Process Control Engineering

Edited by M. Polke

with the collaboration of
U. Epple and M. Heim

and contributions by
W. Ahrens, D. Balzer, M. Björkmann, H. Drahten, U. Epple,
M. Freytag, E. D. Gilles, K. Hartmann, M. Heim, R. Hotop,
N. Ingendahl, N. Kuschnerus, R. Metz, W. Mutz, E. Nicklaus,
W. Noerpel, U. Pallaske, G. Pinkowski, M. Polke, J. Raisch,
G. Schmidt, K. H. Schmitt, H.-J. Schneider, G. U. Spohr,
H. Steusloff, R. Vogelgesang, H. Wehlan

VCH Weinheim · New York · Basel · Cambridge · Tokyo

Process Control Engineering

Edited by M. Polke

VCH

Distribution:
VCH, P.O. Box 10 11 61, D-69451 Weinheim (Federal Republic of Germany)
Switzerland: VCH, P.O. Box, CH-4020 Basel (Switzerland)
United Kingdom and Ireland: VCH (UK) Ltd., 8 Wellington Court, Cambridge CB1 1HZ (United Kingdom)
USA and Canada: VCH, 220 East 23rd Street, New York, NY 10010-4606 (USA)
Japan: VCH, Eikow Building, 10-9 Hongo 1-chome, Bunkyo-ku, Tokyo 113 (Japan)

ISBN 3-527-28689-6

Process Control Engineering

Edited by M. Polke

with the collaboration of
U. Epple and M. Heim

and contributions by
W. Ahrens, D. Balzer, M. Björkmann, H. Drahten, U. Epple, M. Freytag, E. D. Gilles, K. Hartmann, M. Heim, R. Hotop, N. Ingendahl, N. Kuschnerus, R. Metz, W. Mutz, E. Nicklaus, W. Noerpel, U. Pallaske, G. Pinkowski, M. Polke, J. Raisch, G. Schmidt, K. H. Schmitt, H.-J. Schneider, G. U. Spohr, H. Steusloff, R. Vogelgesang, H. Wehlan

VCH Weinheim · New York · Basel · Cambridge · Tokyo

Prof. Dr. Martin Polke
Rheinisch-Westfälische
Technische Hochschule Aachen
Lehrstuhl für Prozeßleittechnik
Turmstr. 46
52064 Aachen
Federal Republic of Germany

Published jointly by
VCH Verlagsgesellschaft, Weinheim (Federal Republic of Germany)
VCH Publishers, New York, NY (USA)

Editorial Directors: Stephen Hawkins, Philomena Ryan-Bugler and Karin Sora
Production Manager: Peter Biel
Cover Design: Phase Model of Production, Process Control System, and Operational Control of Process Plants.

Library of Congress Card No.: 94-061095

British Library Cataloguing-in-Publication Data:
A catalogue record for this book is available
from the British Library

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Process control engineering/ ed. by M. Polke. With the collab
of U. Epple and M. Heim and contributions by W. Ahrens ... –
Weinheim; New York; Basel; Cambridge; Tokyo: VCH, 1994
ISBN 3-527-28689-6
NE: Polke, Martin [Hrsg.]; Epple, Ulrich

Printed on acid-free and low chlorine paper

Cover illustration: Jürgen Wirth, D-63303 Dreieich
Composition, printing and bookbinding: Graphischer Betrieb Konrad Triltsch, D-97070 Würzburg

Dedicated to my father Franz Polke,
my teacher Prof. Dr. Fritz Stöckmann,
my mentor Dr. Otto Koch and
the initiator of process control engineering Dr. Axel Lippert.

Preface

This book surveys the methods, tasks and tools of process control engineering. Its scope has been purposely made broad in order to permit an overall view of this subject. The book is intended both for interested nonspecialists who wish to become acquainted with the discipline of process control engineering and for process control engineers, who should find it helpful in identifying individual tasks and organizing them into a coherent whole. This objective has led the author to forego a detailed discussion of some subtopics; the interested reader can follow these up on the basis of the extensive bibliography.

One problem in a consistent treatment of the content is that different areas have reached different levels of development. For example, the area of measurement and control technology can draw on a long tradition of scientific and practical research, and the reader can consult an established, didactically oriented specialist literature. In contrast, the situation in the field of information structures is quite different. Although some methodological approaches have been described schematically, the extension of these to process control engineering and their integration into an overall structure are relatively new lines of thought and rest on a few quite recent spezialist articles. The didactic treatment of this field on the basis of examplary problems, the testing of deductions in practical applications, and the construction of a broader scientific superstructure must await the coming years.

A central concern of this treatment is to arrive at a consistent and comprehensive way of thinking about process control engineering and to show how the several specialities can be organically fitted into this total view.

The volume is organized as follows:

- The Introduction gives a history of process control engineering and shows that this discipline has grown as a logical consequence of the development of measurement and control techniques and information technology.
- Chapter 2, "Information Structures in Process Control Engineering," describes the architectural principles used to define the Field. The first part deals with classification methods, and the second, with the application of such methods to examples of information structuring in process control engineering.
- Chapter 3, "Knowledge about the Process," gives an overview of the ways in which process knowledge is acquired, organized, and systematized. Examples of important disciplines covered are statistical methods of data reduction, mathematical process models, and the information content of flowsheets or recipes.
- Chapter 4, "From Process Knowledge to Process Control," examines how knowledge about the process can be used for meaningful process control. Open and closed loop control techniques are introduced at this point, as are the various ways in which the operator can manage the process.
- Chapter 5 deals with the fundamental question of acquiring information about the product and the process and, by means of appropriate taxonomies, brings together classical industrial measurement engineering and process analysis under the heading process sensor system technology. Future field communications systems are treated in terms of installation technology in Section 5.5, while an information-logistical treatment is presented in Section 8.6.
- Chapter 6 describes the systems-engineering requirements for intervention in the process by means of actuators. Modern concepts of drive engineering and power supplies are presented.
- Chapter 7 considers the high information- and communications-technology specifications that must be met by process control systems. Possible solutions are presented for current and future control systems.

- Chapter 8 discusses modern information logistics and introduces methods and tools required for company-wide information integration.
- Chapter 9 is devoted to computer-aided methods. It describes the structure and functions of CAE systems used for process control engineering.
- Chapter 10 deals with the design and construction of control systems. It considers the organizational aspects required for the realization of such a project and the phases of its realization.
- Chapter 11, "Operation," describes the activities that must be carried out during operation of a process control system. It deals with both maintenance of the system by the specialist engineer and man–process communication by the plant operator. This chapter also explains how the process can be optimized by variation of process conditions during operation.
- Chapter 12, surveys national and international organizations and activities involved in the standardization of process control engineering.
- Finally, Chapter 13 deals with the integration of knowledge-based systems in process control engineering.

I hope this original and comprehensive overview of modern process control engineering will be received with interest.

