID – User Interface Description Language . . . . .	87
2.6	Chapter Summary . . . . .	88
<b>3.</b>	<b>Quality of Conceptual Models . . . . .</b>	<b>91</b>
3.1	Overview and Evaluation of Existing Frameworks . . . . .	91
3.1.1	Pohl’s Framework . . . . .	92
3.1.2	FRISCO . . . . .	92
3.1.3	Overall Comparison . . . . .	93
3.2	A Framework for Quality of Conceptual Models . . . . .	94
3.3	Physical Quality . . . . .	101
3.4	Empirical quality . . . . .	103
3.5	Syntactic Quality . . . . .	105
3.6	Semantic Quality . . . . .	106
3.7	Perceived Semantic Quality . . . . .	108
3.8	Pragmatic Quality . . . . .	109
3.9	Social Quality . . . . .	110
3.10	Knowledge Quality . . . . .	111
3.11	Quality of Conceptual Modeling Languages . . . . .	111
3.11.1	Domain Appropriateness . . . . .	113
3.11.2	Participant Knowledge Appropriateness . . . . .	115
3.11.3	Knowledge Externalizability Appropriateness . . . . .	116
3.11.4	Comprehensibility Appropriateness . . . . .	116
3.11.5	Technical Actor Interpretation Apropriateness . . . . .	119
3.12	Quality of a Software Requirements Specifications (SRS) . . . . .	120
3.12.1	Physical quality of an SRS . . . . .	121
3.12.2	Empirical quality of an SRS . . . . .	122
3.12.3	Syntactic quality of an SRS . . . . .	122
3.12.4	Semantic quality of an SRS . . . . .	122
3.12.5	Pragmatic quality of an SRS . . . . .	126
3.12.6	Social quality of an SRS . . . . .	126
3.12.7	Orthogonal Aspects . . . . .	127
3.13	Chapter Summary . . . . .	128
<b>4.</b>	<b>Means for Achieving Syntactic Quality . . . . .</b>	<b>131</b>
4.1	Metalanguages for Syntax Specification . . . . .	132
4.2	Chapter Summary . . . . .	141

<b>5. Means for Achieving Semantic Quality</b> .....	143
5.1 Consistency Checking .....	144
5.1.1 Formal Verification of Data Models .....	145
5.1.2 Static Consistency Checking for PPM .....	150
5.2 Constructivity – The Fundamental Principle .....	152
5.2.1 Constructivity in BNM .....	153
5.2.2 Constructivity in PPM .....	158
5.2.3 Approaches to Constructivity Checking .....	158
5.3 Driving Questions .....	170
5.3.1 ONER-Modeling .....	171
5.3.2 Process modeling .....	172
5.3.3 Rule Modeling .....	173
5.3.4 Actor Modeling .....	174
5.3.5 Additional Metrics for Completeness and Validity ....	175
5.4 Chapter Summary .....	175
<b>6. Means for Achieving Pragmatic Quality</b> .....	177
6.1 Overview of Activities .....	177
6.1.1 Translations Facilities .....	182
6.1.2 General Translation Principles .....	184
6.2 Prototyping .....	185
6.2.1 A Taxonomy for Prototyping .....	186
6.2.2 Prototyping Languages .....	187
6.3 Execution of Conceptual Models .....	188
6.3.1 Execution Mechanisms .....	190
6.3.2 Requirements to Tools Supporting Executable Con- ceptual Modeling Languages .....	191
6.4 Tracing of Model Execution .....	191
6.4.1 Requirements to Tracing Components .....	193
6.4.2 General Tracing Principles .....	194
6.5 Explanations Generation .....	195
6.5.1 Construction of Models .....	195
6.5.2 Validation of Models .....	198
6.6 Support for Model Comprehension in PPP .....	202
6.6.1 Example .....	202
6.6.2 Overview of Techniques .....	203
6.6.3 Code Generation .....	204
6.6.4 Translating PPP Models to Ada .....	206
6.6.5 Translating PPP Models to C and Prolog .....	212
6.6.6 Filtering in PPP .....	216
6.6.7 Execution, Tracing, and Explanation Generation in PPP	217
6.6.8 Advantages of the Integrated Approach .....	231
6.7 Chapter Summary .....	232

<b>7. Means for Achieving Social Quality</b> .....	233
7.1 Tool support for model integration .....	233
7.2 Model Integration in PPP .....	238
7.2.1 Intra-project Model Integration .....	238
7.2.2 Inter-project Model Integration .....	242
7.2.3 Inter-organizational Model Integration .....	247
7.2.4 Outline of an Approach to Model Integration .....	248
7.3 Chapter Summary .....	249
<b>8. A Methodology for Conceptual Modeling</b> .....	251
8.1 Classification of Methodologies for Computerized Information Systems Support .....	252
8.1.1 “Weltanschauung” .....	252
8.1.2 Coverage in process .....	253
8.1.3 Coverage in product .....	256
8.1.4 Reuse of product and process .....	257
8.1.5 Stakeholder participation .....	258
8.1.6 Representation of product and process .....	261
8.1.7 Maturity .....	261
8.2 Conceptual Modeling in CIS Support in Organizations .....	262
8.2.1 Principles of Stakeholder Participation .....	262
8.2.2 Process Heuristics in Conceptual Modeling .....	265
8.2.3 Development Based on the Use of Conceptual Modeling .....	269
8.3 Management of Change .....	286
8.3.1 Version and Configuration Management .....	287
8.3.2 Way of Working .....	295
8.4 Use of Viewspec in Modeling .....	298
8.4.1 Inserting Modeling Statements .....	302
8.4.2 Inclusion of Changes .....	305
8.5 Chapter Summary .....	308
<b>A. Evaluating OMT Using the Quality Framework</b> .....	311
A.1 Evaluation of StP/OMT .....	312
A.1.1 Language Quality .....	312
A.1.2 Potential for Creating Models of High Quality .....	314
<b>B. Algorithms</b> .....	317
B.1 Static Consistency Checking for PPM .....	317
B.2 Constructivity Checking in PPM .....	318
<b>C. Mathematical Symbols</b> .....	325

<b>D. Terminology</b> .....	329
D.1 Time .....	329
D.2 Phenomena .....	330
D.3 State and Rules .....	332
D.4 Data, Information, and Knowledge .....	333
D.5 Language and Models .....	335
D.6 Actors and Activities .....	337
D.7 Systems .....	339
D.8 Social Construction .....	341
D.9 Methodology .....	341
D.10 Abbreviations .....	344
<b>Bibliography</b> .....	347

DRAFT January 2, 2000

# List of Figures

1.1	Social construction in an organization . . . . .	4
1.2	The model quality framework of Lindland et al. (From [239]) . . . . .	8
2.1	Six graphs . . . . .	14
2.2	Two graphs with hierarchical tendencies . . . . .	15
2.3	Two general digraphs . . . . .	16
2.4	Two weighted digraphs . . . . .	16
2.5	Hierarchical and non-hierarchical relations . . . . .	17
2.6	Association with a single child . . . . .	21
2.7	Example of a GSM model . . . . .	26
2.8	A conceptual graph linked to a semantic network (From [352]) . . . . .	28
2.9	Symbols in the DFD language . . . . .	29
2.10	Symbols in the transformation schema language . . . . .	30
2.11	Symbols in the real-world modeling language . . . . .	32
2.12	Symbols in the state transition modeling language . . . . .	32
2.13	Example of a state transition model . . . . .	33
2.14	Decomposition mechanisms in Statecharts . . . . .	33
2.15	Activation mechanisms in Statecharts . . . . .	35
2.16	Dynamic expressiveness of Petri-nets . . . . .	36
2.17	Symbols in the ERT languages . . . . .	39
2.18	Symbols in the PID language . . . . .	40
2.19	Relationship between the PID and ERL languages (from [212]) . . . . .	41
2.20	Example of a goal hierarchy (From [356]) . . . . .	43
2.21	Example of a goal-graph (From [63]) . . . . .	44
2.22	General object model (From [396]) . . . . .	46
2.23	Symbols in the OMT object modeling language . . . . .	49
2.24	Example of an OMT object model . . . . .	50
2.25	Symbols in the OMT dynamic modeling language . . . . .	51
2.26	Conversation for action (From [400]) . . . . .	53
2.27	Main phases of action workflow . . . . .	54
2.28	Comparing communicative action in Habermas and Searle (From [92]) . . . . .	55
2.29	The pattern of transaction . . . . .	56
2.30	The symbols of the ABC-language (From [91]) . . . . .	57
2.31	Example of an ALBERT model (From [99]) . . . . .	59

