International Handbooks on Information Systems

Series Editors
Peter Bernus, Jacek Błażewicz, Günter Schmidt, Michael Shaw
Titles in the Series

M. Shaw, R. Blanning, T. Strader and A. Whinston (Eds.)
Handbook on Electronic Commerce
ISBN 3-540-65822-X

J. Błażewicz, K. Ecker, B. Plateau and D. Trystram (Eds.)
Handbook on Parallel and Distributed Processing
ISBN 3-540-66441-6

H. H. Adelsberger, B. Collis and J. M. Pawlowski (Eds.)
Handbook on Information Technologies for Education and Training
ISBN 3-540-67803-4

C. W. Holsapple (Ed.)
Handbook on Knowledge Management 1
Knowledge Matters
ISBN 3-540-43527-1
Handbook on Knowledge Management 2
Knowledge Matters
ISBN 3-540-43527-1

J. Błażewicz, W. Kubiak, T. Morzy and M. Rusinkiewicz (Eds.)
Handbook on Data Management in Information Systems
ISBN 3-540-43893-9

S. Staab and R. Studer (Eds.)
Handbook on Ontologies
ISBN 3-540-40834-7

S. O. Kimbrough and D. J. Wu (Eds.)
Formal Modelling in Electronic Commerce
ISBN 3-540-21431-3

P. Bernus, K. Mertins and G. Schmidt (Eds.)
Handbook on Architectures of Information Systems
Foreword

This book is the first volume of a running series under the title International Handbooks on Information Systems. The series is edited by Peter Bernus, Jacek Blazewicz, Günter Schmidt and Mike Shaw. One objective is to give state of the art surveys on selected topics of information systems theory and applications. To this end, a distinguished international group of academics and practitioners are invited to provide a reference source not only for problem solvers in business, industry, and government but also for professional researchers and graduate students.

It seemed appropriate to start the series with a volume covering some basic aspects about information systems. The focus of the first volume is therefore architectures. It was decided to have a balanced number of contributions from academia and practitioners. The structure of the material follows a differentiation between modelling languages, tools and methodologies. These are collected into separate parts, allowing the reader of the handbook a better comparison of the contributions.

Information systems are a major component of the entire enterprise and the reader will notice that many contributions could just as easily have been included in another volume of the series which is on enterprise integration. Conversely, some traditionally information systems topics, as organisational analysis and strategic change management methods, will be treated in more depth in the Handbook on Enterprise Integration. The two volumes will complement each other.

The second edition of this volume is a representative survey on the most important results on Architectures of Information Systems which are presented by prominent experts. We have to thank not only the contributors for their effort but also various colleagues who helped us by suggesting relevant topics and qualified authors. The editors acknowledge the role of the advisory board members: Andy Bond, Guy Doumeingts, Keith Duddy, Mark Fox, Tom Gruber, Ted Goranson, Rudolf Haggenmüller, Linda Harvey, Matthias Jarke, Jim Melton, Chris Menzel, John Mylopoulos, Elmar J. Sinz, Riitta Smeds, François Vernadat.

One of the challenges was a technical one. We had to compile text and graphics together generated by distributed software systems from all over the world. Jörg Winckler and Anett Wagner for the second edition expertly resolved not only this problem with a number of supporters who are too many to name them all. We sincerely thank them for their help and support.

P. Bernus, K. Mertins, G. Schmidt
Contents

Architectures of Information Systems ......................................................... 1
   Peter Bernus, Günter Schmidt

Part One: Techniques and Languages for the Description of
Information Systems ................................................................................ 11
   Peter Bernus, Günter Schmidt

Properties of Information Modeling Techniques for Information
Systems Engineering.................................................................................. 17
   John Mylopoulos, Alex Borgida

EXPRESS .................................................................................................. 59
   Reiner Anderl, Harald John, Christian Pütter

Object-Role Modeling (ORM/NIAM)....................................................... 81
   Terry Halpin

Database Language SQL ......................................................................... 105
   Jim Melton

Petri Nets ................................................................................................. 133
   Jean-Marie Proth

State Transition Diagrams ....................................................................... 153
   Jules Desharnais, Marc Frappier, Ali Mili

The Process Interchange Format.............................................................. 173
   Jintae Lee, Michael Gruninger, Yan Jin, Thomas Malone,
   Austin Tate, Gregg Yost

Process Language GPN ........................................................................... 197
   Günter Schmidt, Oliver Braun

The IDEF Family of Languages .................................................................. 215
   Christopher Menzel, Richard J. Mayer

The CIMOSA Languages ........................................................................ 251
   François Vernadat
ConceptBase: Managing Conceptual Models about Information Systems
Manfred A. Jeusfeld, Matthias Jarke, Hans W. Nissen
Martin Staudt ................................................................. 273