Munich, Spring 1994 Günther Schmidt

Biographical Notes

Wolfgang Ahrens	Studied Electrical Engineering at TH Karlsruhe, Ph. D 1974. Since 1988, in Process Control Engineering Department at Bayer AG, responsible for CAE development in process control engineering.
Dietrich Balzer	Studied Electrical Engineering at TI Leningrad, Ph. D 1969. Until 1975 worked in a petrochemical plant in Schwindt. Habilitation at TH Leipzig 1976. 1975–1992 Head to Institut für Prozeßrechentechnik, TH Leipzig. Now at Elpro AG, Berlin, Systems Engineering.
Michael Björkmann	Studied Technical Physics at TU Helsinky, since 1982 in development and marketing of drive technology. Now Project Manager at ABB Strömberg Drives.
Hasso Drathen	Studied Physics at Universität Bonn, Ph. D 1975. Since 1976 Process Control Engineering Department at Bayer AG, now Head of Safety Department, Managing Director of NAMUR.
Ulrich Epple	Studied Physics at Universität Stuttgart, Ph. D 1986 at Institut für Systemdynamik und Regelungstechnik. 1986–1990 in Process Control Engineering Department at Bayer AG, since 1991 freelance consultant and Managing Director of Gesellschaft für Prozeßtechnik.
Michael Freytag	Studied Electrical Engineering/Information Technology at RWTH Aachen. 1981–1991 in Process Control Engineering Department at Bayer AG, Leverkusen, since 1991 Head of Process Control Engineering Department at Bayer AG, Antwerpen, N.V.
Ernst-Dieter Gilles	Studied Electrical Engineering at TH Darmstadt, Ph. D 1963 and Habilitation 1966 at TH Darmstadt. Since 1968 Director of Institut für Systemdynamik und Regelungstechnik in Process Technology Faculty, Universität Stuttgart.
Klaus Hartmann	Studied Physics at TH Hannover and TH Stuttgart, Ph. D at Max-Plank-Institut für Metallforschung, Stuttgart. 1972–1977 at Erwin Sick Optoelektronik, München. 1977–1985 at Bayer AG, Compur, Munich and Miles, USA, Process Control Engineering (management and plant design). 1985–1990 Head of Process Analysis Technology at Bayer AG, Leverkusen, 1990–1993 Director of Infrastructure Planning. Since 1993 Senior Vice President of Engineering, Miles, Pittsburgh, USA.
Michael Heim	Studied Energy Technology at RWTH Aachen and INSA Lyon. Graduated 1991. Now Assistant in Process Control Engineering Department at RWTH Aachen.
Reiner Hotop	Studied Physics, Ph. D at Institut für Experimentelle Physik, Universität Kiel. 1981–1993 in Process Analysis Technology Department at Bayer AG, Leverkusen. Now Head of PAT Department.
Norbert Ingendahl	Studied Mining Engineering at RWTH Aachen, graduated 1989. 1990–1991 at Institut für Bergbaukunde II, now Assistant in Process Control Engineering Department at RWTH Aachen.
Norbert Kuschernus	Studied Physics at Universität Hamburg, Ph. D 1984 at Fakultät für Chemie-Ingenieurwesen. 1985–1992 in Process Control Engineering Department at Bayer AG, since 1992 Head of Engineering Department at Bayer Ltd, Japan.
Rudolf Metz	Studied Mathematics at TH Darmstadt, Ph. D 1975. 1978–1990 software development, 1980–1986 System Engineer at Ford, responsible for CAD databank development, since 1986 in Process Control Engineering Department at Bayer AG.
Wolfgang Mutz	Studied Electrical Engineering at Universität Stuttgart. Planning and marketing of process control engineering installations at Bayer AG. Now Head of Process Control Engineering Department.
Eberhard Nicklaus	Studied Physics at Universität Münster, Ph. D in Solid-State Physics 1976. 1976–1987 in Process Analysis Technology Department at Bayer AG. From 1987 diverse responsibilities at Bayer AG.

Wolfgang Noerpel	Studied Physics at Universität Mainz, Ph. D in Molecular Physics 1974. Worked at Bayer AG in Biomedical Technology, Process Analysis Technology and Laboratory Control Technology for different Bayer plants. Now manager at PAT, Elberfeld.
Ulrich Pallaske	Studied Mathematics and Physics in Köln and Freiburg, Ph. D 1969. Since 1969 at Bayer AG, responsible for mathematical modelling, process technology, and process control.
Günter Pinkowski	After graduation, Development Engineer for hardware and software in Process Control Engineering Department at Firma Krohne, Head of Systems Technology.
Martin Polke	Studied Physics at Universität Würzburg, TH Darmstadt, Ph. D at TH Karlsruhe 1963. Since 1964 in Engineering Department of Applied Physics at Bayer AG, 1971 Controller and 1975 Technical Head of Fiber Division. 1982–1990 Head of Process Control Engineering Department. Honorary Professor at Universität Stuttgart 1987. Since 1991 Head of Process Control Engineering, RWTH Aachen.
Jörg Raisch	Studied Control Engineering at Universität Stuttgart, Ph. D 1991. Research in Department of Electrical and Computer Engineering (robust process control, decentralized control, design of hybrid control systems) at Toronto University, Canada.
Günther Schmidt	Studied Electrical Engineering at TH Darmstadt, Ph.D 1966. Worked at Dornier AG, Friedrichshafen. Since 1972 Head of Control Technology at TU München, research area automation and robotics.
Karl-Heinz Schmitt	Head of Process Control Engineering Department at Bayer AG, Krefeld-Uerdingen, until retirement in 1993.
Hans-Josef Schneider	Studied Process/Control Technology at TH Darmstadt. Since 1966 at Bayer AG, technical support of process control engineering installations (inorganic chemicals and environmental protection), Head of Radiometry. Now responsible for process control engineering regulations for process control engineering planning (CAE) in Process Control Engineering Department.
Gerd-Ulrich Spohr	Studied Physics and Electronics at Universität Bochum, Ph. D at Universität Köln 1980. 1980–1992 Process Control Engineering Department at Bayer AG, Dormagen, planning and technical support of installations, Project Leader for CAE System Development, since 1992 Head of Automation Technology Department at Siemens AG, Köln.
Hartwig Steusloff	Studied Communications/High-Frequency Technology at TH Darmstadt and TU München. 1968 Fraunhofer-Institut für Informations- und Datenverarbeitung (IITB). Development of Real Time Computer Systems for Automation Technology. 1977 Dissertation TH Karlsruhe. Since 1984 Head of IITB. 1987 Honorary Professor of Information Technology Faculty, Universität Karlsruhe. Electromechanical and Electronic Development of Analog Computer Technology.
Roland Vogelsgesang	Studied Physics at TH Karlsruhe, Ph. D 1969. Since 1970 various responsibilities at Bayer AG.
Herbert Wehlan	Studied Electrical Engineering at Universität Stuttgart, Ph. D. Worked in Process Control Engineering Department at Bayer AG. Since 1989 Professor for Process Control Engineering in Process Technology Department at Universität Stuttgart.

Contents

1. Introduction

Since the invention of the steam engine in the 1700s, the world has gone through four long economic cycles, which were described by the Russian economist Nikolai D. Kondratieff (1892–1930) and are known as Kondratieff waves or cycles [1.1].

Each upward phase (identified as "prosperity" in Fig. 1.1) was borne by inventions such as the steam engine (WATT, 1769), the mechanized weaving loom (CARTWRIGHT, 1784–1786), the steam locomotive (STEPHENSON, 1814), the electrodynamic principle (SIEMENS, 1866), the diesel and Otto motors (1893–1897, 1876), and fundamental innovations in chemical process engineering such as the manufacture of basic chemicals.

The world economy entered another long upward phase at the beginning of the 1980s.

The development of the new basic innovation called "information technology", characterized by the invention of the transistor [1.3] and the subsequent rapid development of miniaturization and laser technology, but also by results from the field of informatics such as object orientation [1.4] and Petri networks [1.5], is described in depth in [1.6]. The economic boom identified with the new resource "information" [1.7] is a global phenomenon with heretofore unknown dynamic qualities. International business, national economies, and individual companies find in it growth and innovation potential that could scarcely have been imagined. PORTER describes the importance of information as an essential resource for a successful corporate strategy [1.8]: "*The question today is not whether information technology has significant effects on a company's competitive situation, but merely when and how these effects will become apparent.... Anyone who does not react today will be compelled to follow a course that others have set.*"

The fifth Kondratieff cycle, however, is not only an imposing technical and economic upswing but carries along with it profound social, ecological, cultural, and intellectual changes and challenges. These connections, often revolutionary in scope, were pointed out by SCHUMPETER [1.9].