2.32	Example of an actor dependency model (From [407]) . . . . .	61
2.33	Symbols in agents-role-position modeling language (From [406]) . .	62
2.34	Symbols in the OORASS role interaction language . . . . .	63
2.35	A data modeling language used for meta-modeling . . . . .	67
2.36	A metamodel of Wand and Weber's ontology . . . . .	68
2.37	A metamodel corresponding to the methodology framework . . . . .	69
2.38	The unified model in AMADEUS . . . . .	70
2.39	The metamodel of GDR . . . . .	71
2.40	Excerpts of the ARIES metamodel . . . . .	72
2.41	A general semantic data model . . . . .	73
2.42	A comparison of the unified metamodels . . . . .	75
2.43	Symbols in the ONER language . . . . .	76
2.44	Symbols in the PPM language . . . . .	77
2.45	The activities for ordering tickets in the IFIP conference . . . . .	81
2.46	Example of a PLD model . . . . .	82
2.47	Rule-hierarchies . . . . .	83
2.48	Basis for actor models . . . . .	85
2.49	Symbols in the extensions to Statecharts used in UDD . . . . .	88
3.1	Pohl's framework (From [306]) . . . . .	92
3.2	Extended framework for discussing quality of conceptual models . .	95
3.3	A simple ER-diagram . . . . .	101
3.4	Example on poor aesthetics . . . . .	105
3.5	Example of syntactic invalidity . . . . .	106
3.6	Example of syntactic incompleteness . . . . .	106
3.7	Example of semantic invalidity . . . . .	107
3.8	Coverage of this section . . . . .	113
4.1	Coverage of this chapter . . . . .	132
4.2	Portions of a meta-model for an executable DFD . . . . .	135
4.3	Meta-model for execution of conceptual modeling languages . . . . .	136
4.4	Incorporating the use of rules and actors in the PPP meta-model .	139
4.5	Syntactical completeness checks of PPM model . . . . .	141
5.1	Coverage of this chapter . . . . .	144
5.2	A compact unifiability digraph . . . . .	148
5.3	The i/o condition for the process network for $P_1$ of the IFIP ticket booking activities . . . . .	151
5.4	A Behavior Network, before and after abstraction . . . . .	154
5.5	An STD for the network . . . . .	155
5.6	A decomposition of a process $P_1$ . . . . .	159
5.7	Two ways for consistency checking of the decomposition of $P_1$ . . . .	160
5.8	Two process networks which may produce run-time errors . . . . .	161
5.9	Conducting possible execution sequences of a process . . . . .	162
5.10	The execution life cycle and the operation groups of a process . . . .	165

5.11	The IFIP ticket booking activities with canonical ports . . . . .	166
5.12	The STD of the process network for $P_1$ in the IFIP ticket booking activities . . . . .	167
5.13	The canonical port structure and the STD of the process network . . . . .	168
5.14	The canonical port structure and the partially feasible STD of a process network . . . . .	169
5.15	Synthesis for the Properties of the Process Network . . . . .	170
5.16	Construction of the port structures through the STD in Figure 13 . . . . .	171
6.1	Coverage of this chapter . . . . .	178
6.2	Example of a language filter . . . . .	179
6.3	Example of a model filter . . . . .	179
6.4	The example written in the GSM language . . . . .	180
6.5	The architecture of a general translation facility . . . . .	183
6.6	A general architecture of a tracing component . . . . .	194
6.7	An explanation showing what a process looks like . . . . .	197
6.8	Illegal constructions in PrM . In (a) an a posteriori rule is violated, in (b) an a priori rule . . . . .	198
6.9	Portions of a PrM model for the banking system . . . . .	204
6.10	A decomposition of transaction processing . . . . .	205
6.11	A PLD describing process P1.2 . . . . .	206
6.12	Integrating techniques for the support of model comprehension . . . . .	207
6.13	The overall translation strategy for generating Ada code (From [240]) . . . . .	207
6.14	Some selected translation rules the “PPP-Ada assistant” (From [240]) . . . . .	208
6.15	An overview of the Ada translation process . . . . .	209
6.16	An early version of process P1.2 and its corresponding PLD model . . . . .	210
6.17	Test data for the execution session . . . . .	213
6.18	Execution trace of the example . . . . .	214
6.19	Parts of the temporal database after execution . . . . .	215
6.20	(a) A model view resulting from a component abstraction (b) Ports abstracted away and layout is improved . . . . .	216
6.21	Architecture of the explanation generation system . . . . .	217
6.22	Generated Ada code from a PLD indicating the insertion of probes . . . . .	220
6.23	A small portion of a trace graph for execution of the PLD of P1.2 . . . . .	221
6.24	Functions defined to request informations from the trace . . . . .	227
6.25	Deep explanation for the question “ <i>Why was my withdrawal rejected?</i> ” . . . . .	228
6.26	Operator sturcture of deep explanation for the question “ <i>Why do you need New_amount?</i> ” . . . . .	230
7.1	Coverage of this chapter . . . . .	234
7.2	Viewpoint resolution . . . . .	235
7.3	Strategy for viewpoint analysis . . . . .	236
7.4	Relationships in IBIS . . . . .	237
7.5	Rules based on CATWOE analysis . . . . .	240

7.6	Goal-hierarchy extended from CATWOE analysis	242
7.7	IBIS-network on the issue of system platform	243
7.8	Pruned goal-hierarchy after argumentation process	244
7.9	ONER-model for the IFIP-case	245
7.10	ONER-model for the traditional IFIP-case (From [404])	246
8.1	Scale of influence and power	259
8.2	Co-generative learning in systems development	263
8.3	The SPEC-cycle for modeling	266
8.4	Overall actor model of ISDO95	271
8.5	Actor model of project-participants	274
8.6	PPM describing CFP-distribution at the actor-level	278
8.7	PPM describing CFP-distribution at the role-level	279
8.8	PPM describing reception and distribution of papers based on $K_2$	282
8.9	PPM describing distribution of papers based on $K_1$	283
8.10	PPM describing the paper handling	285
8.11	Conceptual modeling in a set of integrated projects	287
8.12	Version graph for system $S$	288
8.13	General hierarchy	289
8.14	Component structure graph	290
8.15	Check-out contract for Address register 1.3	296
8.16	Viewspecs are filter related to their originating models	299
8.17	a) A data model and associated viewspecs b) a process model and associated viewspecs	300
8.18	Viewspecs are variant related to their originating models	301
8.19	Viewspecs are context related to their originating models	301
8.20	Local workspaces	302
8.21	Revisions of models	304
8.22	Resolving the conflict immediately after a viewspec is updated	305
8.23	Building a new revision based on updated viewspecs	306
8.24	The merge relations	308
D.1	Time and duration	330

# List of Tables

2.1	A data flow diagram taxonomy of real-world dynamics . . . . .	31
3.1	Framework for model quality . . . . .	100
3.2	A taxonomy of graph aesthetics (From [363]) . . . . .	104
3.3	A taxonomy of constraints for graph layout (From [363]) . . . . .	105
4.1	Relationships in meta-model for general execution . . . . .	137
5.1	The analysis of the path without loops . . . . .	156
5.2	The analysis of the path with only the t3-loop . . . . .	156
5.3	The analysis of the path with both loops . . . . .	157
6.1	<i>EML</i> characterizations for some elements of the PPP conceptual model . . . . .	224
6.2	Plan operators needed to generate the history deep explanation. . . . .	226
6.3	Additional operators for generating the input justification. . . . .	226
6.4	Instantiated <i>cause</i> operator. . . . .	227
8.1	Links between users and developers . . . . .	260

DRAFT January 2, 2000

# 1. Introduction

*Conceptual models* are normally constructed during the problem analysis and requirements specification process of information systems engineering. They are often used as a starting point for constructing and implementing the information system.