Conceptual Graphs ................................................................. 295
John F. Sowa

GRAI GridDecisional Modelling ................................................. 321
Guy Doumeingts, Bruno Vallespir, David Chen

Modeling of Business Systems Using SOM .................................. 347
Otto K. Ferstl, Elmar J. Sinz

Workflow and Service Composition Languages ............................ 369
Mathias Weske, Gottfried Vossen, Frank Puhlmann

XML - The Extensible Markup Language and its Use in the Field of EDI
Erik Wüstner, Peter Buxmann, Oliver Braun .................................. 391

Modeling Information-Systems with UML
Unified Modeling Language ................................................... 411
Fritz Letters

Part Two: Software Engineering Methods for Information System
Construction ........................................................................... 457
Peter Bernus

Information Engineering Methodology ...................................... 459
Clive Finkelstein

Object-Oriented Software Engineering Methods ........................... 485
Brian Henderson-Sellers

Euromethod Contract Management ............................................ 521
Alfred Helmerich

Part Three: Tools for Analysis and Design ................................. 535
Peter Bernus

An Integrated Enterprise Modeling Environment .......................... 539
Florence Tissot, Wes Crump
WorkParty - Business Processes and Workflow Management..............569
   Walter Rupietta

Business Process Reengineering with PROPLAN®..........................591
   Günther Schuh, Thomas Siepmann, Volker Levering

ARIS – Architecture of Integrated Information Systems.....................605
   August-Wilhelm Scheer, Kristof Schneider

Tools for Analysis and Simulation: BONAPART..............................625
   Herrmann Krallmann, K. Wiener, Gay Wood

MO²GO: User Oriented Enterprise Models for Organisational and IT
   Solutions ....................................................................................649
   Kai Mertins, Frank-Walter Jaekel

Part Four: Reference Models..........................................................665
   Kai Mertins, Peter Bernus

IBM Insurance Application Architecture (IAA)...............................669
   Jürgen Huschens, Marilies Rumpold-Preining

Fraunhofer Simulation Reference Models......................................693
   Markus Rabe, Kai Mertins

Configuring Business Application Systems....................................705
   Stefan Meinhardt, Karl Popp

The SIZ Banking Data Model......................................................723
   Hans-Bernd Kittlaus, Daniela Krahl

ODP and OMA Reference Models................................................745
   Andy Bond, Keith Duddy, Kerry Raymond

Part Five: Selected Topics in Integrating Infrastructures.............765
   Peter Bernus

Architectural Requirements of Commercial Products.....................767
   Ted Goranson

Integration Infrastructures for Agile Manufacturing Systems..........789
   Richard H. Weston, Ian A. Coutts and Paul E. Clements
Distributed Processing: DCE, CORBA, and Java ...................................823  
  Andy Bond, Keith Duddy, Kerry Raymond

Higher Level Integration by Multi-Agent Architectures .....................855  
  Mihai Barbuceanu, Rune Teigen

List of Contributors ...........................................................................887

Index ....................................................................................................893
This chapter is an introduction into the scope of the Handbook on Architectures of Information Systems. We will point out that this volume gives a comprehensive survey of the most important aspects in this area giving not only a list of available alternatives but providing also a guidance amidst the many proposals.

1 What is an Information System?

During the past three decades the concept of information system and the discipline of information systems underwent an evolution, as witnessed by definitions given by various authors.

Mader and Hagin in 1974 [MH74] defined the information system as the system which provided “... transaction processing and decision support ...”. Brookes et al [Broo82] defined it as “... all forms of information collection, storage, retrieval, processing and communication ...” as “... the organization’s instrumentation ... informing decision makers of the state of the organization ... including computer based and human implemented systems”. Inmon [Inm86] defines “... information systems architecture: [as] the modelling of the data and processes of a company and how that model relates to the business of the company ...”. Tatnal et al [Tat95] define an information system as “... [a system] comprising hardware, software, people, procedures, and data, integrated with the objective of collecting, storing, processing, transmitting and displaying information” and elaborate further by defining “functional information systems” which support specific business functions, e.g. accounting, human resource management, manufacturing, marketing, etc. and “integrated information systems” which provide information flow across all areas of application. Sandstrom proposes that the information system “... is a designed tool, the purpose of which is to serve people in active work with information and in an organization. It is an organized construction with subsystems for collecting, processing, storing, retrieving, and distributing information together, influenced by people. It becomes an abstraction of a service function when studied”. In [Stoh92] it is proposed that “... the field
is known now as Information Systems. ‘Systems’ is the operative word, since the field includes not only technologies, but people, processes, and organizational mechanisms as well ...”. All of these definitions contribute to our understanding of information systems.