The fifth Kondratieff cycle will render permanent changes in the way the individual thinks and acts and also into the strategies that corporations and national and international institutions will pursue. A technological advance will become a structural change [1.10].

Along with the classical production factors—energy, labor, raw materials, and proven production technology—information has become not just an additional factor but probably the most important factor.

CIM (computer-integrated manufacturing) and CIP (computer-integrated processing) are not merely slogans today [1.11]. Demands made by society, such as increased environmental awareness, better and more consistent product quality, improved delivery, and more economical management, necessitate production facilities and processes with greater flexibility and ease of operation.

Strategy, that is, the management of the company as a whole, means identifying goals and achieving them. Because a company's definition of goals must always be dynamic, it is important to inquire into what impact changes in the company's environment will have. As trends in socio-

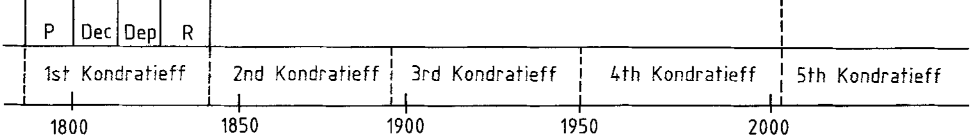


Figure 1.1. The Kondratieff waves [1.2]
P = prosperity; Dec = decline; Dep = depression; R = recovery

cultural, ecological, technological, domestic economic, international economic, and political-legislative settings become stronger and change more rapidly, optimal access to information becomes a central requirement for a successful strategy (e.g., in marketing, sales, and production).

But strategy also means making available the means for business success. An essential success factor is thus the exploitation of new information technologies based on engineering, physics, and information science.

Information is an unlimited resource that can be managed only if it can be structured. In the economic, administrative, and technical spheres, those responsible for information management must optimize the utilization of information by coordinating all parties involved and by creating a technical infrastructure.

The German term *Prozessleittechnik* (process control engineering) was coined at Bayer AG in 1980 as a working title covering the measurement, control, and electrical engineering groups [1.12]. The expression was chosen to denote the integral view of all those concerned with the management or control of production programs based on engineering processes.

Since that time, a number of events have made it clear that classical or signal-oriented measurement and control technology is being supplanted by information-oriented process control engineering.

Long discussions among experts regarding the terms "control" and "automation" came to an end when LAUBER stated a valuable dichotomy, placing humans at the center of the analysis [1.13]: The role of humans can be (1) to specify the framework and the sequence of the process, which then runs automatically, or (2) to gain the maximum information about the process and intervene in it directly, becoming the agent that ultimately controls it. This question is discussed in more depth in Section 4.3. The authors of this article have a clear preference for the more universal term "control" and make use of various levels or degrees of automation (see also [1.14]).

"Essential impetus for development today is coming from information technology, which has become a significant tool in measurement and automation technology. Only information-oriented technology enables the industrial user to operate processes and plants in compliance with the criteria of flexibility, productivity, safety, and environmental protection. The integration of humans into the production process is of great importance if the operator must, for example, perform control functions at the interfaces with the automation system and with the process" [1.15].

The national and international standardization organizations of greatest relevance to process control engineering, such as the International Electrotechnical Commission (IEC) with its Technical Committee TC 65 and the German Electrotechnical Commission (DKE) have also replaced the term "measurement and control technology" with the new complex of "control engineering" (see Chap. 9). The Standardization Working Group on Measurement and Control in the Chemical Industry (NAMUR) had already announced such a new orientation in September 1987 [1.11].

The strong influence of information technology on automation technology and process control engineering is visible not just in the area of devices and systems (especially process control systems; see Section 5.2), but in all aspects of design, construction, operation, maintenance, and so forth (procedures) [1.16].

BASF has made their training programs in operational practice available to the public [1.17]. Detailed procedural instructions can be found in STROHRMANN's [1.18] course on process control engineering (serialized in *atp* since 1984). GRUHN et al. [1.19] and LAUBER [1.13] have published accounts of structuring practices for automation with an emphasis on process engineering and process computing automation technology, respectively. TÖPFER and BESCH have also described principles [1.20].

The time now seems ripe for attempting to present an integrated, information-oriented exposition of this subject.

The information structures employed in control engineering and the architectural principles that generate them are therefore discussed and explained with practical examples (Chap. 2).

Next, the computational and mathematical methods used to acquire, reduce, model, and document process information are presented (Chap. 3).

Information about the process is used for process control (Chap. 4) by applying isolated and integrated control concepts and object-oriented process-control strategies (e.g., recipe management).

The control components of production plants, that is, the electrotechnical devices used for process control (sensor and actuator systems, process control systems including communication technology, process power supply, design of control and power distribution rooms and cable runs), are next analyzed from the information-technology point of view (Chaps. 5–8).

Computer-aided techniques (C-methods) have proved an especially useful information-technological support for the engineering of process control systems. General aspects of such systems and detailed structures of data-model-oriented design aids now in use or under development are described in Chapter 9.

The procedures used for the design, construction, and start up of process control facilities are set forth from a systems viewpoint and on a structural and sequential basis. Decision phases with the most important design document, (requirements and specifications) are clearly distinguished from execution phases. Quality assurance for the design, construction, and start up of process control systems is described in Chapter 10.

In the operation of production plants with process control systems, a change is taking place from signal-oriented man–machine communication to information-oriented, state-based man–process communication. This enables the plant operator to utilize knowledge about the process, formulated in an ergonomically correct way, to control the process. Continuous process analysis and process optimization take on new importance in this context. Strategies for maintaining or restoring the availability of process control functions are described in Chapter 11.

The current status of standardization of process control techniques and systems in national and international bodies, societies, and institutions is presented in Chapter 12.

Finally, the integration of knowledge-based systems and modern concepts of fuzzy logic and neural networks are discussed in Chapter 13.

2. Information Structures in Process Control Engineering

2.1. Principles

Information as a Corporate Resource. In principle, information is an unlimited resource that can be managed only if it is structured. It is the task of information management to optimize the utilization of information by insuring the efficient interplay of all parties involved and providing the technical infrastructure.

The objective of information structuring is to reduce the degree of complexity by appropriate modeling (formation of data models and functional models) and thus create the prerequisites for information processing (input/output, storage, and analysis) by means of information systems. What has to be overcome, above all, is not the material but rather the inadequacy of the human intellect to deal with complexity.

The importance of adopting suitable structuring aids is demonstrated by the way in which technical, administrative, management, and logistical data "circles"—none, as a rule, consistent with the others—frequently come together.

Because they continue to be developed, application systems arise whose functionalities often overlap (Fig. 2.1) [2.1].

What is more, the demand for information depends very heavily on the level of automation present in individual production facilities. Production engineering systems often feature non-communicating automation islands. In fact, all such information circles are necessary, from the technical and economic standpoint, in order for the corporation to accomplish its task unhindered by departmentalization.

If information as a corporate resource is to be optimally deployed, it is essential to integrate information technology, and this can be done only when information concepts have been harmonized.

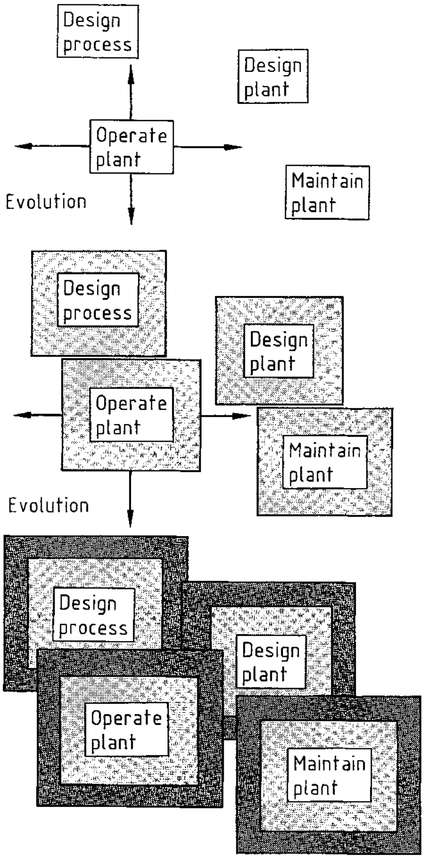


Figure 2.1. Convergence of isolated data "circles" and automation islands [2.1]

Integration of Information/Information Models and of Data/Data Models. Information integration is achieved by consolidating information across projects, leading to a mandatory corporate information model [2.2]–[2.5].

