Pergamon

Computers chem. Engng Vol. 20, No. 6/7, pp. 591–609, 1996

0098-1354(95)00195-6
0098-1354/96 $15.00 + 0.00

TRENDS IN COMPUTER-AIDED PROCESS MODELING*

W. MARQUARDT
Process Engineering, RWTH Aachen University of Technology, D-52056, Aachen, Germany

(Received 7 April 1995; received for publication 9 May 1995)

Abstract—Process modeling is an important task during many process engineering activities such as steady-state and dynamic process simulation, process synthesis or control system design and implementation. The demand for models of varying detail is expected to steadily increase in the future due to advances in model-based process engineering methodologies. Computer assistance to support the development and implementation of adequate and transparent models is indispensable to minimize the engineering effort. The state of the art and current trends in computer-aided modeling are presented in this contribution which is intended to serve as a survey and a tutorial at the same time.

1. INTRODUCTION

Environment and safety regulations, growing demands on product quality as well as increasingly competitive markets necessitate continuous improvement of chemical processes in minimal time at minimal cost. One approach towards improved processes is a detailed evaluation of a larger number of flowsheet alternatives during conceptual design. Another strategy builds on an increase in the experimental effort to support the selection of unit operations, their scale-up or the development of process operation policies and control systems. In general, both approaches contradict the objective of minimizing the development time to respond to the needs of the markets with minimum delay. In many cases, development time and rather reasonable than minimal specific production cost decides whether decent profits are possible. Hence, theoretical or experimental evaluations of process alternatives are only possible within tight time and cost constraints.

The number of experiments during scale-up or even during commissioning of the plant can be reduced if increasingly detailed models of all kinds of unit operations become available. The modeling requirements for process design are quite different from those aiming at a partial replacement of experiments. Instead of always employing a detailed model with high predictive capabilities a suite of models with increasing detail and fidelity can advantageously be used during the screening of process alternatives. Crude models suffice to discriminate among alternatives in the early stages with limited computational effort whereas more and more accurate models are required as the conceptual design progresses towards the final flowsheet. Dynamic models which are consistent with stationary design models are indispensable if a tight integration of process design, control and operation is envisioned.

Hence, model families with varying degree of detail are required for every unit operation in order to meet the needs of the diverse process engineering activities. The development of such model families as well as their structured storage in some archive for later reuse is a major undertaking which needs to be supported to the extent possible by information technology. Their proper application is demanding and therefore needs to be effectively supported by software tools. Such modeling tools and model libraries should form self-contained modules to be integrated into open, computer-aided process engineering environments. Model representation must allow various kinds of interpretation such as steady-state and dynamic simulation as well as optimization to effectively support model-based analysis and synthesis of processes and their control systems.

The state of the art as well as future trends in process modeling and simulation have been reviewed in a number of recent contributions from different perspectives (e.g. Biegler, 1989; Marquardt, 1991; Boston *et al.*, 1993; Pantelides and Barton, 1993; Pantelides and Britt, 1994; Schuler, 1994). As more and more methodologies and numerical algorithms become available modeling is expected to become the major bottleneck in the widespread use of model-based techniques in industrial practice (Marquardt, 1992a). Therefore, this review will focus largely on computer-aided mathematical modeling regardless of the type of application the model is intended to be used with emphasis on concepts for the structuring of process models and of the modeling process as well as on object-

* Revised version of a paper presented at *PSE'94*.

oriented formalization for model base implementation.

2. MODELING TOOLS — AN OVERVIEW

The modeling tools in current simulators may roughly be classified into two groups (Marquardt, 1992a; Boston *et al.*, 1993): block-oriented (or modular) and equation-oriented approaches.†

Block-oriented approaches mainly address modeling on the flowsheet level. Every process is abstracted by a block diagram consisting of standardized blocks which model the behavior of a process unit or a part of it. All the blocks are linked by signal-like connections representing the flow of information, material and energy employing standardized interface and stream formats.

Models of process units are precoded by a modeling expert and incorporated in a model library for later use. Modeling on the flowsheet level is either supported by a modeling language (e.g. AspenTech, 1988, 1991a) or by a graphical editor (e.g. Bär and Zeitz, 1990; AspenTech, 1991b). In both cases, the end user selects the models from the library, provides the model parameters and connects them to the plant model. The incorporated chemical engineering knowledge as well as the model structure are largely fixed and not accessible. A common exception are physical property models which can be selected independently of the process unit model.

Equation-oriented modeling tools support the implementation of unit models and their incorporation in a model library by means of declarative modeling languages (i.e. the SPEEDUP language, AspenTech, 1991a) or by providing a set of subroutine templates to be complete directly in a procedural programming language [i.e. FORTRAN as in ASPEN PLUS, (AspenTech, 1988) or DIVA (Kröner *et al.*, 1990)]. There are no different tools for the modeling expert or for the end user. Hence, modeling on the unit level requires profound knowledge in such diverse areas as chemical engineering, modeling and simulation, numerical mathematics, and computer science. The development of novel process unit models is therefore often restricted to a small group of experts.

2.1. *Evaluation*

Despite the considerable advances in the last decade, where steady-state flowsheeting with modular process simulators became routinely employed by a large community of process engineers, there is considerable incentive to extend the range of model-based applications by improving the handling of models and by increasing the level of detail of process representations. Many process engineering case studies—not only in academia but especially in the research and development laboratories of the chemical process industries—have shown the potential of employing non-standard models such as dynamic models, very detailed models of standard equipment, or models of non-standard equipment.

The modular approach to modeling and simulation, though powerful and easily accessible to many engineers for the solution of standard flowsheet problems, does not adequately support the solution of more involved problems. This is largely due to the lack of precoded models for many unit operations of adequate level of detail. Examples include multiphase reactors, membrane processes, polymer reactors and most units involving particulates. Therefore, costly and time-consuming model development for a particular unit is often required during project work. Equation-oriented languages support the implementation of the models to a large extent, but they do not assist the user in developing models from using engineering concepts, nor is there support for the documentation of the modeling process during the lifecycle of a process or for proper design and documentation of the model library. Reuse of validated unit models by a group of simulator users is therefore almost impossible and redundant modeling is unavoidable. The consistency and soundness of an initially even well-designed model library is inevitably getting lost over time.

2.2. *Some novel approaches*

This experience has triggered considerable effort in recent years in several research groups. All these attempts aim at facilitating model development and maintenance by enhancing the capabilities for model formulation, reuse and adaptation as well as for maintenance and documentation. Ideally, the support should be extended to all phases of modeling including the abstraction of the real process, the development of models from first principles and the symbolic manipulation of the model equations prior to numerical analysis. Some of the recent developments are depicted in Table 1.‡ Common to all approaches are a multi-level modularization of process models and a declarative (in the sense of explicit and symbolical) rather than a procedural (in the

† This distinction does only refer to the modeling tool. In both cases, either a sequential-modular or a simultaneous solution algorithm may be employed (Marquardt, 1991).

‡ The classification of a system as knowledge-based is a delicate problem. Here, according to Newell (1982), this term has been assigned only to those systems which not only incorporate explicit and declarative knowledge (in the sense of symbol structures) to represent process descriptions but also knowledge on the processing of these descriptions such as independent reasoning tools.

Table 1. Some recent developments of advanced modeling tools

Name the system	Some references	General modeling language	Process modeling language	Knowledge based system	Integrated with process simulator
ASCEND	Piela (1989), Piela *et al.* (1991, 1992), Westerberg *et al.* (1994)	*			*
DYMOLA	Elmqvist (1978), Elmqvist *et al.* (1993) Cellier and Elmqvist (1993)	*			
DYLAN	Lund (1992)	*			*
gPROMS	Barton (1992), Barton and Pantelides (1994) Oh and Pantelides (1995)	*			*
HPT	Woods (1993)		*	*	
MODASS	Sørlie (1990)		*	*	
MODEL.LA	Stephanopoulos *et al.* (1990)		*	*	
MODELLER	Preisig (1991, 1992, 1994a, b)		*		
MODEX	Åsbjørnsen *et al.* (1989), Meyssami and Åsbjørnsen (1989)		*	*	
OMOLA	Nilsson (1993), Andersson)1994), Mattson and Andersson (1994)	*			*
PROFIT	Telnes (1992)		*	*	
VEDA	Marquardt (1992a,b), Marquardt *et al.* (1993), Bogusch and Marquardt (1995)		*	*	*
—	Perkins *et al.* (1994)		*		

sense of implicit and algorithmic) representation. The developments can roughly be classified in four groups.