A conceptual model can be defined as

a model of the phenomena in a domain at some level of approximation, which is expressed in a semi-formal or formal language.

In this text, we apply the following limitations:

- The languages for conceptual modeling are mostly diagrammatic with a limited vocabulary. The main symbols of the languages represent *concepts* such as states, processes, entities, and objects. We mostly use the terms 'phenomena' and 'phenomena classes' instead of 'concepts' in this text since the word 'concept' is used in many different meanings in natural language.
- Conceptual models are primarily used as an intermediate representation for development and maintenance of information systems. We recognize that conceptual modeling languages may be useful also for other purposes such as organizational modeling or process modeling when there is no immediate system implementation in mind. We will not treat such usage of modeling languages in detail in the book.
- The conceptual languages presented in this text are meant to have general applicability, that is, they are not made specifically for the modeling of a limited area.

We combine the use of conceptual models with the philosophical outlook that reality is socially constructed. Most of the current modeling approaches are based on an objectivistic ontology, e.g., “the real world consists of entities and relationships” [62, 203]. However, this assumption is not shared by everybody, in [348] for instance, it is focused on that what is modeled is some persons *perception* of the “real world”.

The present practice of modeling includes a large element of subjectivity [232]. This subjectivity exists whether or not the data-focused approaches uses 'entities', 'objects', or 'phenomena' as main concepts. If entities are taken to have real-world existence, then the participants in the modeling effort must

choose from the infinitely large number of entities that exist, only those entities that are relevant and suitable for inclusion in the model. Consequently, the process of creating such a model is not value-free and the resulting conceptual model is not unbiased. If real-world existence of the relevant phenomena is *not* assumed, then the entities are, by definition, created subjectively by the participants in the modeling effort in order to understand the situation at hand. In either case, the conceptual model serves only as an interpretation of “reality”.

Thus, in both cases, it will be useful to admit to this subjectivity and allow several models to co-exist, even if only one of them will be used for building the information systems. There are indeed approaches that acknowledge several realities. Some approaches are grounded in object orientation [165, 312]. Another approach is Multiview [16], which has a constructivistic worldview and uses traditional conceptual languages as an important part of the methodology. A similar attempt to integrate Soft system methodology (SSM) [61] and software engineering approaches is reported in [97].

Even if traditional conceptual modeling languages may support a constructivistic worldview, they usually do not have explicit constructs for capturing differing views directly in the model and making these available to those who use the model. They neither have the possibility to differentiate between the rules of necessity and deontic rules.

Conceptual modeling languages are biased towards a particular way of perceiving the world:

- The languages have constructs that force both analysts and users to emphasize some aspects of the world and neglect others.
- The more the analysts and users work with one particular language, the more their thinking will be influenced by this, and their awareness of those aspects of the world that do not fit in may consequently be diminished recall the Sapir-Whorf hypothesis which states that a person’s understanding of the world is influenced by the language he uses [354].
- For the types of problems that fit well with the approach, neglecting features that are not covered may even have a positive effect, because it becomes easier to concentrate on the relevant issues. However, it is hard to know what issues are relevant. Different issues within a problem situation may be relevant for different people at the same time.

## 1.1 Organizations are Outcomes of Social Construction

Organizations are constantly under the pressure of *change* from internal as well as external forces. Most organizations are supported by a *portfolio of application systems* that likewise have to be changed, often rapidly, for the organization to be able to keep up and extend its activities. The portfolio usually consists of a set of individual, but highly integrated application systems whose long-term evolution should be coordinated as a whole.

Organizational change may be viewed from different philosophical points of view. Two common sets of assumptions are the objectivistic belief system and the constructivistic belief system [149]. They may be distinguished through differences in ontology (what exists that can be known), epistemology (what relationship is there between the knower and the known), and methodology (what are the ways of achieving knowledge).

Organizations are made up of individuals who perceive the world differently from each other. The constructivistic view is that an organization develops through a process of *social construction*, based on its individuals' perception of the world. In the objectivistic view [149] there exists only one reality, which is measurable and essentially the same for all. The objectivistic belief system can simplistically be said to have the following characteristics:

- The ontology is one of realism, asserting that there exist a single reality which is independent of any observer's interest in it and which operates according to immutable natural laws. Truth is defined as that set of statements whose natural or intended model are isomorphic to reality.
- The epistemology is one of dualistic objectivism, asserting that it is possible, indeed mandatory, for an observer to exteriorize the phenomenon studied, remaining detached and distant from it and excluding any value considerations from influencing it.
- The methodology is one of interventionism, stripping context of its contaminating influences so that the inquiry can converge on truth and explain the things studied as they really are and really work, leading to the capability to predict and to control.

The constructivistic belief system has the following characteristics according to [149]:

- The ontology is one of relativism, asserting that there exist multiple socially constructed realities unguided by any natural laws, causal or otherwise. "Truth" is defined as the best-informed and most sophisticated construction on which there is agreement.
- The epistemology is subjectivistic, asserting that the inquirer and the inquired-into are interlocked in such a way that the findings of an investigation are the literal creation of the inquiry process.
- The methodology is hermeneutical and involves a continuing dialectic of iteration, analysis, critique, reiteration, reanalysis, and so on, leading to the emergence of a joint construction and understanding among all the stakeholders.

Many features of the constructivistic paradigm have emerged from "hard natural sciences such as physics and chemistry. The argument for the new paradigm can be made even more persuasively when the phenomena being studied involve human beings, as in the "soft social sciences. Much of the theoretical discussion in the social sciences is at present dedicated to analyzing constructivism and its consequences [75]. The idea of reality construction has

been a central topic for philosophical debate during the last two decades, and has been approached differently by French, American, and German philosophers.

Many different approaches to constructivistic thinking have appeared, although probably the most influential one is that Berger and Luckmann [26]. Their insights will be used as our starting point. Their view of the social construction of reality is based on Husserl's phenomenology. Whereas Husserl was primarily a philosopher, Schutz [326] took phenomenology into the social sciences. From there on it branched into two directions: Ethnomethodology, primarily developed by Garfinkel [130], and the social constructivism of Berger and Luckmann. Whereas ethnomethodology is focused on questioning what individuals take as given in different cultures, Berger and Luckmann developed their approach to investigate *how* these presumptions are constructed.

Organizations are realities constructed socially through the joint actions of the social actors in the organization [136], as illustrated in Fig. 1.1.