Such information models [2.6] are said to be "semantic" or "conceptual" because, at the user level, they reflect the definitions and interrelationships of concepts but not the realization of information in data-processing (DP) terms or the selection of information for specific applications. In contrast to earlier management information system (MIS) implementations, the result is an extremely stable information model. Appropriate representation techniques yield data from information and data models from information models.

Since 1970, theoretical publications on the form of such data models have been concerned chiefly with how the organizational problem of large data-base systems can be solved. For example, the mathematically grounded relational data model with its normal forms [2.7] was created first; the entity–relationship (ER) model, which stresses the semantic aspect of the data, came later [2.8].

In 1975, the ANSI/X3/SPARC DBMS Study Group introduced the "conceptual scheme", a comprehensive, implementation-neutral structure description of corporate information. This conceptual scheme is of central importance for the understanding and management of information systems [2.2], [2.9]; see also the SPARTEN information and control systems (ISS-$\alpha\alpha$) developed at Bayer AG [2.10].

Further detailed exploration and discussion of the conceptual scheme has taken place in the ISO Working Group with the same title. A detailed report is presented in [2.11].

Work on these aspects has continued; some results have been summarized and various methods, procedures, and tools to aid in the creation of data models have been set forth [2.12]–[2.15].

Today, the state of the art is as follows: In the development of information systems, data properties relating to content, application logic, and organizational aspects are first examined in a "data structure analysis." The results must be described in a semantic or conceptual information model before the data model (which depends on the form of presentation) and thence the data-base design can be derived. The data-base design must take account of requirements and restrictions having to do with data-processing hardware and software. Restrictions based on processing considerations are brought in at another level.

The data-base design can thus be represented in a three-level model:

- Conceptual level: The result is the semantic or conceptual information model
- Logical level: The result is the logical data model, which contains restrictions having to do with data-base hardware and software
- Physical level: The result is the physical data model, which makes allowance for processing restrictions

A conceptual information model is constructed with an eye not to isolated information-processing functions, but to the objects of corporate information processing that are common to all such functions. This is the strategic dimension of data modeling.

Such conceptual corporate data models are now being created in many companies. As a corporation-wide, application-neutral, implementation-neutral structural description of corporate information, such a scheme also constitutes the common linguistic basis for communications between persons involved in DP management. Because it is independent of the allocation of data among the company's computers, independent of physical data storage, and independent of the data-base systems employed, such a scheme can unify information systems based on a wide variety of hardware and software.

Integration of Methods. In the first period of software engineering, the software life-cycle model was in the forefront of discussion. Software development environments were designed in accordance with this philosophy. Alternative philosophies are in use today, such as rapid prototyping and participative system development [2.16]. Both aim at the early involvement of end users, whose acceptance is the single decisive criterion for the success or failure of an application system.

The appearance of data bases gave new impetus to the development of data modeling techniques. Because these are based on a variety of implementations, they are largely incompatible. For example, an elevator control system does not require a data-base design; an information system has no time-critical concurrent processes; and an expert system uses different knowledge representations from a procedural program.

If the data model has a hierarchical, network, or relational structure, different design methods are used.

An integrated methodology must generate both the data design and the functional design (Fig. 2.2).

It has often been overlooked that every design level of this methodological concept deals with a different aspect of the problem.

Thus, data and functional models are built up at the uppermost, conceptual level and then, in the further specification, broken down into a representation that can run on certain hardware.

Integration of Representation Means. The basic prerequisite for the common storage of design data and the transfer of information between methods is an appropriate representation form.

Today, graphical and textual representation means can be thought of as being diametrically opposed.

Each, however, has its justification. Graphical means of representation (flowchart, function diagram, entity-relationship diagram, polling

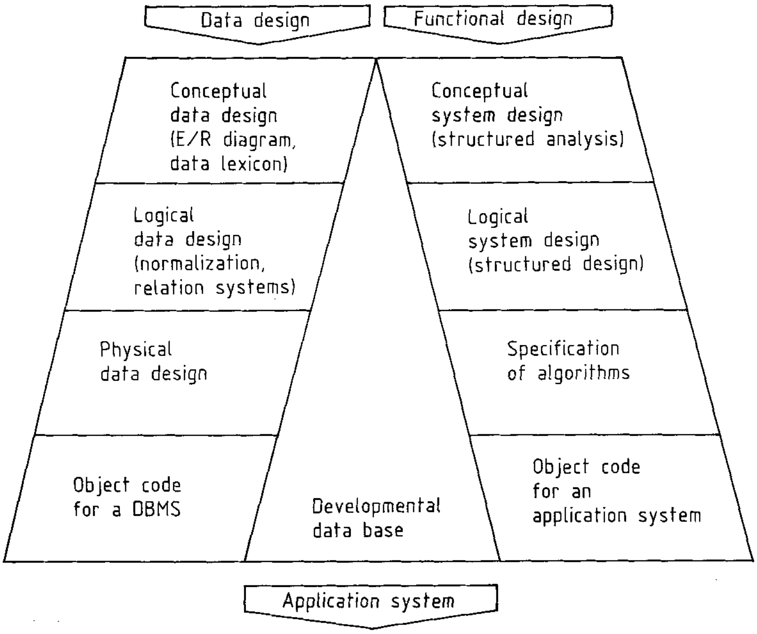


Figure 2.2. Methodological concept [2.17]

hierarchies, etc.) are a suitable idiom for use in the human–computer interface (Table 2.1).

Textual means of representation (pseudocode, structured text, computer languages of various generations) yield expressions that are understandable chiefly to the computer.

There must therefore be a way to translate between graphical and textual means of representation.

The task remains to harmonize these means of representation with one another, obtaining a canonical set of description means that can be used to describe data structures and functional structures with regard to both static and dynamic properties.

Integration of Tools. The most important tools for data integration are data-base systems and data dictionaries [2.4], [2.19].

In order that the data in a data base will represent a consistent system of facts, the permissible and necessary relationships between data are specified by means of integrity conditions and monitored and controlled in real time by a data-base management system. Data-base systems offer a range of concepts that support "information management."

In particular, these include concepts for modeling and manipulating data in an application-oriented fashion (data models), for integrating various applications over the same data base without redundancy (data integration), for making an application neutral to changes in the logical and physical organization of the data (data independence), for insuring data consistency, multiuser synchronization, and data security, and for safeguarding the stored data against unauthorized access.

A data-base system is said to be of hierarchical, network, or relational type, depending on the underlying data model.

Hierarchical and network data-base systems have become established in commercial, administrative applications. Relational data-base systems are gaining ground in engineering applications, such as the support of design processes (in the design of mechanical parts, LSI circuits, process monitoring and control equipment, and software development) and process control. Distributed and object-oriented data-base systems have not yet become widely available from commercial sources.