(a) General modeling languages. DYMOLA, OMOLA or the modeling languages of ASCEND or gPROMS are typical examples to characterize this group. These languages can be viewed as further developments of equation-oriented simulation languages which may be traced back to the CSSL specification of the late 1960s (Augustin *et al.*, 1967). They are designed to support hierarchical decomposition of complex models in order to facilitate reuse and modification of existing models. All of them use concepts from semantic data modeling (King and Hull, 1987) and object-oriented programming (Stefik and Bobrow, 1986) with structured representations of encapsulated submodels organized in inheritance and aggregation hierarchies. These languages are not restricted to chemical engineering applications, since the language definition is confined to a relatively small number of generic elements.

(b) Process modeling languages. The fundamental ideas of these languages are similar to the generic modeling languages. However, from the very beginning they are designed to match the specific issues of a particular application domain in the language definition. Typical examples are MODEL.LA or VEDA where elements tailored to chemical engineering applications are included in the language definition. It is noted here, that VEDA is merely a language definition. Implementations are currently carried out using different software platforms. They are the frame-based knowledge representation language FRAMETALK (Rathke, 1993) at Stuttgart University as well as the expert system shell G2 (Gensym, 1995) and the process-centred design environment PROART/CE (Jarke and Marquardt, 1995) at DWTH Hochen.

(c) Modeling expert systems. The goal of these modeling environments is to ideally produce an adequate process model from a formal description of the modeling problem initially provided by a user with a minimum (or rather no) further interaction. As any expert system (e.g. Hayes-Roth *et al.*, 1983), it must consist of a knowledge base built on some hybrid knowledge representation formalism, a knowledge acquisition interface, an explanation facility as well as a separate reasoning (or inferencing) system which more or less automatically generates the model from a specification. MODEX has been a first attempt to create such a system. Also MODASS shows some signs of this general idea. After implementation and evaluation of a prototype both projects have been suspended. More recent developments drawing on expert systems ideas have been explored in PROFIT. Here, a detailed specification of the structural as well as the phenomenological characteristics completely defining an abstraction of the process under consideration is provided by the user. Based on these facts a rule-based inference engine automatically determines a set of balance equations.

(d) Interactive (knowledge-based) modeling environments. In contrast to autonomously acting expert systems knowledge-based design environments or construction kits (e.g. Fischer and Lemke, 1988) support the combination of elementary building blocks to form an artifact. Since there are many distinct building blocks with a few restrictions confining possible combination, a very large number of valid configurations can be achieved. The core of

such architecture is an interactive direct manipulative user interface which incorporates the modeler into the problem solving process. The system offers solution steps to be approved or rejected by a user rather than automatically solving the problem. Typically, the specification evolves together with the solution. Literally, there is no system as yet complying with this idea. Some characteristics, however, may be found in MODASS or in the knowledge-based user interface of DIVA (Bär and Zeitz, 1990).

Some of the modeling tools are integrated in computer-aided engineering environments to support modeling, analysis and eventually also synthesis of the process and/or its associated control system. To name just a few examples, OMOLA and the simulator OMSIM are integrated in a control system design environment (Andersson, 1994), MODEL.LA is part of DESIGN-KIT, a computer-aided process engineering environment (Stephanopoulos *et al.*, 1987), an implementation of VEDA is currently integrated in the DIVA simulation environment (Räumschüssel *et al.*, 1993), the ASCEND modeling language is an integral part of an interactive modeling and simulation environment (Piela *et al.*, 1991, 1992). Though all of these environments comprise a modular structure, neither of these environments is open in the sense of Barker *et al.* (1993).

At a first glance, all the novel approaches show obvious similarities. An observer may easily get the impression that—despite the claims of the researchers involved—there are no significant differences and all the tools are based on more or less the same ideas and techniques. However, even subtle conceptual differences eventually may result in tools with (widely) varying capabilities. In the remainder of this contribution we will therefore first try to unify the concepts, used in the various attempts towards advanced process modeling tools, to the extent possible. At the same time important conceptual differences with significant implications on the characteristics of the ultimate tool are pointed out. Such a stocktaking may not only be valuable for potential users of novel modeling tools in the future. Rather, a critical comparison and evaluation of the results achieved during an explorative phase of research as currently taking place is hopefully also helpful to researchers active in the area to focus and direct their future effort.

3. MODELING METHODOLOGY

The development of any software tool to support an engineering activity requires conceptualization (or conceptual modeling) of the problem domain. In the context considered here such an abstraction must finally result in a sound methodology of process modeling which is well-suited for computer implementation. Such a methodology comprises: (i) the *decomposition of models and the definition of elementary modeling objects* which can be aggregated to form a consistent model of (ideally) any chemical process; and (ii) generic modeling procedures which support both, the derivation of models from scratch, and the reuse and evolutionary modification of an existing model to meet the requirements of a new context.

Since any model is an artifact comparable to the technical device itself, process modeling can be viewed as some kind of design activity (Marquardt, 1992a; Westerberg *et al.*, 1994). Hence, the concepts of design theory (e.g. Simon, 1981; Van Gigch, 1991) or general systems problem solving (e.g. Klir, 1985) may serve as a useful starting point for the development of such a methodology. The following sections summarize a (not necessarily complete) methodology of process modeling. Its purpose is not only to present our view of a modeling methodology but to contribute to a unification and generalization of the recently published concepts.

3.1. Canonical modeling objects

The natural structure in a process and its associated model can be exploited for the definition of a set of canonical modeling objects to facilitate and effectively support model development.

Exemplarily, Fig. 1 shows a possible structuring of a chemical plant into delimited material entities which can be represented by a submodel. The plant and its associated model decomposed in plant sections, the individual process units such as the catalytic fixed-bed reactor which itself can be decomposed non-uniquely but driven by the modeling purpose in building blocks displaying increasing level of detail. Every entity introduced is characterized by a number of attributes to describe its nature. These attributes form the basis for the formulation of the model equations which are associated with every model entity. As the process itself, every individual equation can be refined hierarchically (Fig. 2) to increase the level of detail of the description. Process quantities are resolved by constitutive equations until the set of equations is determined after specification of technically meaningful degrees of freedom.

This exposition reveals that process unit oriented as well as equation oriented model representations only build on a fraction of the naturally occurring concepts. A more refined conceptualization of processes and its models requires modeling objects

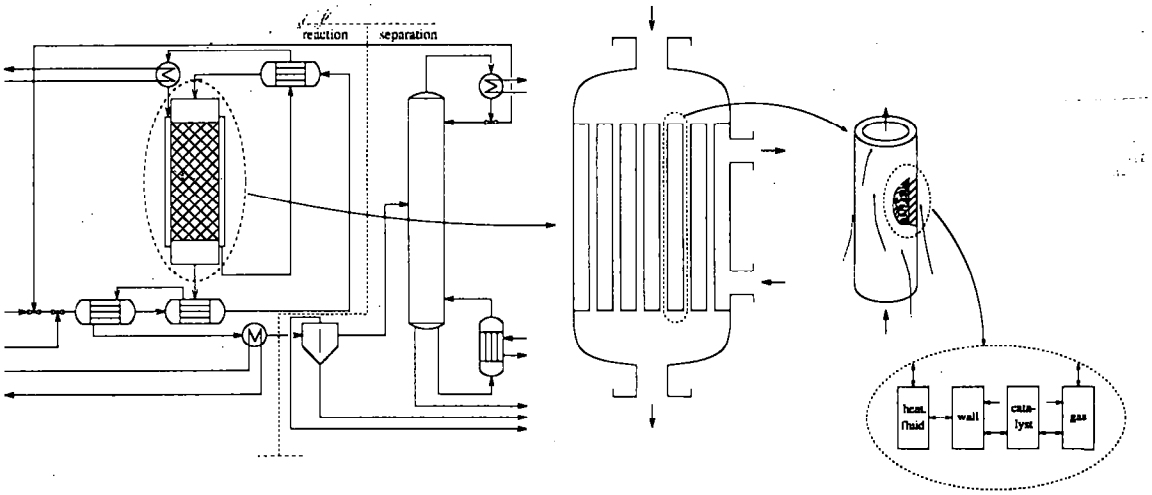

Fig. 1. A chemical plant and a possible decomposition.