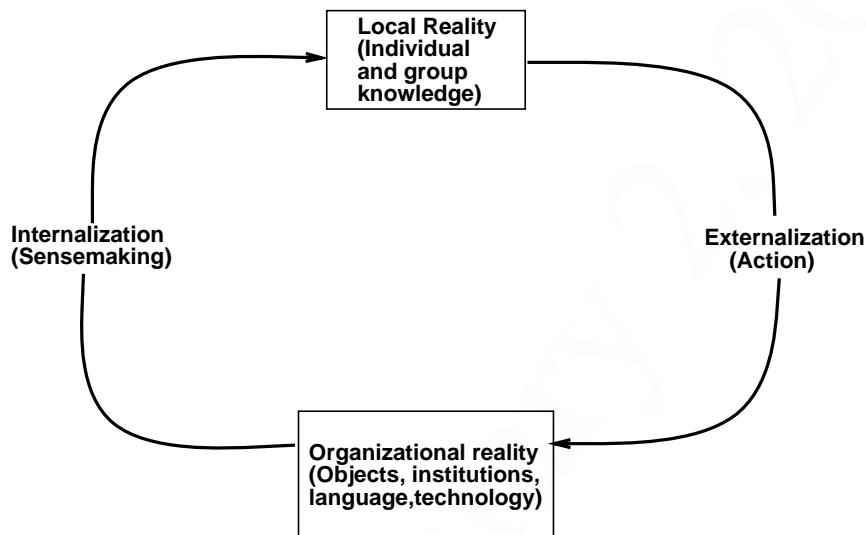


Fig. 1.1. Social construction in an organization

An organization consists of individuals who view the world in their own specific way, because each of them has different experiences arising from work and other activities. The *local reality* refers to the way an individual perceives the world in which he or she acts. The local reality is the way the world is for the individual; it is the everyday perceived reality of the individual social actor. Some of this local reality may be made explicit and talked about. However, a lot of what we know is *tacit*. When the social actors of an organization

act, they *externalize* their local reality. The most important ways in which social actors externalize their local reality are by speaking and constructing languages, artifacts, and institutions. What they do, is to construct *organizational reality* by making something that other actors have to relate to as being part of the organization. This organizational reality may consist of different things, such as institutions, language, artifacts, and technology. Finally, *internalization* is the process of making sense out of the actions, institutions, artifacts etc. in the organization, and making this organizational reality part of the individual local reality. This linear presentation does not mean that the processes of externalization and internalization occur in a strict sequence. Externalization and internalization may be performed simultaneously. Also, it does not mean that only organizational reality is internalized by individuals. Other externalizations also influence the construction of the local reality of an individual.

Changing the computerized information system (CIS) support in an organization, for instance by introducing new application systems may also be looked upon as a process of social construction. This view has received increasing interest in the information systems community in recent years [121, 235, 340, 400]. This outlook is adopted in this book, especially when focusing on the creation and maintenance and further development of conceptual models in connection with improving the computerized information system support of an organization. This does not mean that we are ignorant of the more technical aspects of computerized information systems support. Constructed realities are often related to, and also often inseparable from, tangible phenomena.

## 1.2 Conceptual Modeling as Social Construction

The construction of a conceptual model of “reality” as it is perceived by someone is partly a process of externalization of parts of this person’s internal reality, and will in the first place act as organizational reality for the audience of the model. This model can then be used in the sense-making process by the other stakeholders, internalizing the views of the others if they are found appropriate. This internalization is based on pre-understanding, which includes assumptions implicit in the languages used for modeling. The language in turn is learned through internalization.

After reaching a sufficiently stable shared model one might wish to externalize this in a more material way, transferring it to the organization in the form of computer technology. Here a new need for internalization of the technology is needed for the CIS to be useful for the part of the organization that is influenced by it. Also here, it should be possible to utilize the conceptual models to understand what the CIS does, and most importantly, *why* it does it. Making sense of the technology is important to be able to change it,

and the conceptual models already developed can act as a starting point for additional maintenance efforts on the CIS when deemed necessary.

It should be noted that the abilities and opportunities for the different social actors in the organization to externalize their local reality will differ in several ways. Since the languages and types of languages used are often predefined when a decision to create an application system is made, persons with long experience in using these kinds of languages will have an advantage in the modeling process. This applies especially to the specialists on computer technology. This is not necessarily bad, for if they did not have this knowledge it would not be interesting to include them in the development process in the first place. Rather, it is important to be aware of this difference, to avoid the most apparent dangers of model monopoly as discussed by Bråten [38]. What is also apparent is that some persons in the organization have a greater possibility to externalize their reality than others, both generally (the financiers of an endeavor will for instance usually be in a position to bias a solution in their perceived favor) and specifically, by the use of certain modeling techniques. Gjersvik has for instance investigated how the way management perceive the world can be more easily externalized in a CIS than the way shop-floor workers perceive the world [136].

The use of conceptual models constructed as part of the development and maintenance of application systems has been discussed by several researchers [44, 93, 186, 217, 393]. This discussions can be summarized as follows:

- Representation of systems and requirements: The conceptual model represents properties of the problem area and perceived requirements for the information system. A conceptual model can give insight into the problems motivating the development project, and can help the systems developers and users understand the application system to be built. Moreover, by analyzing the model instead of the business area itself, one might deduce properties that are difficult if not impossible to perceive directly since the model enables one to concentrate on just a few aspects at a time.
- Vehicle for communication: The conceptual model can serve as a means for sense-making and communication among stakeholders. By hopefully bridging the realm of the end-users and the CIS, it facilitates a more reliable and constructive exchange of opinions between users and the developers of the CIS, and between different users. The models both help and restrict the communication by establishing a nomenclature and a definition of phenomena in the modeling domain.
- Basis for design and implementation: The conceptual model can act as a prescriptive model, to be approved by the stakeholders who specify the desired properties of a CIS. The model can establish the content and boundary of the area under concern more precisely. During design and implementation of the CIS, the relevant parts of the model guide the development process. Similarly, the design and implementation might afterwards

be tested against the model to make sure that the different representations are consistent. When the model is formal and contains sufficient detail, it is often possible to produce the application system more or less directly from the model.

- Documentation and sensemaking: The conceptual model is an easily accessible documentation of the CISs that are in use in the organization. Due to its independence of the implementation, it is less detailed than other representations, while still representing the basic functionality of the system. Compared to manually produced textual documentation, the conceptual model is easier to maintain since it is constructed as part of the process of developing and maintaining the application system in the first place. With the introduction of more flexible methodologies and tool support, conceptual models are also likely to be used in reverse engineering and re-engineering, and when reusing artifacts constructed in connection with other application systems.

Summing up, a conceptual model is used both for communication and representation, and faces demands from both social and technical actors. As a consequence of this duality, requirements for conceptual modeling languages and modeling techniques will pull in opposite directions.

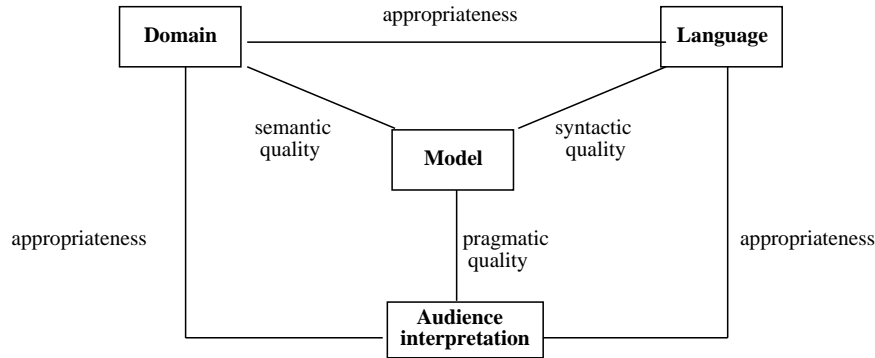
### 1.3 Quality Dimensions of Conceptual Modeling

We have organized this book according to a framework that has been developed for understanding the quality of conceptual modeling and modeling languages. The main structure of this framework, as originally presented in [239], is illustrated in Fig. 1.2. The basic idea is to evaluate the quality of models along three dimensions by comparing sets of statements. These sets are:

- $\mathcal{M}$ , the externalized model, that is, the set of all the statements explicitly or implicitly made in the model.
- $\mathcal{L}$ , the language extension, that is, the set of all statements which can be made according to the vocabulary and grammar of the modeling languages used.
- $\mathcal{D}$ , the modeling domain, that is, the set of all statements that can be stated about the problem at hand.
- $\mathcal{I}$ , the audience interpretation, that is, the set of all statements which the audience (i.e. various actors in the modeling process) think that the model comprises.