The main requirement that an information system must satisfy is to provide and maintain an integrated information flow throughout the enterprise, so that the right information is available whenever and wherever needed, in the quality and quantity needed. This generic requirement defined different tasks for information systems practitioners in the past. The first focus of information systems research and development emerged from the need of physically enabling the information flow, a level of integration that we call today physical integration. As physical integration became reality through the installation of networks and adoption of standards it became possible to concentrate efforts on the interoperability of applications, i.e. to enable the various business applications to be combined and interconnected for new tasks, without having to re-design them. Interoperability is not yet achieved in many business areas, but practice of the 1990s brought success in some of them, such as database interoperability. The next challenge after application integration is business integration, which is the question how various business functions can be interconnected and efficiently combined through information systems.

An information system is a system for collecting, processing, storing, retrieving, and distributing information within the enterprise and between the enterprise and its environment. The information system is a functionally defined subsystem of the enterprise, i.e. it is defined through the services it renders. It may be implemented by the enterprise’s own resources (automated equipment and humans), but parts of the information system’s services may be provided to the enterprise by other enterprises.

2 What is an Information System Architecture?

An architecture is the integrated structural design of a system, its elements and their relationships depending on given system requirements. The notion of an architecture is widely used in the context of buildings and computers. When applied to information systems we follow the definition of Wall [Wal96] and assume that an architecture is the abstract plan including the corresponding designing process of the system’s structure appropriate to the goals of the system based on design principles and a methodological framework.
Below, we treat the required components of information system architecture according to the Generalized Enterprise Reference Architecture and Methodology (GERAM) [TF97], defining the information system within the context of the enterprise (see Fig. 1 for an overview on GERAM). GERA, the Generalized Enterprise Reference Architecture is one component and defines several important ingredients of architectures for any enterprise entity, including the information system.

Fig. 1. GERAM framework components.

Entities involved in the information system’s architecture are the enterprise and its products. Both must be considered for the purposes of information systems design, implementation, and operation, especially when
more and more systems are designed for virtual enterprises. Thus the information system must support the information flow:

- which integrates the value chain, i.e. the business process involved in producing the product(s) and service(s) of the enterprise.
- which integrates the development of the enterprise throughout its entire life.

Both entities, i.e. the enterprise and the product have a life history, which is the history in time of all relevant events, transformations and milestones that happened or are planned to happen to the entity. Life histories are unique and particular, therefore a functional abstraction is used to describe the common functional elements of life histories, called life-cycle. The life-cycle model is defined to contain “phases”, which are regarded as types of transformation rather than as temporal sequences. E.g. GERA defines the life-cycle phases: identification - concept - requirements - design - detailed design - implementation - operation and decommissioning. For more details about the relationships among life-cycles of enterprise entities see [TF97]

In the early phases the enterprise and its strategies, objectives, mission, vision, values, policies etc. are defined, and at this stage the separation of the information system from the rest of the enterprise is not always possible. Rather, this separation is only one of the possible outcomes of the identification of involved enterprise entities, it happens if the enterprise decides to outsource information system services to an external provider. Consequently (i) methodologies developed for strategic information systems management and strategic management are very similar - both essentially managing change, and (ii) information system considerations are important but not exclusive ingredients in that process. However, if it is demonstrated early in a change process that it is the information system of the enterprise that needs change (which is often the case), then specialized information systems planning methodologies may be utilized. In the ensuing enterprise life-cycle phases the information system becomes more and more a separate component, thus information systems specific design and implementation methods and tools can be made available.

3 Modelling Framework and Views

An architecture has to represent all relevant aspects of a system. These aspects are defined by models representing different system views. They are derived from the goals the system has to fulfil and the constraints defined
by the system’s environment. The GERA modelling framework describes what models of the enterprise may need to be created and maintained during the enterprise’s life history. The following views on information systems are considered essential to be represented by the models of an architecture.

1. **Information, Functions, Co-ordination and Synchronisation.** The major elements of information systems are the data, the functions using or producing the data, and relationships describing how functions relate to data and other functions. The modelling framework therefore needs to represent
   - the structure of data.
   - the structure and behaviour of functions, and
   - the rules for co-ordination and synchronisation (defining the dynamic properties of a system).
Depending on the actual selection of a modelling language these three views may or may not be separate.

2. **Organization.** Information systems are invariably integrated into organizations. Thus an organizational view needs to describe the relation between the users and the system. It shows how the information system is used by an organization in terms of collecting, processing, storing, retrieving, and distributing information. There are two important issues which have to be covered: (i) the structure of the organization where the information system is used has to be represented, i.e. which department, group, and individual takes over the responsibility for correct usage of the system, and (ii) how the how of information is organized to meet the requirements of the organization.

3. **Resources.** Resources are used to physically implement and to run the information system. The most important information processing resources are software, hardware, and humans to carry out innovative or otherwise not automated information processing tasks.