capturing all levels of granularity as depicted in Figs 1 and 2. At a first glance the uniform treatment of material entities (Fig. 1) and abstract concepts (Fig. 2) may appear confusing. However, in the setting of general systems theory an object is just "any part of the world" which "can be either material or abstract" [according to Klir (1985, pp. 33/34)]. Presuming a complete set of modeling objects—we comment on this seemingly impossible assumption below—the advantages of both, the flexibility of nowadays equation-oriented modeling tools and the ease-of-use of current block-oriented modeling tools could be combined: a complex model of (ideally) any chemical process may be configured just by selection, parameterization and aggregation of modeling objects taken from a model library.

Since the decomposition of a process and its associated mathematical model is not unique there are also many degrees of freedom in the definition of suitable modeling objects. Though some general guidelines (Marquardt, 1992a, b) may be stated, the definition of modeling objects seems largely determined by pragmatism and by the experience and taste of the designer. However, it seems to be useful to build process model structuring on the general principles of scientific ontology (e.g. Bunge, 1977, 1979) and of general systems theory (e.g. Klir, 1985) though a proof of the usefulness of a structuring scheme can only be shown experimentally by the implementation and evaluation of modeling tools. The structuring is outlined briefly in the following [see Marquardt (1995) for a more detailed exposition].

3.2. *Structure description*

Motivated by the structure of a chemical process, two conceptually different classes of modeling objects, namely *devices* and *connections*, are distinguished for structure description (Fig. 3). Devices represent a delimitable part of a process such as the tubular reactor or the wall of a single tube in the tube bundle (see Fig. 1) whereas connections depict

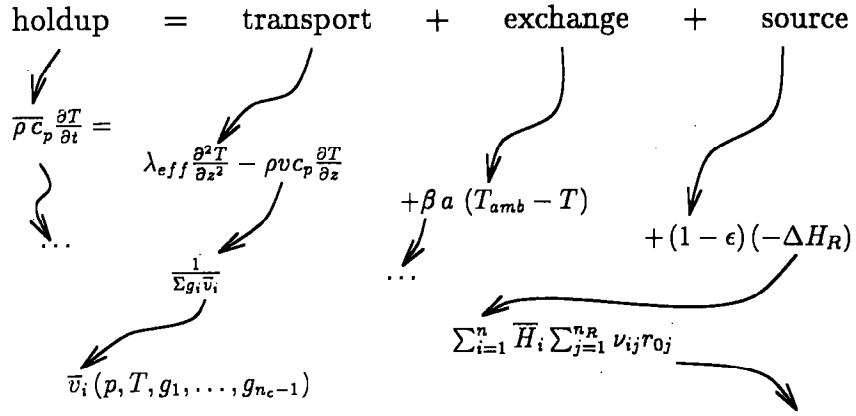

Fig. 2. The energy balance of the reactor tube and a possible decomposition.

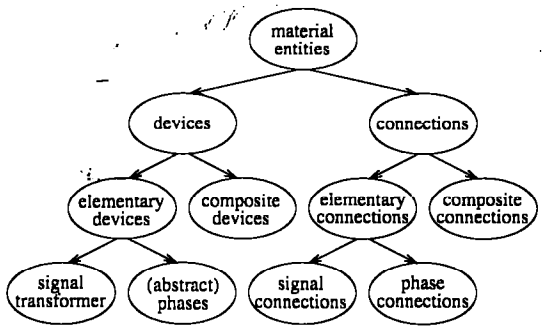

Fig. 3. Devices and connections and their specializations.

those parts of the real process which are considered to connect devices. Typical examples are the pipes between the tubular reactor and the heat exchangers or the solid–fluid phase boundary between the gas and the catalyst phase of the reactor tubes in Fig. 1.

The major conceptual distinction between devices and connections is their role in a real process (Fig. 4). The role of a device is the determination of some vector of characterizing state variables ($\mathbf{x}$) like pressure, temperature, concentrations, etc. from known fluxes ($\boldsymbol{\phi}$) like mass, energy or momentum flows from the surroundings of the device. Hence, the device responds to flux information by providing state information. In contrast, the role of a connection is the transformation of a driving force (e.g. a difference in some potential) determined by the known states of two adjacent devices into a flux. Complementary to a device, a connection responds to state information with flux information. Consequently, in dynamic modeling only devices but not connections may display a holdup for extensive quantities. To illustrate the concept let us look at a process pipe, which, according to the modeling context, may be either abstracted as a connection or as a device depending on the relevance of the pipe dynamics.

Devices and connections may be termed *elementary* if they are not further decomposed in a certain context (i.e. the catalyst phase of Fig. 1). The elementary device *generalized phase* is any delimitable—but not necessarily homogeneous—non-decomposable material entity in a process. A prototype is the fluid in a reaction vessel. Its definition extends the concept of a phase as introduced in axiomatic thermodynamics by additional characterizing attributes such as geometry or (possibly multiphase) aggregate state (Marquardt, 1995). In the case of the tubular reactor in Fig. 1, the generalized phases are for example given by the tube wall, the reacting gas, the catalyst, or the heating fluid surrounding the tube.

In contrast to the physically motivated generalized phases, *signal transformers* are motivated by the control systems in a chemical process. Consequently, signal transformers do not depict the physicochemical details of some device but represent only its input/output behaviour. Prototypes are a thermocouple of a PI-controller.

The prototype of the elementary *phase connection* is the boundary between phases of different aggregate state whereas the prototype of the elementary *signal connection* is an electrical or a pneumatical transmission line of a process control system. The definition of phase connections is largely based on thermodynamics of irreversible processes (Haase, 1963). According to this theory, major attributes of a phase connection comprise the fluxes of extensive quantities the connection can transmit as well as the generalized forces determined by the state variables in the adjacent phases causing the fluxes (Marquardt, 1995). Some examples of elementary phase connections are shown in Fig. 5. The transport through *film connections* is dominated by conduction and diffusion processes whereas *valve connections* are generalizing convectively dominated transport. A pipeline is a specialization of a *permeable valve*, a selective porous membrane refines a *semipermeable valve* and an *impermeable valve* typifies a solid wall.

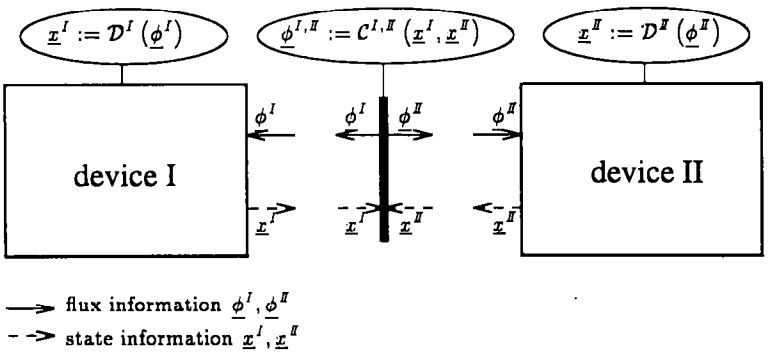

Fig. 4. Information flow between devices and connections.