Model quality is defined using the relationships between the model and the three other sets:

- *Syntactic quality* is the degree of correspondence between model and language extension, that is, the set of syntactic errors is  $\mathcal{M} \setminus \mathcal{L}$ .



**Fig. 1.2.** The model quality framework of Lindland et al. (From [239])

- *Semantic quality* is the degree of correspondence between model and domain. If  $\mathcal{M} \setminus \mathcal{D} \neq \emptyset$  the model contains invalid statements; if  $\mathcal{D} \setminus \mathcal{M} \neq \emptyset$  the model is incomplete. Since total validity and completeness are generally impossible, the notions of *feasible validity* and *feasible completeness* are employed.
- *Pragmatic quality* is the degree of correspondence between model and audience interpretation (i.e., the degree to which the model has been understood). If  $\mathcal{I} \neq \mathcal{M}$ , the comprehension of the model is not completely correct. Usually, it is neither necessary nor possible that the whole audience understand the entire conceptual model – instead, each group in the audience should understand the part of the model which is relevant to them. *Feasible comprehension* is defined along the same lines as feasibility for validity and completeness.

In addition to these primary quality concerns, the correspondence between domain and language, between domain and audience interpretation, and between language and audience interpretation may affect the model quality indirectly. These relationships are all denoted *appropriateness* Fig. 1.2. We also differentiate between the quality goals and means for achieving these goals.

## 1.4 Outline of the Book

The quality framework is presented in more detail in Chap. 3. In Chap. 2 we first give an overview of the different mechanisms and perspectives used in conceptual modeling. Means to support quality on different levels are then discussed in Chapters 4–7. Aspects of a modeling methodology based on the use of the framework is outlined in Chap. 8. An example on how to apply the framework to evaluate a modeling approach is given in Appendix A.

Mathematical symbols are explained in Appendix C, and the terminology is explained in Appendix D.

The main example used throughout the book concern an information system for supporting the arrangement of a scientific conference. It is a variant of the widely used “IFIP working conference” example which was originally presented in 1982 [285] and has been used since then to illustrate a variety of information system modeling approaches.

In this book, the example describes a real situation, since we have been developing a system of this sort using some of the described languages. Thus, the case description is not the traditional one, but is taken from an actual project where we developed, used and maintained a support system for the IFIP WG8.1 Working Conference on Information Systems development for Decentralized Organizations (ISDO95), which took place in Trondheim, Norway, in August 1995 [349].

IFIP is the acronym for International Federation for Information Processing. An IFIP working conference is an international conference that provides an opportunity for the computer scientists from IFIP member countries to discuss and interchange research results and new ideas on selected research fields.

The management of such a conference is usually done by two cooperating committees. The *program committee* (PC) handles the contents of the conference, say, the reviewing of papers, comprising sessions and tutorials, etc. The *organizing committee* (OC) handles the administration work, e.g. sending out invitations, registration of attendants, arranging time and places for sessions, dealing with financial matters, etc.

We will return to this and other examples throughout the book, adding more detail to the case description as needed.

DRAFT January 2, 2000

## 2. Conceptual Modeling Languages

In this chapter, we will give an overview of mechanisms and perspectives used in conceptual modeling. We will first look upon modeling in general as hierarchical abstraction. Then we will present different modeling languages according to the main phenomena they describe, and discuss the usefulness of the possibility of applying several such perspectives at the same time in an integrated manner. An approach supporting all the described perspectives, PPP, is presented in the end of the chapter.

### 2.1 Modeling as Hierarchical Abstraction

A conceptual model is an abstraction. One mechanism for abstraction used in many of the existing languages for conceptual modeling is the use of *hierarchies*. The importance of hierarchical abstractions is based on the following assumptions.

- Hierarchies are essential for human understanding of complex systems.
- Thinking in terms of hierarchical constructs such as aggregation and generalization appears to be very natural.
- Information systems are complex systems because they must reflect the part of the world they process information about,
- A proper support for hierarchical constructs is an essential requirement throughout the entire information system development and maintenance process.

#### 2.1.1 What Is a Hierarchy?

Here we will discuss what a hierarchy is in more detail. The first subsection discusses the possibilities for arriving at a precise definition of the term 'hierarchy' in terms of graph theory. As will be seen, however, it is difficult to come up with a definition which is precise and at the same time satisfactory. The second subsection thus argues that being hierarchical is very much a question of degree.

**Hierarchical: A Question of Definition.** In [147, 341] a hierarchy is defined rather vaguely as any collection of Chinese boxes (where each box can contain several smaller boxes). [261] refrains from giving any exact definition of what a hierarchy is, but lists some properties which all hierarchies should have, namely “vertical arrangement of subsystems which comprise the overall system, priority of action or right of intervention of the higher level subsystems, and dependence of the higher level subsystems upon the actual performance of the lower levels. More precise definitions are given in [17] and [49].

Some works also identify different kinds of hierarchical systems. [17] distinguishes formally between *division hierarchies* and *control hierarchies*. [261] operates with three notions of hierarchical levels, namely *strata* (levels of description or abstraction), *layers* (levels of decision complexity), and *echelons* (organizational levels). All in all it seems that the word “hierarchy” may be used in rather different ways by different authors — as stated in [352] some use it indiscriminately for any partial ordering, whereas the above definitions require something more.

It is difficult to come up with a strict and precise definition distinguishing hierarchical systems from other systems. However, since it is important to make clear what we are talking about, we need some kind of definition of what a hierarchy is.

To this end it is illuminating to look at the definition presented by Bunge in [49]:

$\mathbf{H}$  is a hierarchy if and only if it is an ordered triple  $\mathbf{H} = \langle \mathbf{S}, \mathbf{b}, \mathbf{D} \rangle$  where  $\mathbf{S}$  is a nonempty set,  $\mathbf{b}$  a distinguished element of  $\mathbf{S}$  and  $\mathbf{D}$  a binary relation in  $\mathbf{S}$  such that

1.  $\mathbf{S}$  has a single beginner,  $\mathbf{b}$ . (That is,  $\mathbf{H}$  has one and only one supreme commander.)
2.  $\mathbf{b}$  stands in some power of  $\mathbf{D}$  to every other member of  $\mathbf{S}$ . (That is, no matter how low in the hierarchy an element of  $\mathbf{S}$  may stand, it is still under the command of the beginner.)
3. For any given element  $\mathbf{y}$  of  $\mathbf{S}$  except  $\mathbf{b}$ , there is exactly one other element  $\mathbf{x}$  of  $\mathbf{S}$  such that  $\mathbf{D}\mathbf{x}\mathbf{y}$ . (That is, every member has a single direct boss.)
4.  $\mathbf{D}$  is transitive and antisymmetric.
5.  $\mathbf{D}$  represents (mirrors) domination or power. (That is,  $\mathbf{S}$  is not merely a partially ordered set with a first element: the behavior of each element of  $\mathbf{S}$  save its beginner is ultimately determined by its superiors.)

As pointed out by Bunge, this definition does two things:

- The first four points state what a hierarchy is in a graph-theoretic sense, namely a strict tree-structure.<sup>1</sup>
- The fifth point introduces an extra requirement on the nature of the relations (i.e. edges) between the nodes, namely that they represent domination or power.