Each of these views are represented by models belonging to a life-cycle phase, such as described in Fig. 2. Accordingly

- the models of the “management and control” of the enterprise describe the service of the information system traditionally rendered by a management information system.
- the models of the enterprise’s “service to the customer” describe the information exchange requirements among the business processes, supporting business transactions with product related information.

The purpose of an information system is derived from the mission of the enterprise which it needs to serve. Requirements level models of the sys-
tem describe its functionality (necessary tasks) while design level models propose a solution to how these tasks can be performed. Design level models are more detailed and concrete in the phases of detailed design and implementation.

![GERA modelling framework](image)

**Fig. 2.** GERA modelling framework.

The first part of the handbook describes a representative selection of modelling languages supporting the analysis and the design of information systems, while the third part presents tools which are suitable for model representation and analysis at each of these levels. It is to be noted, that the model categories of GERA are not only meant for information systems representation, but for the modelling of the entire enterprise, and the handbook describes only those languages which are most important from the point of view of the integrated information flow in the enterprise, i.e. information system models. For this reason there is no chapter in this handbook about “languages to describe functional models of technological equipment”, “languages to model factory layouts (detailed design level resource models)” or “financial models of the resources”. Not that these models would be less important, but because they are beyond the scope of this volume. Even organizational and resource modelling languages are treated less prominently, for exactly the same reason.
3.1 Models and Methodologies

Using the modelling framework and associated tools information systems models are built. An architecture has to guarantee that the mission of the enterprise is taken into account in the process of design, and that the system will support the enterprise in achieving its objectives. The models of the information system should provide sufficient evidence for the designer to believe that this will indeed be the case. From the models the system properties should be derivable and conversely, the models have to be designed so that the system requirements can be fulfilled. The second part describes methodologies for information system construction which are intended to ensure that the system is consistent and supportive of the enterprise mission. We also plan to amalgamate enterprise engineering and information systems engineering methodologies in a forthcoming volume, the Handbook on Enterprise Integration to broaden the scope of methodologically supported change.

Information systems design methodologies should safeguard that basic modelling requirements are met. Among these are the following:

- correctness, integrity, consistency, completeness.
- low level of complexity through modularity.
- clarity and ease of communication.
- Adequacy, as a basis for system development.
- provision of a guideline for research.

It would be impossible to design good quality models without relying on suitable reference models. Typical models, or reference models of the information system are presented in the fourth part. Such models are also often called “Type - Reference Architectures” [BNW96]. Information system architectures are defined for the long term and thus have to cope with continuous change, they must be stable, open, flexible, extendible and should be supported by standards. These properties also ease the re-use of different models, methods and techniques within the same architectural framework.

Reference models may be provided for certain classes of enterprises on certain levels. This property is referred to as granularity in [Sch96]. Thus there exist generic reference models of good practice which are general enough to cover a broad spectrum of applications, while a more specific model may be related to a certain class of enterprise, so that all companies belonging to this class might use the enterprise model as a guideline for more detailed model building. The most specific model refers to a particular enterprise and its information system integrating its business functions.
3.2 Building Blocks of Information Systems

Significant resources of the implementation of the information system are: humans (individuals, groups, and higher level organizational units), and computer software and hardware systems. From these this handbook treats in its last part some basic modules, or product types which are likely to play very significant roles in the building of any information system. The treatment includes a strategic analysis of the direction of information technology in the enterprise, as well as an overview of the latest distributed system technologies, and the requirements and examples for an information integration infrastructure. Readers familiar with information systems literature will be missing from this handbook a chapter on organizational analysis, agent modeling, or on information system evaluation methods. After all the human organization plays a significant part in the information system, both as user and as producer of information. Hirschheim et al [HS88] state that “Organizations are complex social and political entities which defy purely objective analysis. As information systems form part of organizational reality (i.e. the gestalt, they cannot be viewed in isolation”. We therefore plan to treat the social organizational domain of information systems, combining analysis and design, using interpretive approaches in the larger context of enterprise engineering, including it in a forthcoming volume, the Handbook on Enterprise Integration.

References

Techniques and Languages for the Description of Information Systems

This part is about those techniques and modelling languages which are typically used to specify and design information systems. A modelling language is a set of constructs for building models of systems, such as an information system. Models can be prepared of a system at various stages of the system life-cycle (e.g. specification, design, implementation), and from various viewpoints (e.g. information, function, resources). Depending on the goal of modelling the selected modelling language should be adequate or competent for the purpose of the modelling task. From the point of view of the user of the language it must be understandable, easy to use, and models developed using the language must be presentable and easy to interpret for the intended audience. From the point of view of the use of the language it must have sufficient expressive power to be able to capture all the information that the required type of model needs to contain. E.g., if the model of the system must be used for calculating the minimum time necessary to perform a process, then a pure functional modelling language which has no notion of time is inadequate [Sch97].