In contrast to data-base systems, a data dictionary is a system for storing and processing data definitions (data descriptions) for one or more software systems. In its simplest form, a data dictionary can be regarded as a well-organized notebook containing basic information

Table 2.1. Methods and their points of emphasis [2.18]

Method	Points of emphasis
Structured Analysis (SA)	Representation of functions, data regions, interfaces, and data flows connecting them, in the form of hierarchically organizable nets
Structured Design (SD)	Representation of modules and their interfaces in "Structured Charts" and module level model
SADT	Representation of functions, as well as data and control flows connecting them, in the form of hierarchically organizable nets
Entity-Relationship Model	Representation of objects and their quantifiable relations (1:1, 1:n, etc.) in the form of nets
Decision Tables	Representation showing under what combination of conditions certain actions can be performed, in the form of tables
Finite-State Automata	Representation of states and state transitions, and also of inputs initiating changes of state as well as outputs initiated upon changes of state, in the form of nets
Petri Nets	Representation of states and events in the form of nets whose behavior can be dynamically investigated with the aid of the Petri net rules
ISAC	Representation of information sets, as well as information flows and functional sequences, in the form of seven types of nets
Jackson Structured Programming (JSP)	Representation of the data structure of a problem, derivation of the program structure from this data structure
Jackson System Development (JSD)	Representation of objects and their actions in the form of trees, and of the connections between objects in the form of nets
MASCOT	Representation of processes (and devices) and their communication regions in nets
Functional nets	Representation of processes and devices as well as their communication, in the form of nets
HIPO	Tree-type breakdown of a system and definition of every part in terms of input and output data and required operations (steps)

about all the data elements stored in a computer for an application area [2.20]-[2.22].

These support not only a one-time, uniform, consistent documentation of information objects, but also the management of other objects used in the DP environment (such as jobs, projects, users, etc.) as well as their relationships. Data dictionary systems are themselves data-base systems that use a central data base to manage all information needed to describe, process, and use the objects of an application area, regardless of their internal organizational structure. Data dictionary systems also support data management for application systems and thus contribute to the efficient development, implementation, and description of software systems.

A number of other tools have been devised in the course of the software engineering discussion. Each of these tools is suited to a certain class of problems; each either extends over multiple phases or is specific to one phase [2.6], [2.23], [2.24].

For example, editors are for the creation of graphics and texts for use in a variety of methods; compilers, linkers, debuggers, and so forth, for the implementation phase; test tools, for the testing phase.

Tools for retrospective documentation are becoming more and more important at present.

Decades of software development have produced software systems that can scarcely be maintained any more (the "software crisis"). The documentation of these systems is often in a wretched state.

Because of changes in methodology, it is no longer sufficient to document such systems in the context of their own (usually obsolete) approach. What should be aimed for is to create new documentation for old systems, with an eye to future trends in software engineering.

Retrospective documentation is a more and more pressing need in a management context, where decades' worth of investment in software must be safeguarded. This does not necessarily mean that obsolete software systems must continue to be used in their existing form. The minimum requirement is that the information and technical knowledge contained in the system must be revealed and made accessible for new developments.

The integration of tools means that documents can be passed along from phase to phase and from tool to tool.

Integration of tools can succeed only if information, data, and methods have been integrated. This applies not just to all engineering activities during the lifetime of a technical device (see Chap. 10), but also to the design of monitoring and control equipment itself, especially of process control engineering systems (see Chap. 7).

2.2. Architectural Principles for Information Structuring

Systems and Architectural Principles. Real-world systems and elements have internal structures. Along with Aristotle's doctrine of categories [2.25], the Linnaean system—still a valid descriptive model employed in botany—provides an example [2.26]. Darwin's ideas on the origin of species [2.27] also illustrate this structuring process.

In the 1700s, "system" at first meant a collection of linked truths. The links had to be methodologically correct and completely deducible from an underlying principle.

KANT's concept of system stresses the theory of systematics. In his "Critique of Pure Reason," he emphasizes the significance of architecture as the art of systems.

Thus a system, traditionally, is a closed whole whose parts are interrelated, interact, and satisfy certain constraints.

The parts, regarded as elementary, are called elements of the system. A system consists of elements with properties; the elements are structured and linked together by the relationships in accordance with a system principle. Systematization involves specifying system principles and devising system structures. Systems are characterized by the fact that they have properties that cannot always be accounted for by the properties of their elements. Thus simply collecting elements does not form a system. Instead, each system has a structure that is generated by the relationships among its elements.

Systems can be subdivided into parts on as many levels as desired, whereby all parts that are not elements of the system are referred to as subsystems. System elements are the parts of a system whose further subdivision is impossible or undesired.

The sections that follow state principles for the structuring of information. They are called architectural principles because they can be identified at various points in the design process and in various representations of the information system, thus marking the architecture of the information system. The first principles discussed are general ones in the phases of design and implementation.

Principles in the "definition of requirements" phase, quality assurance in parallel with other activities, and operation of a facility might be added here. These are also referred to as construction principles, since they are applied mainly in the construction of DP systems.

This is demonstrated by the example of modern operator interfaces, where structuring of information is used to support human–process communication (see Section 11.2).

Higher concepts of functional and data structuring also make use of the design principles discussed above but represent a more application-oriented aspect of structuring.

The focus is on the object-oriented formulation, which can be applied to both functions and substances [2.28] in an integrated methodological concept. It is the tool for the decomposition principle, with which complicated objects are broken down into their constituent parts.

The next most important structuring principle is abstraction, which is a fundamental approach to concept formation. Adopted recently from expert-systems technology, abstraction is introduced as a construction principle of application systems through class-forming, concept-forming, complex-forming, and functional abstraction. Closely related to the principle of abstraction is that of inheritance [2.29]–[2.31].

The last section deals with transformation principles. Here the various models used for problem-solving are described, starting with procedurally oriented formulations (with their counterpart in functional design) and passing through logic-oriented formulations (as now employed by many expert systems) to state-oriented or event-oriented concepts (which come much closer to process-related DP problems).

State-oriented and event-oriented formulations in turn are closely related to object-oriented ones. Objects are in certain states and communicate with one another via established protocols (see Chap. 8).

The complexity of present-day systems increasingly calls for a holistic, network style of thinking [2.32], [2.33] if the basic laws governing systems, their dependences, and their interactions are to be understood. It will take a new kind of thinking to comprehend the dynamics of a continuous process, plan a monitoring and control system for it, erect and operate the system, employ higher-level information systems, and in the process not overlook human beings [2.34].

The Decomposition Principle. *Design Principles.* Structuring principles occur in all phases of system development. In the design phase, for example, the top-down and bottom-up principles are often encountered. In the top-down approach, a whole is broken down into parts step by step, whereby the complexity of the parts under consideration is systematically decreased (decomposition principle).

In bottom-up design, single parts are assembled step by step and made into a whole.

In practice, mixed forms are often encountered (Fig. 2.3).

Bottom-up design comes into play when prefabricated elements are available from which the overall systems can be assembled. This principle is often used in mathematical and statistical applications, where the elementary mathematical and statistical functions are known and can be made available in program libraries. New applications can be analyzed only by the top-down approach, though there is a trend toward reusable elements here too (see, e.g., Section 4.5).

Implementation Principles. The implementation phase also exhibits principles that ultimately determine the structure and thus the understandability and maintainability of software systems.

The more ordered the structures are, the better they lend themselves to verification, modification, and maintenance. The simplest ordering is linear. Iterations, recursions, and branches generate more complex structures (Fig. 2.4).

Recursive data and functional structures are preferred today because of their mathematical compactness. It can be shown, however, that iterative and recursive structures are equally powerful and can be transformed to one another.