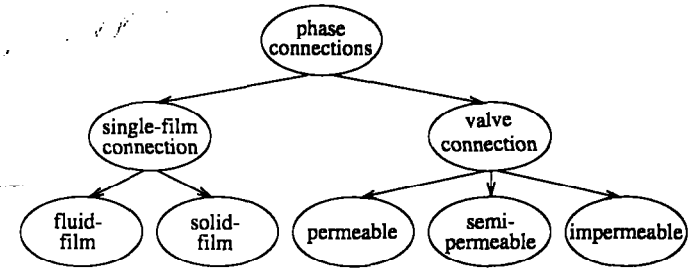

Fig. 5. A taxonomy of some elementary phase connections.

Signal connections transfer selected process quantities from a device to a signal transformer. In contrast to phase connections (see Fig. 5) a more abstract setting is employed, since any signal connection is completely defined by the directionality and the process quantities transferred regardless of the acting generalized forces (e.g. a voltage difference).

Composite devices and connections are aggregations of elementary or composite devices and connections, to form structured models of process (sub-)units and control (sub-)systems. For example, a subsystem of the reactor in Fig. 1 composed of the tube wall and the boundary layers on the shell and tube side may be termed a composite connection whereas the reactor tube which comprises the wall, the tubeside film, the gas phase, the catalyst phase boundary layer and the catalyst phase itself can be viewed as a composite device. The distinction between devices may seem arbitrary. They are distinguished, however, by their purpose and by the convention on cause and effect as depicted in Fig. 4. Further, the distinction is necessary in order to support arbitrary refinement or lumping of both, devices and connections.

3.3. *Behavior description*

The structural description of a complex system—given by its elements and their interactions—needs to be complemented by a behavioral description of every elementary subsystem in order to fix the behavior of the whole system.

The behavior of a modeling object is reflected by the values of the assigned process quantities. Figure 6 shows part of a possible taxonomy built on physical arguments. There are three major groups to be distinguished for a behavioral description of a phase: *generalized fluxes, thermodynamic states and state functions* as well as *phenomenological coefficients*. Generalized fluxes include the variation of the holdup, the transport in a phase and across a phase boundary as well as sources caused by chemical reaction or some external potential field. Obviously, the fluxes are depending on the thermodynamic state functions and on phenomenological

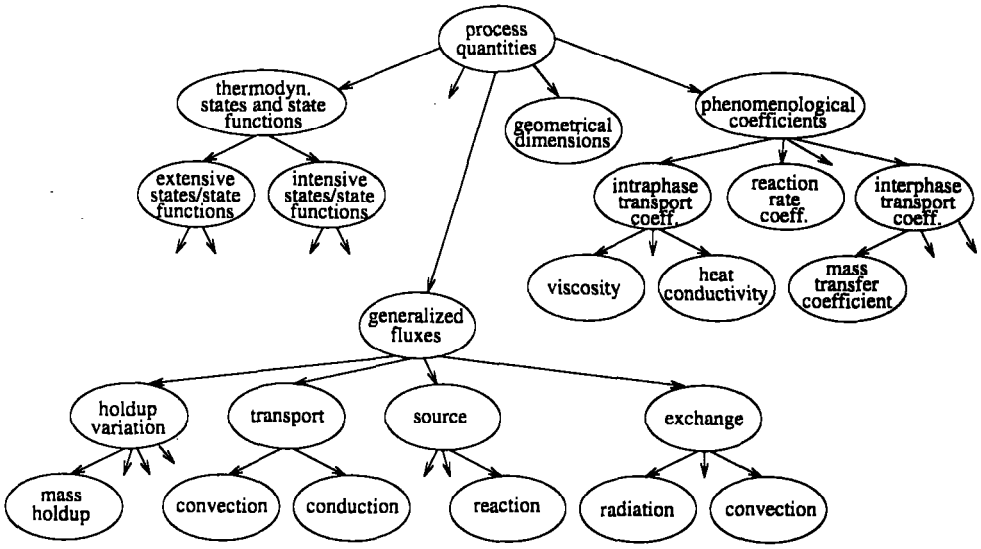

Fig. 6. A taxonomy of process quantities.

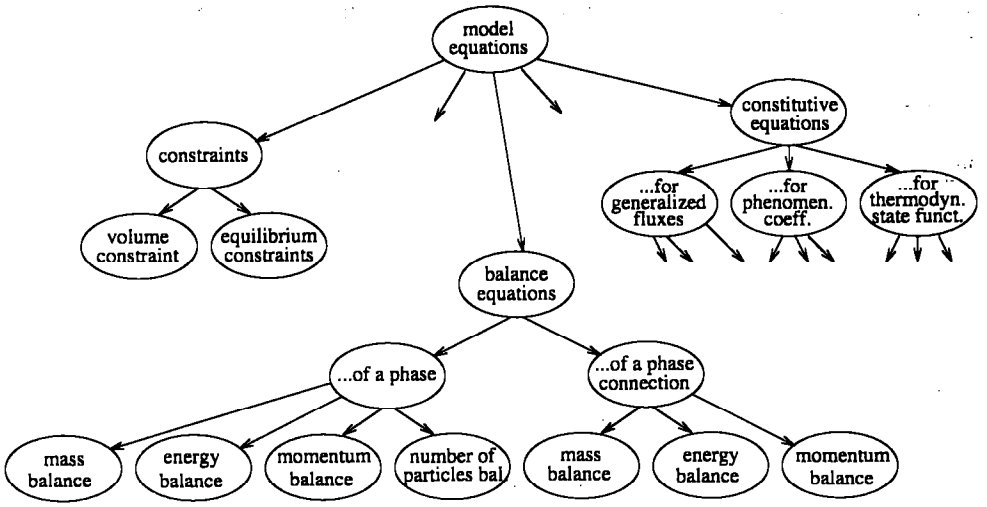

Fig. 7. A taxonomy of model equations.

coefficients in a complicated manner. All the generalized fluxes and their refinements directly correspond to the physicochemical phenomena occurring in a phase or at its boundary according to the phase description. Any process quantity assigned to a particular phase may depend on one or several coordinates such as time and spatial dimensions.

The values of the process quantities are restricted by "laws" of various kinds. These laws may either represent fundamental or empirical physicochemical relationships (a white-box model) or experimentally identified equations (a black-box model). In contrast to black box models [see for example Ljung (1987) or Pearson (1994)] which are used to fix the behavior of signal transformers, white-box models are used to represent generalized phases and elementary phase connections. There, the model equations reveal all the characterizing attributes given in the structure description. A part of a taxonomy of the model equations based on physicochemical laws is presented in Fig. 6. They include *balance equations, constitutive equations* and *constraints*.

Balance equations involving extensive quantities are forming the upper level of a model equation hierarchy in the behavior description of phases and phase connections. In general, balances for total mass, for the mass of all but one species occurring in the mixture, for momentum, for total energy and for the particle number in case of particulate systems (e.g. Ramkrishna, 1985) is sufficient. For convenience, balances of kinetic and internal energy should be added. Balances for any other quantity of interest can easily be derived by simple symbolical manipulation. Though, there is a large variety of balance equations some unification as suggested by Haase (1963) or Müller (1973) is possible in order to better structure and condense this knowledge (Gerstlauer *et al.*, 1993).

As illustrated in Fig. 2, balance equations are refined on the next hierarchical level by constitutive equations which describe generalized fluxes (see Fig. 6). The thermodynamic state functions, the transport coefficients and the reaction rates need to be further refined to finally lead to expressions in the primary process quantities required for monitoring the state of the phase or the phase connection and a number of additional auxiliary variables.