Modelling languages can be described by their syntax and semantics. The syntax of a language defines what are the legal constructs of that language. The most often used form of syntax definition is the Bacus-Naur Form (BNF). The definition of the language’s syntax defines all legal constructs of the language, including terminal symbols which have no further structure and expressions, i.e. structures which can be built out of these symbols. The syntax definition of a language is useful for being able to build a parser that will examine an arbitrary expression and accept or reject it as a legal expression of the language. Furthermore, if the expression is legal, then the parser is able to analyse the structure of the expression and present it in the form of a parse tree or of parse trees determining how the expression is built using structure definitions given in the BNF.

The semantics of a language defines the meaning of the expressions written in that language. There are several ways to define the meaning of a language. Denotational semantics is used to define how expressions formed in the given language can be mapped to an interpretation or model which may be a real world or a symbolic system. If the language is mapped to an equivalent representation in a suitably selected logic (mostly first order logic) then the model theory of that logic will be suitable for the
definition of the semantics. It is also customary to define a proof theory that allows reasoning about the constructs of the language, in particular proving properties of expressions. The meaning of expressions in the language will then be determined by what possible models are described by those expressions. E.g., the meaning of an Entity Relationship Schema is what is common in all possible implementations of that schema. For further details on denotational semantics refer to [Sch86].

For languages that describe operations the definition of the semantics can be using operational semantics. This can be done, for example, by defining an abstract machine and describing the effect of operations on the state of that abstract machine. Depending on the reason why the operational semantics is developed operations may be described by their preconditions and post-conditions i.e. statements that must be true to be able to execute the operation, and statements that will be true after the execution as well as invariants i.e. properties that are not effected by the execution of the operation. Some languages developed for the purpose of specifying the meaning of languages especially programming languages are the Vienna Definition Language [Weg72] and Z [Spi88].

The formal specification of the operational semantics for a language can be used for the unambiguous definition allowing compatible and certifiable implementations of interpreters for the language. However, for any language of appreciable size this is a complex matter and due to the nature of these definitions other more simple definitions of the semantics are also necessary for end users. Users will still wish to verify the models developed in the language, but the verification will use several independent means, such as (i) execution of the models using test examples, (ii) in certain cases formal proofs, (iii) informal means, such as discussions. Even if the formal specification of the language’s semantics was used only by the implementors or interpreters and not used by the end users of the language, it will be ensured that the evaluation of these models across different implementations will produce identical results.

Informal specification of the language semantics is usually given by formal presentation of the language syntax accompanied by natural language description of the intended meanings of the constructs (both in case of denotational and operational semantics). This is the approach that authors of this part have taken. The focus is on the question which languages are available to support information modelling and systems description. There is not one language which is equally suited for all purposes; each language has its individual strength to meet specific modelling requirements. Some languages might be applicable for a broad range of applications while others are more specialised and purpose oriented.
A model of an information system must represent all relevant views on the system. These are related to the system’s elements and their relationship, i.e. the data and the objects of the application domain, the processes and activities to be carried out, the organizational environment and the communication needs. This part contains three groups of techniques and languages according to their purpose or intended use:

- **Data and object modelling languages** - intended for the modelling of the information view, i.e. the information that is stored or processed by the information system at various phases of the system life-cycle,

- **Activity and process modelling languages** - intended for the specification, design, and implementation modelling of the function of the information system,

- **Multi view languages** - those languages which are suitable for the representation of multiple views of the information system, possibly serving the modelling needs of multiple levels of the system life-cycle.

The choice of languages for information system modelling is so great that to select a few that will get prominent exposition in this book was extremely hard. We intended to provide examples of languages which can cover the life-cycle phases of the information system, from initial specification to implementation and operation.

Some languages are defined together with a modelling method or technique. A modelling method gives guidance for the user regarding how models are best built using the language. For example a modelling method would give specific instructions for information gathering, model building, model quality control etc. Information systems design methodologies would in turn incorporate such modelling methods or techniques as components of the methodology.

This part starts with a contribution by John Mylopoulos. It gives a state of the art survey on information modelling techniques for knowledge representation, data modeling, and requirements analysis. It also offers a comparative framework for information modelling approaches classifying them according to ontologies, abstraction mechanisms, and available tools.

The next three contributions are related to the group of data and object modelling languages. Reiner Anderl, Harald John and Christian Pütter give a description of Express which is a formal modelling language for the specification of static aspects of information representation. Terry Halpin presents Object-Role Modelling also known as ORM or NIAM. This is a language designed for modelling and querying an information system at the conceptual level. Jim Melton surveys the main features of the database language SQL, in fact SQL2. We did not include SQL3 in this handbook.
because we felt that the SQL3 standard was still in developing stage and therefore its description would better wait until a next edition.