Another type of structure is the directed tree with strict hierarchy or polyhierarchy, which ultimately becomes the general network if the rigorous requirement that every node has only a single input is dropped (Fig. 2.5). The last is es-

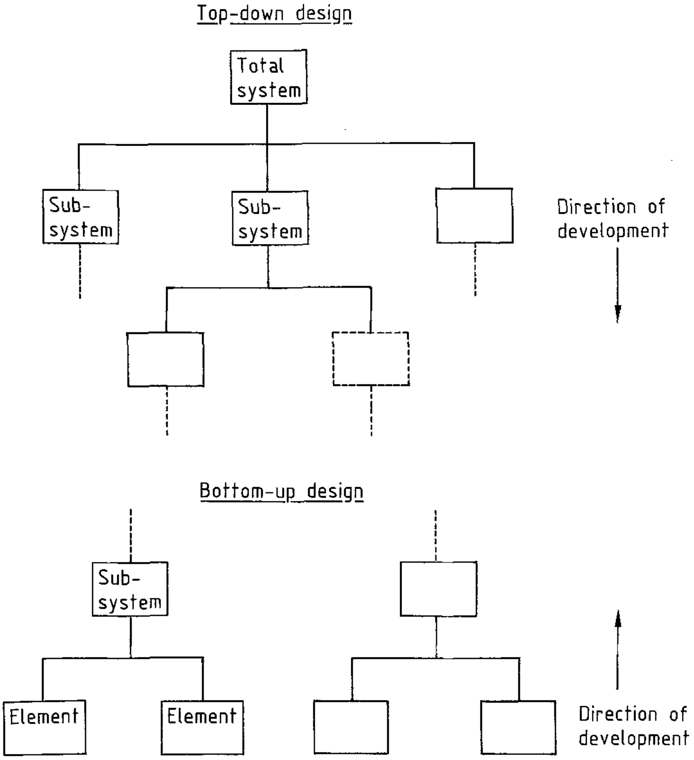


Figure 2.3. Design principles

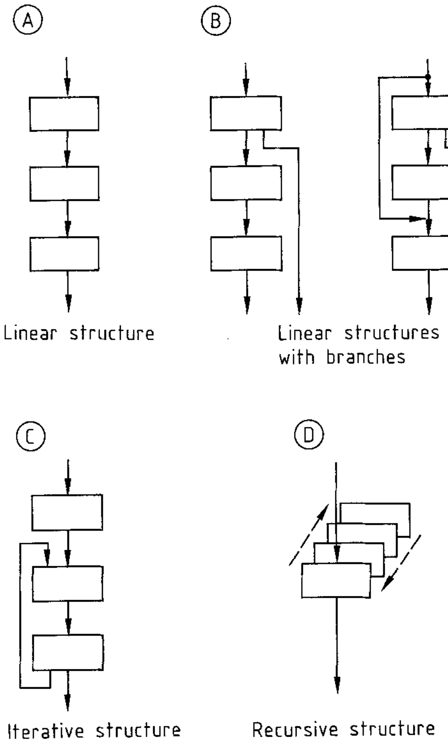


Figure 2.4. Types of structure

pecially important in so-called master–slave structures. The slave should have only one master, since otherwise there can be conflicts in execution. Ring and star structures form a separate class.

Principles for User Interface Design. The user gains access to the system through the user interface. A number of structuring principles apply to this interface, not all of which can be discussed here.

Design principles such as the law of like form, the law of color, or the law of proximity (Fig. 2.6) should be employed in the layout of masks and printouts [2.35], [2.36].

Principles such as conformity to expectation in dialog with the system (also called "principle of minimum surprise"), consistency of dialog elements, and visualization of the application context also play an important part.

Modern software ergonomics focuses on job design for people, a process that is heavily influenced by software [2.37], [2.38].

An important goal in software writing must be the design of complete activities. A complete activity has been defined as an activity that, together with simple execution functions, includes the following:

- Preparatory functions: identification of goals, development of procedures
- Organizational functions: agreement with other workers as to tasks
- Monitoring functions: feedback to the worker on attainment of goals

Incomplete activities are activities where there is essentially no possibility of independent goal identification and decisionmaking, no scope for individual ways of performing tasks, or no adequate feedback. Shortcomings in job design impair the motivational and learning potentials of the work process and affect the worker's welfare and job satisfaction. Intimately linked with the completeness of activities is the degree to which the worker has opportunities to make decisions as to time and substance, that is, freedom of action.

This freedom of action must permit creative users of an information system to devise an individualized procedure in accordance with their experience and working style. At the very least, defining freedom of action should not be part of the program author's responsibility; it is a function of corporate management and personnel practices. Observed acceptance problems often stem from misjudgments in this area.

A new type of human–computer interaction has come into being with the new class of workstations. This user interface has the new feature that it presents the user with graphical images of customary objects belonging to the office world. Such objects can be selected and then manipulated with the aid of a pointing device (mouse, light pen).

Shneiderman [2.39] introduces "direct manipulation" as a collective term for several new principles in the user interface. The most important principles are the following [2.40], [2.41]:

- Permanent visibility of the object as an icon or pictogram
- Quick, reversible, one-step user actions with immediate feedback
- Replacement of complicated commands with physical actions (see also Section 11.2)

Higher Concepts of Functional and Data Structuring. Historically, the functional structuring of application systems was emphasized in the early period of data processing. A complex function was successively broken down into partial functions until elementary functions realizable in

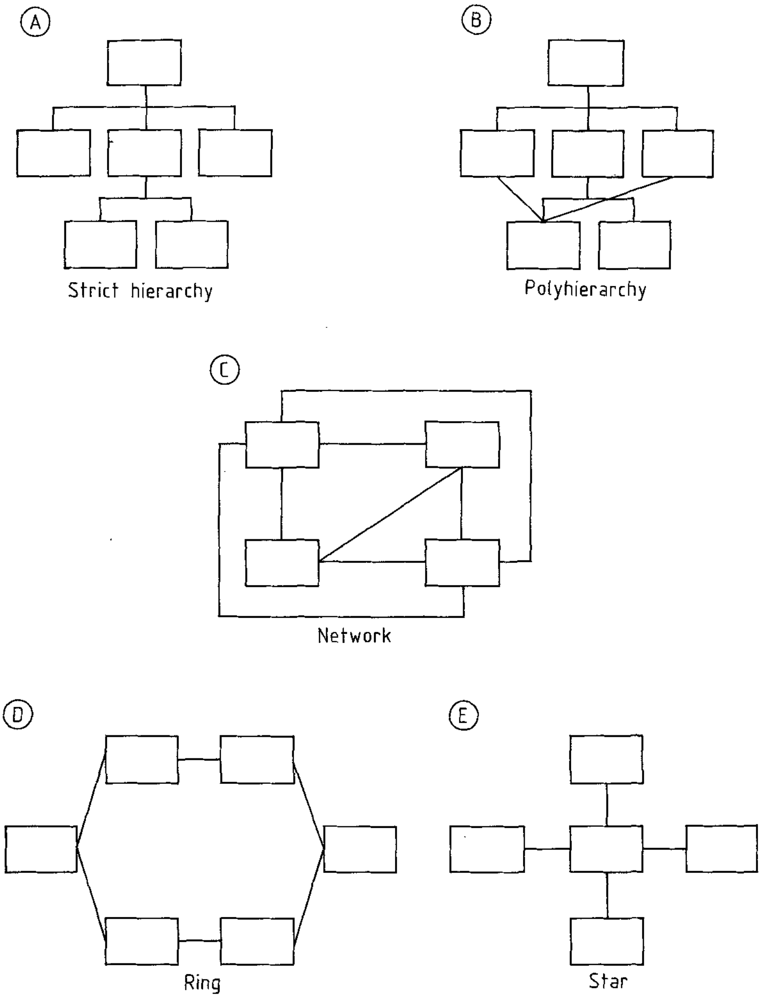


Figure 2.5. Types of tree and network structure

DP hardware and software were arrived at (flowcharts).

Data were fitted into the functional structure but remained meager, as they had to be given in the programming languages known at the time.