Finally constraints need to be added to complete the set of equations. Besides closure equations (to sum, for example, concentrations to one) constraints always occur in the models of a phase connection. There, they relate the state variables. An illustrative example is given by a vapor–liquid (non-equilibrium) two-film model, where the liquid and the vapour concentrations at the interface are constrained by the equilibrium relations. Constraints may also be formulated for composite devices to state either a geometrical or some thermal, chemical, or mechanical equilibrium restriction between phases. As an example, a two-phase model of a distillation tray may be considered (Ponton and Gawthrop, 1991): the volume of both phases is constrained by the volume of the tray; some equilibrium condition, such as thermal equilibrium, may be stated to simplify the model.†

Process quantities and equations can be thought of being conceptually organized in an *and/or-graph* with two types of nodes, one corresponding to a process quantity and the other to an equation. The

† The unavoidable increase of the index of the differential-algebraic system must be treated by symbolical manipulation during preprocessing (see below).

edges of the graph are interpreted either as "and" if they point from an equation to an occurring process quantities, or, as an "or" if they point from a process quantity to the various possible equations which may be used to compute its value [see Marquardt (1992a) and Bogusch and Marquardt (1995) for details].

3.4. *Experiment description*

Models stored in some model base should serve different kinds of experiments such as simulation of the steady-state or dynamic behavior, design calculations or optimization to fix operational or constructive degrees of freedom, or estimation of model parameters from experimental data (see Fig. 8).

The major difference between these modes of application is the interpretation of the role of the process quantities involved in the model. Therefore, the experiment description first of all consists of a classification of *all* model variables (but physicochemical constants and parameters describing the properties of a mixture) as either *computed variables* or as *design variables*. Computed variables are determined from the model equations employing user-provided or default initial values, whereas design variables are specified by the user.

In an experiment involving a dynamic model design variables may either be time-invariant *parameters* or time-variant *forcing functions* which typically model the external actions the process (and hence the model in an experiment) is exposed to. If the operation of a batch process or the start-up of a continuous process is imagined as a typical example, forcing functions may be quite complex. They have to represent many sequential or parallel tasks triggered by conditional statements either driven by the elapsed process time or by the actual value of some process variable. As in the case of the model description where a structural decomposition is suggested to reduce complexity, a procedural decomposition of complex action applied to some model during an experiment in *composite* and *elementary* tasks (or functions) is appropriate (Barton, 1992; Andersson, 1994; Barton and Pantelides, 1994). Instead of purely language-oriented definitions of tasks as typically used in general modeling languages (e.g. Elmqvist *et al.*, 1993; Barton and Pantelides, 1994), a higher level task definition analogous to the structural and behavioral modeling objects above seems to be advantageous. Further investigations are required to strengthen this conjecture.

In case of estimation or optimization problems an objective function must be formulated in addition to the constraining model equations. In many situations (especially in mixed structural and operating point optimization), the formulation of an adequate cost function is a difficult modeling problem itself. This particular facet of process modeling has not yet received the attention it deserves.

3.5. *Generic modeling procedures*

The former sections are primarily dealing with *knowledge about models*. As already stated a modeling methodology additionally has to comprise generic procedures, or, *knowledge about the modeling process*. Though there are excellent expositions of how to derive and analyze a mathematical model in general (i.e. Aris, 1978; Denn, 1985), the procedural aspect of modeling has not yet been treated satisfactorily in the context of computer-aided modeling.

General modeling flow diagrams like the one depicted in Fig. 9 and commented on subsequently are quite common. Such task sequences are however of a too coarse granularity in order to guide the modeling process in adequate detail. In order to understand the complex modeling process one may decompose every task into *elementary canonical modeling steps* as the process model has been decomposed into modeling objects. A modeling step which typically is associated with a modeling object displays distinctive properties and relationships. Properties of a modeling step are, for example, the assumptions involved or the results produced by its

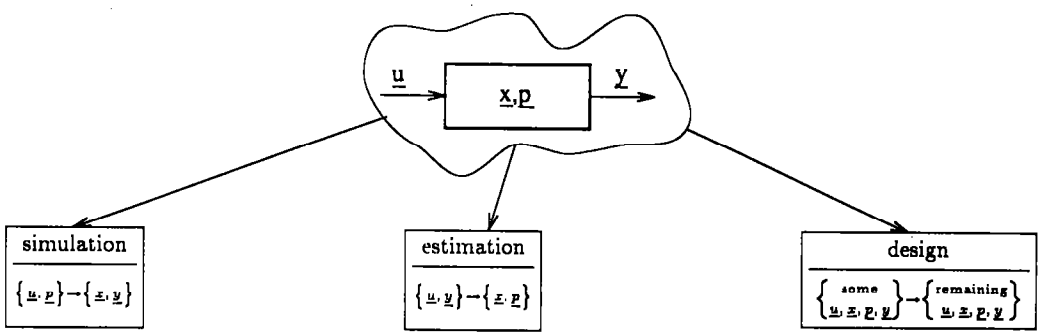

Fig. 8. Different classes of experiments applied to a model.

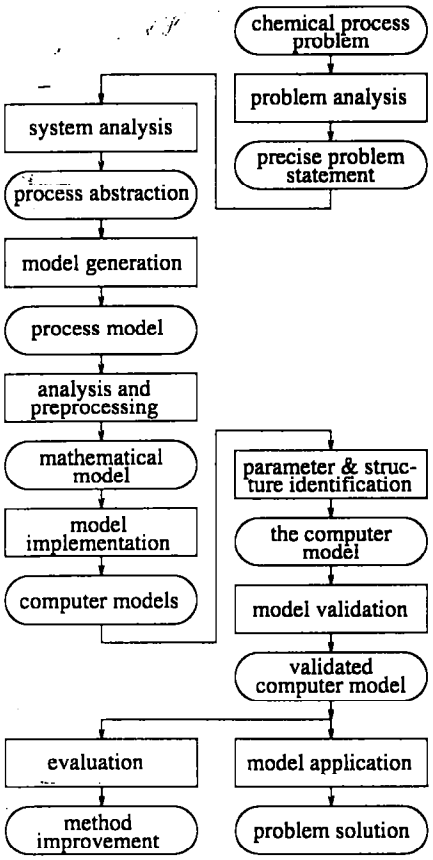

Fig. 9. Flow diagram of process modeling.

application. A relationship between modeling steps may be given by the sequence they can be invoked. Modeling steps can be aggregated to complex modeling procedures to be reused in different contexts. Certainly, there is no single sequence of modeling steps in the sense of a rigid algorithm leading to a certain process model. Rather, depending on the experience and style of the modeler, the modeling steps may be carried out in many valid sequences complying with the relational constraints given. The derivation of canonical modeling steps is under development at the moment in the author's research group (Lohmann and Marquardt, 1996).

The first coarse modeling task as displayed in Fig. 9 is proper *problem analysis* to develop a precise formulation of the problem. Requirements on the model, on the quantities determined by the model, or on the experiments to be carried out with the model can usually be stated only in a very crude manner at this stage.

Mathematical model development is accomplished by three distinct but highly intertwined conceptual tasks with many iteration loops not shown in Fig. 9 [see Marquardt (1995) for more details]. First, *system analysis* (or abstraction) leads to an informal description of the process. This verbal or graphical problem description needs to be refined step by step by supplying more detailed information. Model structuring should serve as a guideline, since the specification of the modeling objects, their behavior and their aggregation must result in a complete process description. Its information content must suffice for the *generation* of a symbolical *process model* which may comprise of various equation types and include discontinuities to reflect the discrete–continuous nature of chemical processes (Marquardt, 1991; Barton and Pantelides, 1994). After *analysis* and *symbolical preprocessing* the process model is converted into a (sometimes approximate) *mathematical model* being eventually interfaced to some numerical algorithm after *model implementation*.

The target of *symbolical preprocessing* is determined by the type of experiment (e.g. simulation or optimization), the numerical techniques chosen and the type of mathematical model required as well as the computer hardware envisioned. Besides a transformation of the original model equations into some standard form required by the numerical algorithm (i.e. a linear-implicit differential-algebraic system of index one in dynamic simulation), symbolical preprocessing can also be employed to analyze feasibility or improve efficiency and robustness of numerical analysis.