Although the editors were keen to include contributions on the Entity Relationship (ER) data model, and on the Object Database Management Group’s (ODMG) data model, the contributions did not make this edition. The extended ER data model is prevalently used as a requirements and design level data model and is usually followed by a mapping to the relational data model on the detailed design and implementation levels. The ODMG data model serves as the common data model of many object-oriented database management systems. For details of the ER data model the reader is referred to [Elm94, Bat92] and for ODMG to [ODMG97].

The next four contributions belong to the group of activity and process modelling languages. Jean-Marie Proth gives an introduction into the theoretical background of Petri Nets. They are widely used for the evaluation and simulation of discrete event systems. State Transition Diagrams have the same focus and are discussed by Jules Desharnais, Marc Frappier, and Ali Mili. This language has a long tradition being expanded in the recent past to include features to represent hierarchy, timing, and communication. Jintae Lee, Michael Gruninger, Yan Jin, Thomas Malone, Austin Tate and Gregg Yost present PIF, the Process Interchange Format. It is designed to help automatically exchange process descriptions among different process tools using a single interface. Recent developments regarding PIF should be mentioned, especially the likely merger of PIF with the Process Specification Language (PSL) effort currently underway at the US National Institute of Standards (NIST). PSL has the ambitious objective to describe manufacturing processes such that the semantics of the language is axiomatised in form of ontological theories. Günter Schmidt gives a survey on GPN, a language especially developed for planning and scheduling of processes. GPN is mainly used for the optimisation of business processes in terms of time and cost. It directly relates to the framework of scheduling theory. It is possible to build models which match to the application of optimisation algorithms.

The third group is related to multi view languages and contains seven contributions. Christopher Menzel and Richard J. Mayer describe the IDEF family of languages. They cover the syntax and the semantic rules of the three most widely used IDEF0, IDEF1X, and IDEF3. Note that the languages are used in conjunction with a modelling method, thus the authors refer to IDEF0, 1X and 3 as modelling methods, not languages only. The CIMOSA languages are presented by Francois Vernadat. These languages are based on an event driven process model and cover functional, information, resource and organisational aspects of an enterprise and are defined for all life-cycle phases. It is expected that this wide scope approach to
modeling would eventually get harmonised with many of the languages less wide in their coverage, or based on the formal definition of the semantics of these modeling languages more and more semantic translators would become available. Manfred A. Jeusfeld, Matthias Jarke, Hans W. Nissen, and Martin Staudt write on ConceptBase. This is a meta data management system intended to support the cooperative development and evolution of information systems with multiple interacting formalisms. ConceptBase, which is the implementation of a version of the Telos specification language, allows its user to extend the basic modelling formalism, because of the ability of the language to specify meta-schemas on arbitrary levels (meta, meta-meta, etc.). In spite of the seemingly higher order nature of the language it has a first order semantics, which is important for efficiency reasons. The next contribution is on Conceptual Graphs (CGs) given by John F. Sowa. These graphs show the logic designed for the visualization of knowledge represented in computer systems. Conceptual graphs can be thought of as a graphical notation for First Order Logic, which determines the expressive power of CGs. In fact CGs have been proposed as graphical representation of KIF (Knowledge Interchange Format) [GF92]. One important application is the possibility to use KIF for the formal specification of the meaning of different modelling languages through the expression of their semantics in form of ontological theories. Guy Doumeingts, Bruno Vallespir, and David Chen describe a language called the GRAI Grid which has been developed for the modelling of the management system of enterprises. As the paper shows management is best described in terms of decisions, thus the name decisional modeling. The uniqueness of this language lies in the fact that it has been developed on the basis of an ontology which has proven correct in systems theory and control system theory. This ontological underpinning, though not fully formalized, gives the language an advantage over other languages in which the user needs to develop a theory of what the best representation of management may be. The approach defines decision centres and their relationships defined by information links and decision frameworks. The Semantic Object Model (SOM) is described by Otto K. Ferstl and Elmar J. Sinz. SOM supports modelling of business systems on multiple levels of the life-cycle, such as planning, analysis, and design. The last contribution of this part is given by Mathias Weske and Gottfried Vossen discussing workflow languages. They survey the requirements, concepts, and usage patterns of such languages which are used in commercial workflow management systems.

Peter Bernus, Günter Schmidt
References

[Sch86] Schmidt, D., Denotational Semantics, Allyn and Bacon, 1986
Properties of Information Modeling Techniques for Information Systems Engineering

John Mylopoulos, Alex Borgida

Information modeling is concerned with the construction of symbolic structures which capture the meaning of information about some domain of discourse, and organize it in ways that make it understandable and useful to people, and serves as a core technology for information systems engineering.