The structuring principle was oriented to the procedure employed in problem-solving. As a result, functional structuring led to process-oriented or program-oriented structures (Fig. 2.7).

Even today, functional structuring is still the classical approach, because it leads directly to realizable, procedurally-oriented solutions. At Bayer, for example, all engineering tasks were modeled by the SADT method, somewhat as shown in Figure 9.1. [2.43], [2.44].

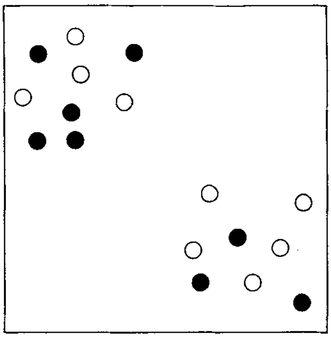
Thus the modeling concept (functional structuring) and the implementation concept (procedural programming language) correspond.

Data structuring is of secondary importance: What is interesting is the processing of the data, and data-base systems were not yet in use at this time.

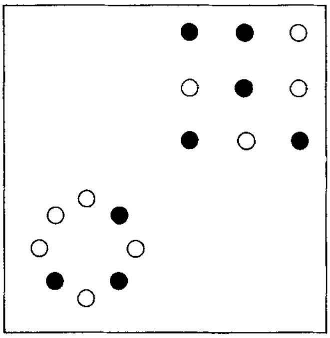
However, functional structuring has several weaknesses:

- There are scarcely any objective criteria for breaking the function down into partial functions. Decomposition is often a matter of the developer's discretion.
- The resulting data structures unavoidably grow out of the functional structure and are

Design law of:
-Proximity



Design laws of:
-Proximity
-Symmetry



Design laws of:
-Proximity
-Symmetry
-Similarity

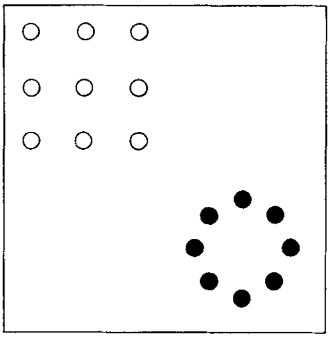


Figure 2.6. Simple design laws [2.35]

monolithically bound to it. Any change in the data structure impacts on the functional structure, and vice versa.

- Frequently, insufficient stress is laid on reusability, since it cannot be known how and in what connection an arbitrarily defined function or an arbitrarily defined module may be used again—in contrast to mathematical and statistical program libraries, where reuse is straightforward. The complexity of such a DP system increases more than proportionally to the functionality.

In the early 1970s, greater value was accorded to data structuring. This was the period of developments such as the PASCAL programming language and the first hierarchical data-base systems. Data structuring became more important.

Subsequent development is marked by attempts to shift the meaning of the data (the information) out of the programs and into the stored data, and to generate program systems in a manner directly opposed to the functional approach. Now the first step in system development is to design the data model on which the later program systems will work.

One speaks of data-oriented or object-oriented structuring, since the analysis is focused on information objects (data structures) as representations of real or abstract objects.

Objects are described and related to one another through their properties, also called attributes. Typical relationships are abstraction and mapping (correspondence) relationships [2.3].

The result of this approach to structuring is the semantic data model, which is designed independently of the functions defined on it [2.12], [2.15].

In his epistemological discussions, Wendt [2.13] classifies objects as concrete and abstract. Concrete objects include things and processes; abstract objects include concepts (see [2.28] and Fig. 2.8).

After the existence of objects and the properties of objects, the relationships of objects are taken up third. A first type of object relationship is described by ordering, which is arrived at through a comparison of some property.

Other important object relationships are part-whole relationships and instrumentality relationships. An instrumentality relationship is said to exist between a thing (e.g., a process facility, a process monitoring and control system) and a process (e.g., a continuous process, a control process) if the thing plays a part in the process.

Inseparably connected to processes are states. States include the initial and final times of processes or subprocesses.

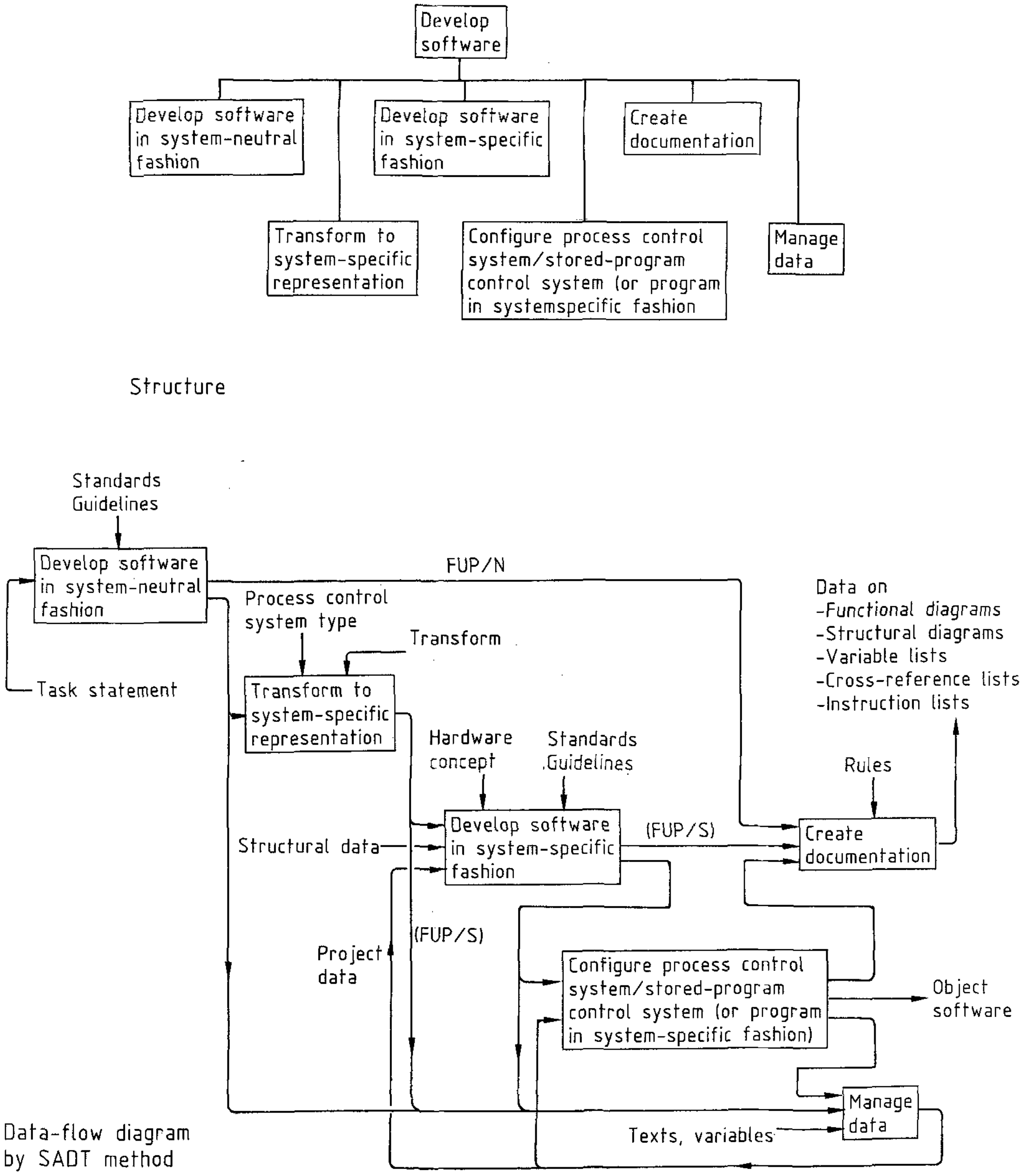


Figure 2.7. Functional structuring for the example "develop software" [2.42]

If countability is associated with the concept of object, and measurability with the concept of property, then the concept associated with the concept of relationship is decidability [2.13]. Objects are counted, properties measured or described (see "Scales" in Chapter 3), and relationships are decided.