A first example is given by the analysis of structural defects revealing specifications of the degrees of freedom which either lead to infeasible or high index problems. As suggested by Marquardt (1992a) a degrees of freedom analysis can be carried out on the level of each modeling object (Westerberg *et al.*, 1994) to complement the results of a purely structural technique (e.g. Westerberg *et al.*, 1979) by physical insight. The local effects of a certain specification should support the search for a feasible specification for the complete process model. If high index problems occur, proper analysis can reveal whether and how the index can be influenced by alternative modeling and/or choice of design specifications (Lefkopoulos and Stadtherr, 1993; Unger *et al.*, 1995). Further tasks of symbolical preprocessing are the partitioning of the equation system, the derivation of the Jacobian, or the replacement of numerically ill-conditioned expressions (such as square-root power laws in reaction kinetic expressions at low concentrations) by a well-behaved approximation. The tight integration of symbolical and numerical computing is still a fruitful research area in order to contribute to more robust and efficient solution algorithms. An additional dimension of complexity is introduced if a mapping of the

mathematical model on a multi instead of a single processor computer is envisioned in the future.

The final modeling tasks are the *discrimination* of competing model structures and the *identification* of unknown model parameters as well as the *validation* of the model. Especially in the case of nonlinear models, all these activities are not sufficiently well understood and supported by adequate methodologies.

3.6. Discussion

The definition of canonical modeling objects should aim at a complete framework. Completeness is impossible to achieve, however, regarding the continuously increasing variety of process units and theories to describe physicochemical phenomena. Instead, an extensible classification scheme for the knowledge about process models based on the concepts defined above should rather be introduced. Such a scheme is required to support (or even permit) continuous and consistent enlargement of a properly structured and well-documented model base, either by collecting and classifying all the modeling knowledge in a certain area of chemical engineering, or, by introducing novel concepts step-by-step as they are worked out during regular project work. This way, the significant investment and the accumulating expertise in a modeling team can be conserved over a longer period of time.

The concept of model decomposition suggested definitely reduces complexity and seems to be the only means to effectively support modeling. However, it has also a serious drawback which is related to model validation. Typically, only a complete model can be validated for a particular or at best a certain class of applications. Even if (validated) submodels of varying granularity are provided in a model library it is very difficult to predict the validity of any of the numerous possible aggregated models in general. The same problem, as pointed out by Zeigler (1976) originally, occurs in principle, if a monolithic model description is completely separated from the experiment as suggested above. Zeigler's *experimental frame* might be extended to an *application frame* which should additionally include constraints on the aggregation of validated submodels to result in a validated process model.

The identification of all important types of modeling steps, their properties and temporal as well as logical interrelationships seems to be indispensable, especially if we are interested in knowledge-based tools which effectively guide modelers with widely varying modeling skills. This search for canonical modeling steps might provoke some readers since often the problem of modeling, as a special design problem, is viewed as ill-conditioned in the sense that neither a precise problem statement nor sound performance measures are available. This fact often led to an understanding of modeling as an art rather than a science (e.g. Aris, 1978) emphasizing a modeler's creativity and intuition rather than a scientific methodology. Such an objection is undoubtedly valid. However, there are not only explorative but many routine modeling problems to solve which could effectively be supported by some formalization of the modeling process.

3.7. Relation to previous work

As already stated, the modeling concepts presented here are not entirely new. The first approach to the definition of elementary submodels which are application specific in the sense above has been advocated by Stephanopoulos *et al.* (1990). Similar ideas have been presented in our own work (Marquardt, 1992a,b; Marquardt *et al.*, 1993; Bogusch and Marquardt, 1995) as well as in the works of Åsbjørnsen *et al.* (1989), Meyssami and Åsbjørnsen (1989), Nilsson (1993), Perkins *et al.* (1994), Piela (1989), Piela *et al.* (1991, 1992), Preisig (1991, 1992, 1994b), Sørlie (1990) and Telnes (1992). It should be pointed out, however, that the structuring scheme as presented above seems to be the most complete to date as it rigorously incorporates not only distributed parameter systems but also the different kinds of interactions between generalized phases which primarily distinguish chemical engineering applications. A classification of process quantities and equations has also been given by Åsbjørnsen *et al.* (1989), Meyssami and Åsbjørnsen (1989) and Sørlie (1990). The most concise structuring of control systems has been presented by Nilsson (1993). A thorough treatment of language based experiment descriptions has been introduced by Barton (1992).

Structuring of knowledge about the modeling process is largely missing until now. There are only modest attempts to formulate algorithms for a few isolated modeling activities. Examples are given by Gerstlauer *et al.* (1993), who treat the generation of all kinds of balance equations from a minimal set of information, by Telnes (1992), who lays out a sequence of algorithms to be used for the derivation of certain types of balance equations from a process specification, by Pfeiffer and Marquardt (1993), who present the development of an algorithm for the spatial discretization of partial differential equations, and by Preisig (1994a) who suggests an algorithmic distribution of chemical species in a network

of interconnected modeling objects. Symbolical preprocessing is employed in several modeling tools for different purposes. Quite advanced techniques are provided by DYMOLA (Cellier and Elmqvist, 1993), OMOLA (Andersson, 1994) and gPROMS (Barton and Pantelides, 1992; Oh and Pantelides, 1994).

4. FORMALIZATION OF MODELING KNOWLEDGE

The logical concepts identified during conceptual modeling must be formalized to result in some modeling language. The language can also be interpreted as a data model (or schema) from a data engineering perspective (e.g. Kim, 1990), or it can be viewed a knowledge representation scheme as developed in the AI community [see Fikes and Kehler (1985), Christaller *et al.* (1992), Rathke (1993) for some examples]. The formalism should not only serve for the implementation of specific process models in a model library but also for the representation of knowledge about models and about the modeling process in a modeling tool.

Systems concepts have been used to formalize simulation models in the past (e.g. Zeigler, 1984). Klir (1985) presents four essential subsystems in order to describe systems in general:

- *a source system* defining the variables, the set of values they can take on, and their support (i.e. the coordinates they depend on);
- *a data system* which forms a collection of values of the variables ordered along some coordinate;
- *a structure system* defined by a collection of elementary types of system components and connections as well as their instances;
- *a behavior system* which includes a set of relationships between variables of the source system (e.g. mathematical equations) restricting the set of values the variables can take on.

Obviously, all these subsystem types can be identified in the exposition above. Systems theories often employ set theory or some kind of predicate logic to define their concepts (Bunge, 1977; Klir, 1985). Such representations, being rigorous and complete, are advantageously used for theoretical analysis. Though in principle also suitable for a representation of process models [see Piela (1989) or Preisig (1991) for examples] these abstract formalisms are not recommended since they lack any transparency and explicitness. The application of such formalisms for conceptual modeling in some application domain is therefore unnecessarily complicated. Instead, object-oriented approaches seem to be preferable because they overcome the disadvantages of logic or set theoretic schemes to a large extent without losing their expressiveness (Hayes, 1979). The adequacy of such schemes for general systems representation is shown by Orchard and Tausner (1988) using set theoretic arguments. Wand (1989) also presents a formal definition of an object system based on Bunge's systems theory (Bunge, 1977, 1979).

4.1. *Object-oriented concepts*

Conceptually, any real or abstract entity (such as the fixed bed reactor in Fig. 1 or its mathematical model) is considered an *object*. Hence, any object can be referenced by a unique *identifier*. Objects have at least one *attribute* to express the entity's properties and possibly some *methods*† to operate on the *values* of the attributes. Attribute values can be objects themselves to enable the construction of semantic links between objects. Objects are encapsulated in the sense that the data structures (given by the attribute definitions) are hidden and only accessible by carefully designed means‡. All objects which share the same set of attributes and methods can be viewed as *instances* of a *class*. Any (sub-)class can be derived from another (super-)class by inheriting (all) attributes and methods. Typically, more attributes and methods are added to complete the definition of the subclass. This *inheritance* relationship allows the construction of class hierarchies. The given outline of object-orientation comprises the core concepts of object-oriented data modeling according to Kim (1990).