Thus, Bunge makes the important point that whether something is a hierarchy or not cannot be determined by graph-theoretic considerations alone. However, Bunge’s definition might be a little too strict:

- the graph-theoretic demands are very limiting. In real life it often happens that a node can have more than one boss, or even that there are cycles in the graph, and still many people might consider the system to be of a hierarchical nature.
- the requirement that nodes are related by domination severely limits the scope of hierarchical systems — as stated by Bunge himself reciprocal action, rather than unidirectional action, seems to be the rule in nature (which leads Bunge to the conclusion that it is misleading to speak of hierarchies in nature: “Hierarchical structures are found in society, e.g. in armies and in old-fashioned universities; but there are no cases of hierarchy in physics or in biology”). Since one might want to be able to model practically anything, we have to recognize other kinds of hierarchical relations in addition to domination or power.

To achieve more generality, we will allow more general graphs to be considered as hierarchical systems. But it will also be useful to have a specific term for those systems which satisfy the rather restrictive requirements stated above. Below we will use the following terminology:

- Strictly hierarchical graph: a digraph whose underlying graph is a tree, and for which there is one specific vertex  $\mathbf{b}$  from which all other vertices can be reached (this is the distinguished element of Bunge’s definition).
- Weakly hierarchical graph: a connected acyclic digraph which deviates from the former in that there is no distinguished element and/or in that its underlying graph is cyclic. Mathematically, this class of graphs are called DAGs (directed acyclic graphs).
- Cyclic hierarchical graph: a cyclic digraph.

Obviously, the latter two notions should be used carefully – there is no point in calling any graph a hierarchy. Thus, even if we allow some DAGs, and maybe even some cycles, we should still require that a graph is pretty close to being a *strict* hierarchy if we call it hierarchical.

---

<sup>1</sup> To be precise, it is an open-ended directed graph whose *underlying* graph (i.e. the undirected parallel of a directed graph) is a tree, since trees, graph-theoretically, are undirected graphs. For an introduction to graph theory, including definitions of graphs (directed and undirected), trees, and underlying graphs, see for instance [399].

The meaning of our suggested terminology can be visualized by Fig. 2.1.

Of these graphs (a) would not be a hierarchy because it is not connected (but it might be two hierarchies), and (b) would not be a hierarchy because the edges are not directed. (c) on the other hand, is the kind of graph which satisfies Bunge’s requirements, i.e. it is a strict hierarchy. (d) would not be accepted as a hierarchy according to Bunge’s definition because the underlying graph has a cycle (i.e. the middle element at the lowest level has two bosses), but we might call it a weak hierarchy. Similarly, (e) does not have one distinguished element – there are two elements on top which do not control each other. This could also be a weak hierarchy in our terminology. Finally, (f) contains a cycle and is thus clearly excluded by Bunge’s definition, whereas we could call it a cyclic hierarchy (because although containing a cycle, the graph is not very far from being a strict hierarchy).

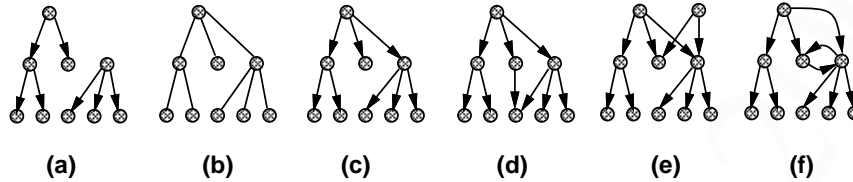


Fig. 2.1. Six graphs

The motivation for removing some of the restrictions of Bunge’s definition is that we want to be as general as possible, and clearly many people might feel that systems are hierarchical even when they are not strictly hierarchical. This is exemplified by the two graphs of Fig. 2.2, where (a) breaks the single boss requirement, and (b) breaks the antisymmetry requirement. If the edges denote the relation like “is the boss of”<sup>2</sup>, it is still likely that both systems will be considered as hierarchical. Moreover, our definition has not required that the relations denoted by the edges be transitive. Clearly, most hierarchical relations are transitive (e.g. if A is the boss of B, and B the boss of C, it is also true that A is the boss of C), but there is no point in rejecting cases where this does not apply (e.g. if A is the parent of B, and B the parent of C, it will not be true to say based on this that A is the parent of C, and still people might feel that “parent of” is a typically hierarchical relation).

Having loosened up Bunge’s graph-theoretic restrictions it might seem that we may end up calling any kind of directed graph a hierarchy. However, this is not our intention. We still need some requirement corresponding to the fifth point of Bunge’s definition. *Dominance* or *power* is too narrow. Still we need to make some restriction on the semantics of the relation denoted by the

<sup>2</sup> In (b) Bo and Dan might for instance supervise two different business areas, both working on both.

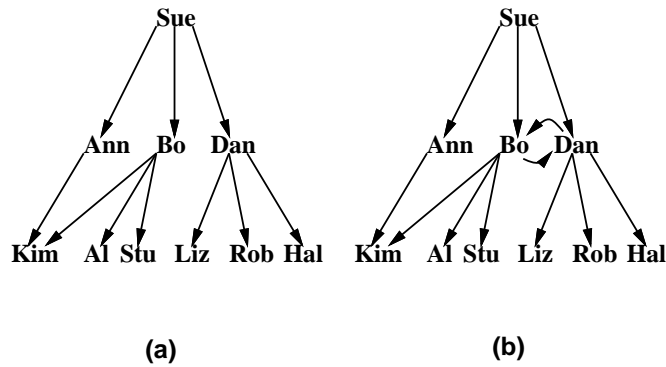


Fig. 2.2. Two graphs with hierarchical tendencies

edges. This is not easy, and can only be dealt with when we have discussed in the next subsection what it means to be more or less hierarchical.

**Hierarchical: a Question of Degree.** If one takes everything into consideration, a model of a situation will be a general graph rather than a hierarchy. Depicting something as a strict hierarchy will therefore be a simplification, and this simplification may be more or less appropriate, depending on the *distance* between the hierarchy depicted and the actual situation as it is perceived.

How can such *distance* be measured? Given a general connected digraph, how would you answer the question “How close is this graph to being a hierarchy?” From a general connected digraph, a strict hierarchy can be obtained by cutting some edges, so a first attempt could be to count how many edges one would have to cut, or rather the ratio of cut edges to the total number of edges. With this approach we would say that the digraph of Fig. 2.3(a) is obviously closer to being hierarchical than the one of (b), since in (a) we have to cut only 2 out of 10 edges, whereas in (b) we have to cut 4 out of 10. However, the soundness of this kind of computation relies on the assumption that all links are equally important, which need of course not be true. To deal with other cases, each edge must be assigned a weight, signaling its *importance*. In Fig. 2.4 weights have been assigned, and now it is (b) which is closest to being a hierarchy, because the minimum cut has a total of only 5 weights, whereas the same number for (a) is 12.