We present a brief history of information modeling in Computer Science, surveying techniques developed within Knowledge Representation (Artificial Intelligence), Data Modeling (Databases), and Requirements Analysis (Software Engineering and Information Systems).

The presentation then offers a comparative framework for information modeling proposals which classifies them according to their ontologies, i.e., the kinds of fundamental notions they make available, the set of abstraction mechanisms (or, structuring principles) they support, as well as the tools they provide for building, analyzing, and managing application models. For example, the ontology of static world modeling includes notions such as “entity/concept” and “relationship/association”. In turn, generalization, aggregation, and classification are well-known abstraction mechanisms, adopted by many information models and used widely in information modeling practice.

The final component of the paper uses the comparative framework proposed earlier to assess well known information modeling techniques, both from a user and a designer perspective.

1 Introduction

Information modeling constitutes a cornerstone of information systems engineering and management. To build, operate and maintain an information system, one needs to capture and represent the meaning and inherent structure of a variety of rich and multi-faceted information, including the system’s subject matter, its internal structure, its operational environment and its development history. Once captured, the information can be used for communication between people -- say, the information system owners, users and developers -- but also for building tools which facilitate their management throughout their lifetime.
The DAIDA project [Jarke92], whose aim was the development of an environment for building information systems, characterized this information in terms of four “worlds”, illustrated in Fig. 1. The subject world consists of the subject domain for the information system, i.e., the world about which information is maintained by the system for the benefit of its users. For instance, the subject world for a banking system consists of customers, accounts, transactions, balances, interests rates and the like. The system world, on the other hand, describes the information system itself at several layers of implementation detail. These layers may range from a specification of the functional requirements for the system, to a conceptual design and an implementation. The usage world describes the (organizational) environment within which the system is intended to function, and consists of agents, activities, tasks, projects, users, user interfaces (with the system) and the like. Finally, the development world describes the process that created the information system, the team of systems analysts and programmers involved, their adopted methodology and schedule, their design decisions and rationale. All of this information is relevant both during the initial development of the system and, later, during operation and maintenance. Consequently, all of this information needs to be represented, somehow, in any attempt to offer a comprehensive framework for information systems engineering. This is precisely the task of information modeling.

![Diagram showing the four worlds of information systems engineering](image)

Fig. 1. The Four Worlds of Information Systems Engineering.

Information modeling has been practiced within Computer Science since the first data processing applications in the ‘50s, when record and file structures were used to model and organize information. Since then, there
have been literally thousands of proposals for information models, covering many different areas of Computer Science and Information Systems Engineering.

The purpose of this chapter is to propose a comparative framework for information modeling techniques and practice, and also to hint at some directions for further research. Section 2 of the paper introduces basic definitions, while section 3 presents a brief (and admittedly biased) history of the field. Section 4 offers a comparative framework for information models in terms of the modeling ontology and abstraction mechanisms they support, also the tools they offer for modeling, analysis and management. Section 5 assesses particular information modeling techniques, while section 6 summarizes the basic thesis of the paper and suggests directions for further research.

2 Preliminaries

Information modeling involves the construction of computer-based symbol structures which model some part of the real world. We will refer to such symbol structures as information bases (generalizing the term from others terms in Computer Science, such as database and knowledge base). Moreover, we shall refer to the part of the (real) world being modeled by an information base as its application. Fig. 2 illustrates the fundamental nature of information modeling. Here, the information base is modeling some real-world situation involving several individuals. The atoms out of which one constructs the information base are assumed to denote particular individuals in the application, while the associations within the information base denote real-world relationships, such as physical proximity, social interaction, etc. The information base is queried and updated through special-purpose languages, analogously to the way databases are accessed and updated through query and data manipulation languages.

It should be noted that an information base may be developed over a long time period, accumulating details about the application, or changing to remain a faithful model of a changing application. In this regard, it should be thought of as a repository that contains accumulated, disseminated, structured information, much like human long-term memory, or databases, knowledge bases, etc., rather than a mere collection of statements expressed in some language. Consequently, the organization of an information base should reflect its contents and its use, not its history. This implies that an information base can’t be simply a collection of statements about the application, added to the information base over time. Rather,
these statements have to be organized according to their subject matter and interrelated according to their content.

![Diagram](image)

**Fig. 2.** Modeling an application with an information base.

As indicated earlier, an information base used during the development of an information system will contain models of one or more of the four worlds of Fig. 1. Some of these models may be used during the definition of databases and applications programs which are part of the information system under development. Others may be used for operation and maintenance purposes, e.g., explaining to users how to use the system, or explaining to maintenance personnel how the system works.