The organization of objects in class taxonomies with an inheritance relationship is a common but debated approach (Stein *et al.*, 1989). A popular alternative, for example, is the notion of *prototypes* and *delegation* instead of classes and inheritance. In prototype-based schemes, there is no strict distinction between a class and its instances. A new object is created by cloning and modifying an existing object.

In order to illustrate the capabilities offered by the core concepts for model representation let us again look at the example of Fig. 1. Any entity of the decomposition hierarchy is represented by an object. The attributes of the objects correspond to

† It should be noted here, that any object-oriented data model focuses on the definition (and inheritance) of methods. If the methods operating on the data of the object are not included, the schema should be called a semantic data model instead (King, 1989). This distinction will not be exaggerated subsequently.

‡ In object-oriented programming (Stefik and Bobrow, 1986) encapsulation is implemented by message passing. In modeling formalisms it is achieved by so-called terminals.

the characterizing properties of *generalized phases* and *signal transformers* as introduced above. The conceptual elements introduced for model decomposition may directly correspond to the major classes of the representational schema. Class hierarchies can be constructed by a refinement of these major classes. For example, a *generalized phase* may be refined to a *well-mixed homogeneous phase* by fixing the values of the attributes *dispersive state* and *spatial distribution* to *well-mixed* and *homogeneous* (see Fig. 10). An instance of such a class is a certain object with unique identity such as the gas phase in the 65th tube of reactor R101 in the styrene plant. Methods associated with the data not only include basic operations like editing the values of the object's attributes but also complex procedures to derive new knowledge. If, for example, the heat conduction equation to describe the behavior of the tube wall in Fig. 1 is represented by an object, an associated method could provide a symbolical spatial discretization of the partial differential equation prior to numerical solution.

In addition to the core concepts several semantic extensions are essential in order to capture the requirements of our application. The most important is the notion of composite objects (e.g. Kim, 1990) to allow an *explicit* representation of decomposition hierarchies (e.g. the fixed bed reactor and its decomposition in Fig. 1). Backed by general systems arguments composite objects should not only explicitly refer to their components (or devices, as in most object-oriented data models) but also to the connections. Further, the way connections relate connected objects needs to be specified explicitly because of the complex interactions possible. It is not sufficient, for example, to state that a particular pipe connects the reactor with a certain heat exchanger in the process of Fig. 1; a concise specification must refer to the pipe unions of both units the pipe connects in addition.

The representational formalism should also be able to explicitly deal with the multi-facetted nature of process models (Zeigler, 1976; Stephanopoulos *et al.*, 1990). Frequently, there are various modeling *alternatives* considered simultaneously which differ in the degree of detail or in the assumptions made. All of them need to be consistently integrated in a complex object. To the author's knowledge there are no particular concepts coping with this issue. The representation of alternatives should not be mixed with the notion of a *perspective* in the sense of Mariño *et al.* (1990). Perspectives provide support for multiple views of an object in different contexts. Any device such as the fixed bed reactor in Fig. 1

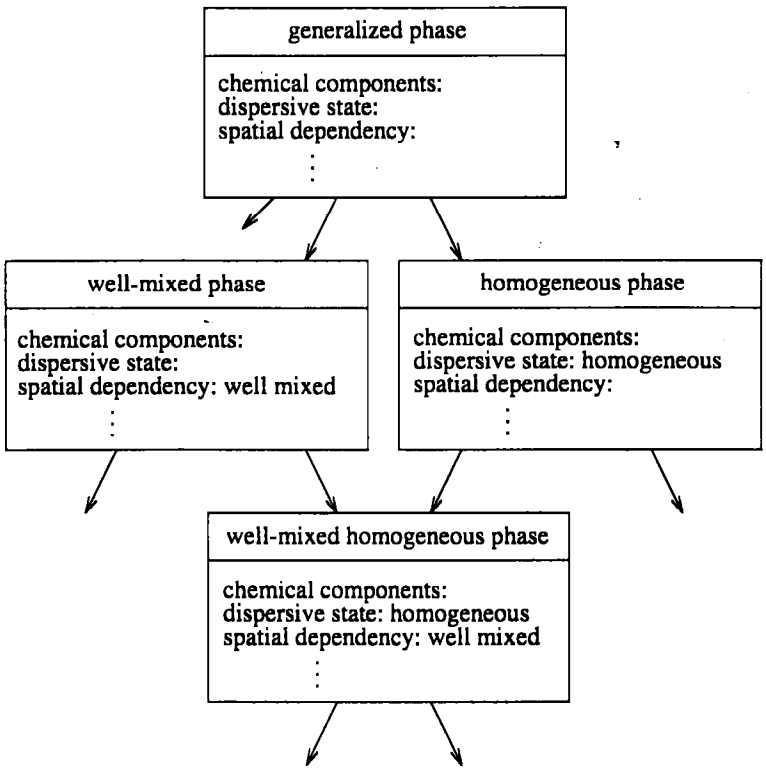

Fig. 10. A simple class hierarchy.

can either be viewed from the perspective of the composite device it is part of, i.e. the plant, or from the perspective of its own structure and behavior. The properties of the device corresponding to the attributes of an object can be partitioned into two distinct groups defining the *interface* and *implementation* perspectives of a device.

An important issue of modeling in general is the consistency or correctness of a model. Several measures can be considered. These include for example restrictions on the value of an attribute with respect to its type, its cardinality or even on the set of instances the value can take on. Such consistency constraints, though restricted in expressiveness, are quite powerful if a strongly typed object system in the sense of a class rather than a prototype is employed. Further, cross referential integrity constraints are indispensable to guarantee consistency of the data structure. For example, a composite device is characterized by the components it consists of whereas at the same time the components will contain information about the composite devices they belong to. The values of attributes stating this fact need to be kept consistent automatically. Besides these essentials, we believe that more general constraints—loosely defined as some relation between the values of two or more attributes associated with the same or different objects—need to be incorporated in the representational formalism. They not only allow a flexible formulation of any kind of integrity relation to check consistency of the data structure, but may also be used for knowledge processing such as constraint satisfaction (Kumar, 1992). Advanced knowledge processing capabilities are required if a (partially) autonomous behavior of a modeling tool is projected. More common production rules as used in rule-based expert systems (Hayes-Roth *et al.*, 1983) are forming an alternative to these efficient search techniques.

The model development process (as any design activity) is evolutionary and iterative in nature. Different versions of some models will be constructed, analysed and discarded until a satisfactory solution is found. The versions of the model or of any of its components and their temporal as well as their logical relationships should be incorporated in the representational formalism [see Katz (1990) for a comprehensive review on object versioning].

4.2. *Object-oriented model representation*

The application of object-oriented concepts to the design of a modeling language requires two major decisions. The first is related to the selection of the conceptual building blocks to represent a process model whereas the second is concerned with the choice and definition of representational elements in the framework of an object-oriented formalism. Regarding the conceptual elements either few and necessarily generic or many and therefore specific concept types are possible as extremal options. Typically, the variety of representational elements employed increases with the number of conceptual elements. In order to illustrate the different approaches some of the novel languages are investigated. Both aspects, the conceptual as well as the representational, are explicitly included in our discussion.

Some key features of the ASCEND modeling language (Piela, 1989; Piela *et al.*, 1991, 1992) as a first prototype of a general modeling language are briefly sketched. The major modeling objects of ASCEND are *model, elementary* and *atom types* as well as *relationships*. An ASCEND model is a structured type (or class) built hierarchically from instances of other models, elementary or atom types and relationships. Elementary types are primitive data types like real, string, etc. whereas atom types represent process quantities. They are characterized by a number of attributes including physical dimension, default values or lower and upper bounds. Relationships are either algebraic equations or language specific operators. The latter comprise of an inheritance relation between types, an instance-type relation and three additional operators to merge or refine instances. In particular, merging of instances of process quantities defined in different models is used to implement composition. Hence, ASCEND retains only process quantities and (general) equations from the variety of conceptual elements identified in process models. Methods are used for symbolical and numerical processing of the model equations (Westerberg *et al.*, 1994).