If the given relation is “is the boss of” weights should reflect the degree of influence that the supervisor has over the subordinate; if the graphs are call graphs for a software system, importance will depend on the frequency of calls. Generally, importance is a very problematic notion, and we will not enter any further discussions of it here. Instead we will only conclude that:

- A general directed graph can be more or less hierarchical
- Its closeness to a strict hierarchy is dependent on:

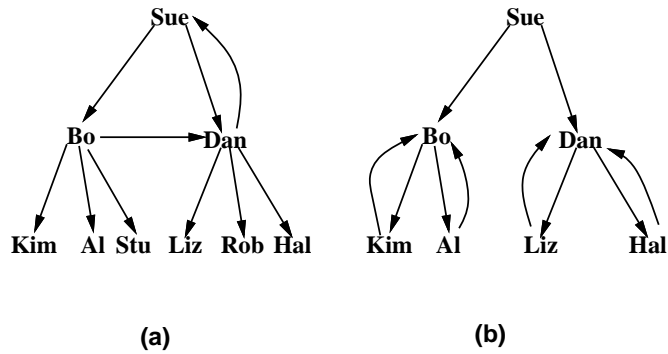


Fig. 2.3. Two general digraphs

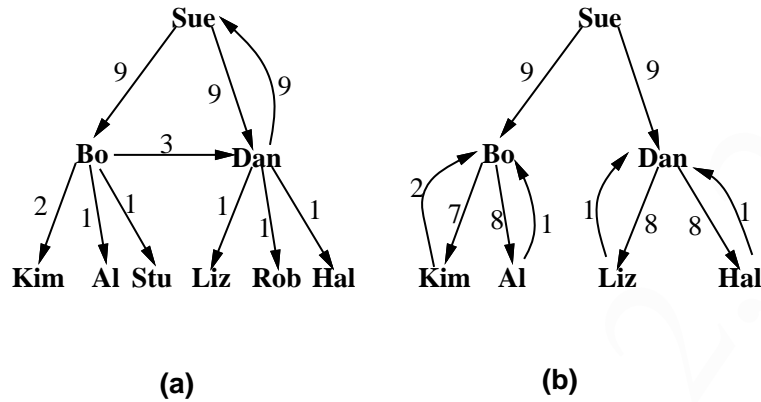


Fig. 2.4. Two weighted digraphs

- The structure of the graph.
- The importance of individual connections.

Being hierarchical is thus a question of degree. Not only specific graph models of the real world can be evaluated according to this; we can also compare different kinds of relations. Obviously, some relations, like “is the boss of”, tend to result in rather hierarchical graphs, whereas for instance “loves” is not likely to do so.

In Fig. 2.5, (a) is a plausible picture of “who is the boss of who in Dept. X” and (b) is a plausible picture of “who loves who in Dept. X”. As can be seen, (a) is almost a strict hierarchy, whereas (b) is not even connected (and thus very far from being a hierarchy). That the “who loves who in Dept. X” should form a hierarchy, like in (c), seems pretty unlikely because a relation like “loves” is inherently non-hierarchical (as opposed to for instance “is the boss of”). Consequently, even if the situation in (c) occurred, one might not feel that this is a hierarchy. Thus,

- Some kinds of relations are hierarchical of nature, and others are not.
- For the former it might be interesting to simplify the presentation of some knowledge by cutting edges to obtain hierarchies.
- For the latter, it seems that such an approach would make no sense, as it would be confusing rather than enlightening to present them as hierarchical.

Still, we have basically only made it clear that we want to deal with more situations than what falls under the rather strict definition of Bunge – in fact we want to be able to deal with almost any situation where something like a hierarchical abstraction construct occurs. In the next section we will identify some useful constructs in this respect.

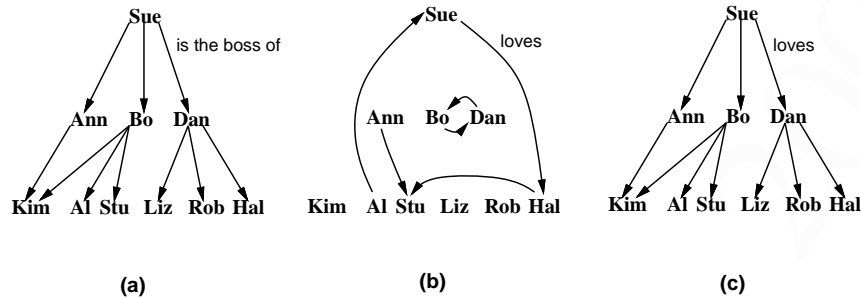


Fig. 2.5. Hierarchical and non-hierarchical relations

### 2.1.2 Four Standard Hierarchical Relations

There is a vast number of hierarchies that one might want to model, and these have rather diverse properties. Imagine for example organization hierarchies, definition hierarchies, goal hierarchies, file system hierarchies, and operating system process hierarchies.

Work in the field of semantic data modeling [174, 301, 307]) and semantic networks ([117]) has lead to the identification of four standard hierarchical relations:

- **classification**,
- **aggregation**,
- **association**, and
- **generalization**.

We define the following abbreviations:

CAGA  $\stackrel{\text{abbr}}{=}$  classification, aggregation, generalization, and association;

AGA  $\stackrel{\text{abbr}}{=}$  aggregation, generalization, and association.

The four constructs have the following definition [307]:

**Classification:** specific instances are considered as a higher level object type via the *is-instance-of* relationship (for example, “Rod Stewart” and “Mick Jagger” are specific instances of “singers”).

**Aggregation:** an object is related to the components that make it up via the *is-part-of* relationship (for example, a bicycle has wheels, a seat, a frame, handlebars etc.).

**Generalization:** similar object types are abstracted into a higher level object type via the *is-a* relationship <sup>3</sup> (for example, an employee is a person).

**Association:** several object types are considered as a higher level set object-type via the *is-a-member-of* relationship (for example, the sets “men” and “women” are members of the set “sex-groups”). Association is also likely to be encountered under the names of *membership* (e.g. [307]), *grouping* (e.g. [174]), or *collection* (e.g. [156]).

*Classification* may be regarded as orthogonal to the other three – whereas the others construct bigger things from smaller things (on the same meta-level), classification results in a shift of meta-level, in accordance with the philosophical notions of *intension* and *extension* [56, 96]. The *intension* of “man” is the property of being a man, whereas the *extension* of “man” (in any specific world, at any specific time) will be the set of all existing men (in that specific world, at that specific time). Going one meta-level higher from “man”, one can get to “species”, of whose extension “man” is a member (in this particular world, at this particular time). One does not have to go much higher until there are only very abstract notions like “words” and “concepts”, so it is of limited interest to use many meta-levels in a model.

For the other three constructs, the complicated notions of intension and extension are unnecessary, and rather straightforward set-theoretic definitions can be provided:

- Aggregation corresponds to the *Cartesian product*: If the set  $A$  is said to be an aggregation of the sets  $A_1, \dots, A_N$  this means that  $A \subseteq A_1 \times \dots \times A_N$ , i.e. each element of  $A$  consists of one element from each of  $A_1, \dots, A_N$ .
- Generalization corresponds to *union*: If the set  $A$  is a generalization of the sets  $A_1, \dots, A_N$ , this means that  $A \subseteq A_1 \cup \dots \cup A_N$ .
- Association corresponds to *membership* (i.e. embracing by set brackets): If the set  $A$  is an association of the sets  $A_1, \dots, A_N$  this means that  $A = \{A_1, \dots, A_N\}$ .

Classification should not be confused with set-theoretic membership, nor the notion of class with that of set, although there are clearly similarities in both cases. A class can be viewed as a collection of its instances. Moreover, each instance can be thought of as ‘a member of’ a class. However, a set is an extensional notion whose identity is determined by its membership. Thus, two

<sup>3</sup> Some authors use “is-a” for classification.

sets A and B are equal if they have the same members, unlike classes where equality cannot be decided by simply comparing their instances. Turning to cognitive psychology, one has identified three types of theories to explain how people develop and use categories [104]:

1. Attribute theory: Contends that one think of a list of defining attributes or features. For example, fish swim and have gills. We have in this book defined the term class according to this theory. There are some deficiencies of this approach. It is not always possible to specify defining features, and it does not take into account goodness-of-examples effects; that some instances are more typical and representative than others.
2. Prototype theory. States that when a person is presented a set of stimuli, they abstract the commonalties among the stimulus set and the abstracted representation is stored in memory. A prototype is the best representation of a category. For example, a prototypical fish might be the size of a trout, have scales and fins, swim in an ocean, lake, or river and so forth. We have a general or abstract conception of fish which somehow is typical or representative of the variety of examples with which we are familiar. When given a particular example, we compare it to the abstract prototype of the category. If it is sufficiently similar to the prototype, we then judge it to be an instance of the category.
3. Exemplar theory: Assumes that all instances are stored in memory. New instances encountered are then compared with the set of exemplar already known. This theory does not assume the abstraction of a prototype, a best example.