Thus the three fundamental concepts—object, properties, and relationship—have been introduced. These make it possible to structure experience, and thus they govern the world of human thought [2.13], [2.45], [2.46].

Concrete objects are the plant, the apparatus, the process monitoring and control system, the production process, the reaction process, the sensor–actuator process, the data-processing process, and so forth. This group of objects is also called "devices"; devices must be designed,

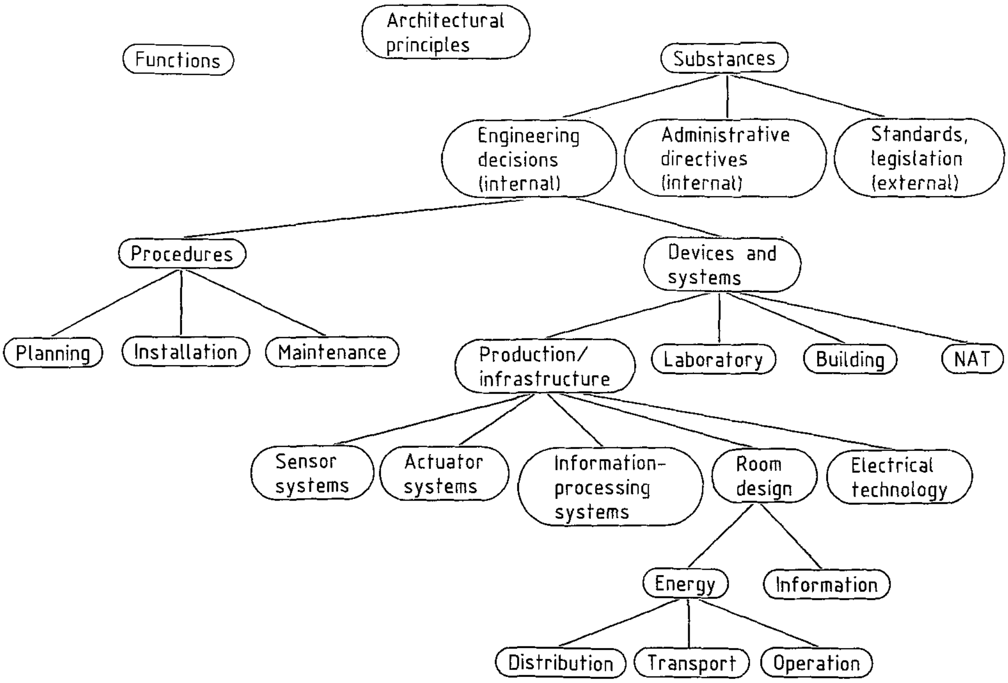


Figure 2.8. Structure of PCE propositions

built operated, maintained, and optimized, these activities being known as "procedures." The outcomes of these procedures are represented as flowcharts, construction drawings, process monitoring and control station diagrams, and so forth. In the Cassirer scheme, both groups of objects are called substances. Functions that describe a task or activity (e.g., plant/apparatus coding; see Section 2.3) also have a representation in the object-oriented formulation (see Figs. 2.22, 2.24, and 3.26).

Functions (i.e., tasks) are implemented through procedures (abstract objects) from devices (concrete objects). Figure 2.8 gives a large-scale survey of "object worlds." For clarity in terminology, relationships are described not with the function concept usual in informatics, but with the epistemologically based relation concept (cf. entity relationship) [2.28].

The abstract representations are the objects that can be manipulated in the computer. They always relate to an object, which they describe from a certain abstract point of view.

The object-oriented approach thus represents a "natural" modularization concept, since the objects can be extracted directly from the application. If the functions are likewise structured in an object-oriented manner, this constitutes a principle of functional structuring that is far less sensitive to modifications.

The Principle of Abstraction. The (provisionally) last step in development combines relational and object-oriented structuring. In the "abstract" data type, data structures and relational structures are merged into a whole. The relations enclose the data as if in a shell; the data can be manipulated only through the relations "guarding" them. The high modularity of these constructs suggests that there may be significant gains in the reusability and maintenance of software. This principle is referred to as data encapsulation or "information hiding" [2.31], [2.47]–[2.49].

The structuring principle is thus concerned with the structuring of data and relations, their (static) structural and (dynamic) sequential structures, and their interactions.

Usually, the sensory or empirical view of things is said to be "concrete," while any description in terms of concepts is said to be "abstract" (see also [2.13]).

Abstraction means reducing complexity by neglecting all inessential properties or features. One also speaks of models.

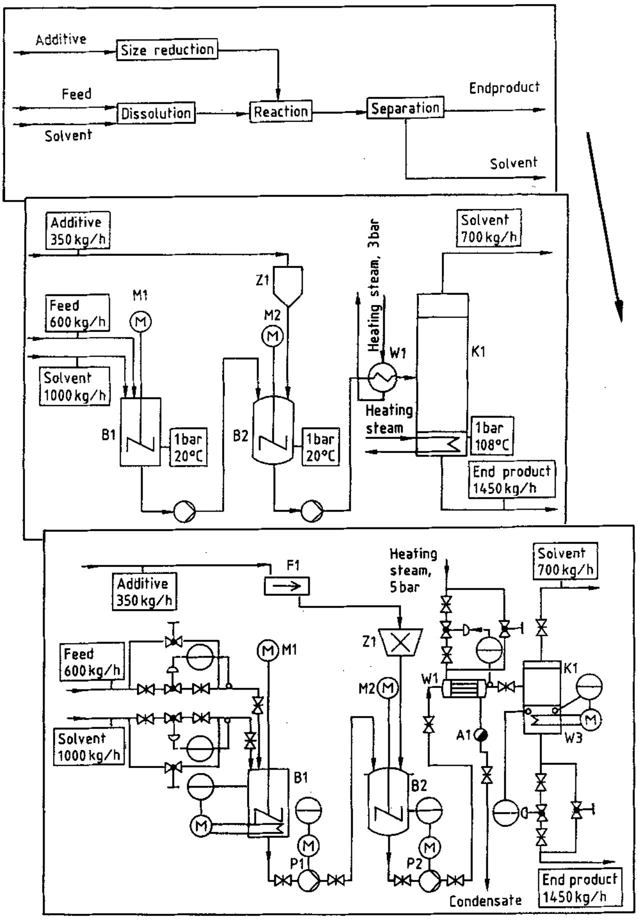


Figure 2.9. Design steps in process engineering [2.50]

Models are always simplified pictures of the real objects (abstractions) as viewed from a certain aspect (see also Sections 3.1 and 3.3).

In process engineering, flowsheets are models of a real facility in graphical form. They are standardized and represent the medium of communication for persons involved in the design, construction, and operation of plants (Fig. 2.9).

Mathematical models in the form of systems of algebraic and differential equations depict partial aspects of the physicochemical and apparatus-specific parts of processes. Such models are used, for example, in model-aided measurement techniques for the control of processes, the detection of faults, and simulation (see Chaps. 3 and 4).

A plant model is a representational picture of the plant that is much favored as an object of study by plant designers and operators.

The three-dimensional representation of plants in CAD systems is a way of getting the advantages of such models while using only DP resources.

Along with mathematics, the field of informatics now offers methods with which complex engineering systems can be modeled, simulated, and optimized.

It must not be overlooked that data models are always reductions from real systems. Models, no matter what kind or in what medium, represent a given system from various points of view. They emphasize essential properties and suppress inessential ones. Within these limitations, however, models are an indispensable part of any modern information technology.

Class-Forming Abstraction. Object orientation means that once the objects have been identified, they are then described in terms of their