What kinds of symbol structures does one use to build up an information base? Analogously to databases, these symbol structures need to adhere to the rules of some information model. The concept of an information model is a direct adaptation of the concept of a data model. So is the following definition.

An *information model*\(^1\) consists of a collection of symbol structure types, whose instances are used to describe an application, a collection of operations which can be applied to any valid symbol structure, and a collection of general integrity rules which define the set of consistent symbol structure states, or changes of states. The *relational model* for databases [Codd70] is an excellent example of an information model. Its basic symbol structure types include *table*, *tuple*, and *domain*. Its associated operations include *add*, *remove*, *update* operations for tuples, and/or *union*, *intersection*, *join*, etc. operations for tables. The original relational model supported a single integrity rule: No two tuples within a table can have the same key.

---

\(^1\) Adopted from Ted Codd’s classic account of data models and databases [Codd82].
Given this definition, one can define more precisely an *information base* as a symbol structure which is based on an information model and describes a particular application.

Is an information model the same thing as a *language*, or a *notation*? For our purposes, it is not. The information model offers symbol structures for representing information. This information may be communicated to different users of an information base (human or otherwise) through one or more languages. For example, there are several different languages associated with the relational model, of which SQL is the most widely used. In a similar spirit, we see notations as (usually graphical) partial descriptions of the contents of an information base. Again, there may be several notations associated with the same information model. e.g., the graphical notations used for data flow diagrams.

The information models proposed and used over the years have been classified into three different categories. These, roughly speaking, reflect a historical advance of the state-of-the-art on information modeling away from machine-oriented representations and towards human-oriented models which are more expressive and can cope with more complex application modeling tasks.

**Physical information models.** Such models employed conventional data structures and other programming constructs to model an application in terms of records, strings, arrays, lists, variable names, B-trees, and the like. The main drawback of such models is that they force on the programmer/modeler two sets of conflicting concerns, one related to computational efficiency, and the other to the quality of the application model. For example, if one chooses to model persons in the application in terms of 8-character strings and structure an information base in terms a B-tree, these choices are driven by efficiency considerations, and have nothing to do with the application being modeled.

**Logical information models.** The early ‘70s saw several proposals for *logical data models* which offered abstract mathematical symbol structures (e.g., sets, arrays, relations) for modeling purposes, hiding the implementation details from the user. The relational and network models for databases are good examples of logical models. Such models free the modeler from implementation concerns, so that she can focus on modeling ones. For instance, once the modeler has chosen the relational model, she can go ahead and use tables to build an information base, without any regard to how these tables are physically implemented. Unfortunately, logical symbol structures are flat, and have an “impedence mismatch” with the conceptual structure of most applications, which is object-centered.
**Conceptual information models.** Soon after logical information models were proposed, and even before relational technology conquered the database industry, there were new proposals for information models which offered more expressive facilities for modeling applications and structuring information bases. These models (hereafter, *conceptual models*) offer terms for modeling an application, such as *Entity*, *Activity*, *Agent* and *Goal*, which are related to the *semantics* of the application. Moreover, they offer means for organizing information in terms of *abstraction mechanisms* which are often inspired by Cognitive Science [Collins88], such as generalization, aggregation and classification. Such models are supposed to model an application more directly and naturally [Hammer81]. In the sequel, we focus the discussion on conceptual models, since they have constituted the state-of-the-art in the field for over two decades.

### 3 Brief History

Over the years, there have been hundreds of proposals for conceptual models, most defined and used once, within a single research project. We note in this section some of the earliest models that launched fruitful lines of research and influenced the state-of-practice. Interestingly enough, these models were launched independently of each other and in different research areas within Computer Science.

Ross Quillian [Quillian68] proposed in his PhD thesis *semantic networks* as directed, labeled graphs that are convenient for modeling the structure of human memory (1966) Nodes of his semantic network represented concepts (more precisely, word senses.) For words with multiple meanings, such as “plant”, there would be several nodes, one for each sense of the word, e.g., “plant$_1$” as in “industrial plant”, “plant$_2$” as in “evergreen plant”, “plant$_3$” as in “I plant my garden every year”, etc. Nodes were related through links representing semantic relationships, such as $\text{isa}$ (“A bird is a(n) animal”), $\text{has}$ (“A bird has feathers”), and $\text{eats}$ (“Sharks eat humans”). Moreover, each concept could have associated attributes, representing properties, such as “Penguins can’t fly”.

There are several novel ideas in Quillian’s proposal. First, his information base was organized in terms of *concepts* and *associations*. Moreover, generic concepts were organized into an $\text{isa}$ (or, generalization) hierarchy, supported by attribute inheritance. In addition, his proposal came with a radical computational model termed *spreading activation*. Thus, computation in the information base was carried out by “activating” two concepts