Parsons and Wand [298] presents some guidelines for how to decide upon classes and class structure based on the cognitive economy and inference. Cognitive economy means that , by viewing many things as instances of the same class, classification provides maximum information with the least cognitive effort. Inference means that identifying an instance as a member of a class makes it possible to draw conclusions. To decide upon potential classes two principles are discussed:

1. Abstraction form instances: A class can be defined only if there are instances in the relevant universe possessing all properties of the type that defines the class.
2. Maximal abstraction: A relevant property possessed by all instances of a class should be included in the class definition.

They propose two additional principles that apply to collections of classes: Completeness, which requires that all properties from the relevant universe be used in classification, and nonredundancy, which ensures that there are no redundant classes. A class that is a subclass of several other classes should be defined by at least one property not in any of its superclasses.

As indicated by [174], some works may use these terms somewhat differently:

- Some languages (like for instance SDM [160] and TAXIS [271]) represent aggregations by means of attributes (instead of cross product type construction). The part-of relation can be looked upon as a special case of aggregation. Based on work on object-oriented databases, this relation is further Specialized [264]. A set of component objects which form a single conceptual entity is referred to as a composite object, and the links connecting the components with this object are called part-of links. The model allows to specify for each composite link whether the reference is exclusive, i.e. the component exclusively belongs to the composite at a given point in time, or shared, meaning that the component may possibly be part-of several composites. Further, a part-of link can be defined to be either dependent, which means that the existence of the component depends on the existence of the composite, or independent, i.e. having existence irrespectively of the composite. These specializations are orthogonal, giving four possible relationships as exemplified below:
  1. A brain is part-of a person (exclusive, dependent).
  2. A paper is part-of a journal (exclusive, independent).
  3. A subprogram is part-of a program library (shared, dependent).
  4. A figure is part-of a paper (shared, independent).
- Some languages have identified several kinds of generalization. The following types of generalizations are defined by Kung [219].
 

A set of *subclasses* of a *class* **cover** the *class* if all *members* of the *class* are *members* of at least one of the *subclasses*.

A set of *subclasses* of a *class* are **disjoint** if no *members* of a *subclass* are *members* of any of the other *subclasses* of the *class*.

A set of *subclasses* which are both *disjoint* and *cover* the *class* is called a **partition** of the *class*.
- It is often useful to define association in terms of the powerset operator. As suggested both by [174] and [301], association is commonly used for constructing sets of objects of the *same type*. Consider the example of Fig. 2.6 (taken from [174]), where the \*-node denotes the association of the “person” node, meaning that the former is a subset of the powerset of the set of persons (i.e. each committee will have some group of members taken from the set of persons). Since we do not want to express at an abstract level the exact members of each committee, and since all members are persons, the association operator will have only one child in this case (whereas “men” and “women” being members of “sex-groups” earlier in this chapter signaled a use of association with several children).

### 2.1.3 Strengths and Weaknesses of Suggested Relations

We will here briefly discuss the strength and weaknesses of the suggested constructs.

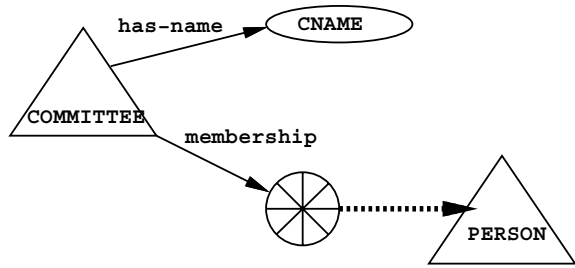


Fig. 2.6. Association with a single child

**Strengths.** As indicated by [174, 301, 307], many modeling languages provide at least some of the CAGA constructs, and the effects of introducing such constructs are positively described. [301] reports improvements in expressive economy, integrity maintenance, modeling flexibility, and modeling efficiency. But why is it that languages tend to predefine exactly CAGA and not any other hierarchical relations (like “is the boss of”)? The main reason is their *generality* and *intuitivity*.

The generality of CAGA can be accounted for by the fact that they are asubstantial. Whereas relations like “is-the-father-of” and “is-the-boss-of” contain substantives “father” and “boss”, whose semantics clearly limit the applicability of the relations, “is-part-of” uses the semantically very anonymous substantive “part”. Anything can be a “part” of something – the set of potential fathers is much more limited. The substantives “instance”, “subset”, “member” are similarly weak in semantic content. Defined in terms of sets, with no commitment as to what these sets contain, these abstraction mechanisms should be able to cover any application area. Thus, they can be useful in organization modeling, process modeling, data modeling, hardware modeling etc.

Moreover, CAGA are apparently very intuitive abstraction mechanisms, which must be why they have become so popular in the first place. We seems to find it natural to think of things as being put together from smaller parts (aggregation), as being of a specific type (classification), as being members of groups (association) which can have smaller subgroups (generalization). This might partly be because we are being trained pretty much in using such hierarchies in school, for instance learning languages (aggregation: assembling words from letters, sentences from words, etc., classification: distinguishing between word classes, identifying phrases as subject, predicate, direct object, indirect object etc., generalization: different kinds of sentences, substantives, verbs etc., association: memorizing lists of prepositions demanding a certain case in German), learning biology, learning mathematics — whatever!

**Weaknesses.** However, there are also some weaknesses to be mentioned:

- The set-oriented definition of CAGA cause some limitations on their use.

- Also, there are hierarchies which are certainly of interest in conceptual modeling which are not covered by the CAGA scheme.

As can be seen from the set-theoretic definitions given in this chapter, classification means to move up one meta-level, from an instance to a type. The other three are set-level constructs. Thus, there are two problems:

- What to do about instances?
- What to do about masses?

*1. Instances:* Instances are not necessarily such a big problem. The association construct is trivially applicable, since it can produce a set of instances just as easily as a set of sets (“Peter, Patricia, and Joey are members of the Party Committee”). Moreover, if we treat instances like sets with only one member (like Quine does in [309]), the definition of aggregation just presented is also trivially applicable (“The car # 346 was constructed from the chassis # 9213, the carrossery # 2134, and the engine #905”), and so is generalization (with the limitation that it only seems to be useful in situations where the general notion is a variable: “Joey’s murderer must have been either Peter or Patricia”, in which case “Joey’s murderer” can be said to be a generalization of “Peter” and “Patricia”).

Another question is hierarchical relations between instances (like “father-of”, “boss-of”). It is difficult to know which instance level relations people might want, and we cannot define an enormous amount of them in advance. The wisest thing for a general framework might be to provide a generic relation construct from which the users can define all the relations that they need.

*2. Mass Concepts.* As Sowa points out in [352] the set-oriented way of thinking which permeates so many information systems models work well for things that are countable, whereas there are problems for the so-called *mass nouns*, like water, love, money. Again it appears that the notions of AGA are applicable (“Chocolate is made of cocoa, sugar and milk” (aggregation), “Milk and water are both liquids” (generalization), “*Milk* and *Water* are members of the set *Liquids*” (association)). However, the problem is that we cannot use the set based definitions presented earlier in this chapter. There are two possible ways out of this:

- One can go for a more general definition of AGA which is not at all based on sets (but for instance on types).
- One can use the definitions already suggested and add some special tactics for dealing with masses.

We will not delve into this in more detail in this book.