A second example to be illustrated in more detail is the OMOLA language (Andersson, 1994). Compared to ASCEND the major difference is the more explicit notion of composition in the language definition. OMOLA distinguishes *structured* and *primitive models*. A primitive model is a class containing attributes to denote process quantities (either classified as *variables* or *parameters*), *equations* (either differential or algebraic) and *terminals*. Terminals describe the interactions between models and hence form the interface perspective of a model. Structured models correspond to class definitions with attributes representing *submodels* (either structured or primitive), *connections* and *terminals*. As in ASCEND, the language elements are not specific to chemical engineering applications, but they nicely comply with general systems theory concepts.

Hence, from the conceptual elements introduced above, OMOLA just employs process quantities, equations as well as elementary and composite signal transformers. In contrast to ASCEND strict encapsulation is enforced in OMOLA via the interface definition: only the variables included in the terminal definition are accessible by another model and hence may be used for composition purposes. In OMOLA, the class definitions only include attributes but no methods.

Process modeling languages (see Table 1) are based on a comparably large number of semantically rich conceptual elements which are reflecting the needs of process engineering applications. MODEL.LA (Stephanopoulos *et al.*, 1990) uses most of the conceptual elements introduced above, though only a subset of the concepts seem to have been rigorously worked out and implemented in the system. For example, the language is limited to steady-state problems and does not support distributed parameter systems. The knowledge representation formalism employed is heavily influenced by semantic networks ideas.

In VEDA (Marquardt, 1992a, b; Marquardt *et al.* 1993; Bogusch and Marquardt, 1995), the development of which is still in progress, all the conceptual elements introduced for the structural and behavioral description of process models are mapped to an object-oriented representational formalism. Consequently, every entity—such as all structural modeling concepts, equations or process quantities—are represented by an object. The representational formalism includes all the core concepts of object-oriented representation (including methods) as well as some of the extensions discussed. *Devices* are represented by an object with an *interface* and in *implementation* perspective. The attributes of an elementary device (or connection) are not only explicitly stating the process quantities and equations involved (as for example in ASCEND or OMOLA) but all important properties as listed above. Though, supporting interface definitions as in OMOLA, no strict encapsulation is enforced in VEDA. The complete data structure rather than only interface data of a device is visible. Interfaces provide default access to the data describing a device according to the physical interaction of the device with its environment whereas direct access to any data may be used for parameterization of the device or monitoring of the device's state. *Composite devices* are explicitly modeled by their parts, i.e. devices and connections, and by a logical relation precisely describing component interaction (Marquardt *et al.*, 1993). The concepts for the representation of the model structure are analogously applied to the representation of process quantities and their restrictions by means of *variables* and *equations* (Bogusch and Marquardt, 1995). Variables reveal an interface and an implementation perspective. The interface refers to information on the use of the variable in an equation such as its name, dimension, support etc. whereas the implementation refers to an interpretation of the variable in a certain experiment (parameter, forcing function or computed), to the variable's value and/or to a set of equations which may be used to compute this value. Hence the logical relation of equations and variables in an and/or-graph has been implemented in the data structure explicitly. Equations are objects expressing (mathematical) relations between variables. Modeling alternatives are represented in VEDA in an integrated manner just using the core concepts. Methods are used to incorporate not only declarative but also procedural modeling knowledge [see Gerstlauer *et al.* (1993) for an example]. Integrity of the data structure is enforced by several measures ranging from attribute type restrictions to general constraints added to the object definition. General constraints, version control, the notion of a perspective (currently implemented by means of core concepts) and the integration of rules are projected.

4.3. *Discussion*

The significant differences between general and process modeling languages merits an in depth discussion in this summarizing section.

The small number of general modeling language elements makes them easy to use in a variety of application areas and limits at the same time the effort in tool development. Depending on their design these languages support certain types of mathematical equations (such as algebraic equations in ASCEND or partial differential-algebraic equations in gPROMS). Assistance is provided for model implementation, preprocessing and debugging. No support is given for other modeling tasks shown in Fig. 9 such as model development from first principles. Proper documentation and process model structuring, being essential for the design of meaningful model libraries, are largely left with the responsibility of the modeler as in established modeling tools.

As a simple example of a heat-exchanger discussed by Mattson *et al.* (1994) shows, a trade-off between lean code and maximal transparency to facilitate reuse is difficult to achieve with general modeling languages. As a consequence, users must be trained in object-oriented design to take full advantage of these novel developments.

Uneducated or even unreflected use of these languages inevitably leads to model representations which are almost impossible to comprehend and to reuse. Encapsulation of objects, a valid technique to hide complexity, does not sufficiently support model reuse, since complying interfaces are only a necessary condition for reuse. More often, the implementation of a model decides on possible reusability in a new context. Satisfactory transparency of all details of the process model as well as of the modeling process is an essential requirement for model reuse.

By definition, process modeling languages must include object-oriented design of the model base schema capturing both, the model as well as the development process. The burden to come up with a satisfactory solution is thus taken from the modeler and left with the designer of the modeling tool. Even more important, the designer is able to enforce a certain way of model structuring and documentation by the class definitions contributing to more transparent model libraries. The constraints imposed by the tool, however, should balance the striving for some kind of standardization and the need for flexibility not to impede creativity. A striking advantage of process modeling languages are the high-level modeling concepts referring to the structure and phenomena in a chemical process. These concepts correspond to engineering thinking and therefore facilitate conceptual modeling including model verification and modification on an appropriate level of abstraction.

A semantically rich representation schema certainly offers far-reaching possibilities regarding the support of the modeling process since specifically designed procedures may directly exploit the declarative knowledge about the model and the procedural knowledge about the modeling process in various ways. The knowledge can be used to enforce consistency, to critique the modeling decisions of a user, to generate a detailed documentation of some modeling object or of a complete process model, to retrieve a model from the library or to abstract a model only partially meeting the requirements of a new context prior to an appropriate refinement.

Typically, the language definition comprises only a limited number of major classes but a large number of refinements. As in object-oriented programming in SMALLTALK (Goldberg, 1983), a user has to spend a long time to learn about all the predefined classes available. Only sophisticated user interfaces which effectively assist browsing of the library of modeling objects and the retrieval of fully or partially matching objects may solve this problem. Hence, not only the design and implementation of the variety of tailored classes but also the development of appropriate user interfaces contributes to a tremendously increased effort for tool development.

Our vote for application specific modeling tools fully corresponds to a general trend in data engineering where more and more application specific extensible schemas are migrating into database languages [see Lockemann *et al.* (1993) for supporting arguments and an example]. In the longer run, such a concept should have a greater potential to disseminate modeling and simulation technology to a wider group of chemical engineers instead of keeping the technology with the experts (provided that appropriate human-tool interfaces become available). Certainly, general modeling languages may be viewed as a first step towards more sophisticated process modeling languages [see Nilsson (1993), Westerberg *et al.* (1994) for recent attempts]. However, it is expected that the language kernel needs to be extended to meet the particular requirements of the area of application as worked out by Nilsson (1993, 1995) during the development of a process model library in OMOLA.

4.4. *Software for model representation*

In the novel modeling tools the implementation of the object-oriented formalism for model representation is accomplished by means of either an object-oriented programming language such as C++ (Stroustrup, 1986) or CLOS (Steele, 1990), or, using a hybrid knowledge representation formalism like KEE (Fikes and Kehler, 1985), in G2 (Gensym, 1995) or in FRAMETALK (Rathke, 1993). These languages provide (at least) some of the concepts used to construct the model representation formalisms suggested and hence to facilitate implementation. Obviously, the latter offer more powerful concepts as the former and are therefore better suited for implementation of more complex representation formalisms.

However, as pointed out before (Marquardt, 1992b), these programming languages suffer from common disadvantages hampering concurrent engineering. They can only be overcome by introducing capabilities provided by database management systems (DBMS), which include persistency, concurrency control, authorization schemes or distribution in a computer network. In order to limit data conversion efforts only object-oriented DBMS (Kim, 1990; Ahmed *et al.*, 1992) should be considered. Also, these systems support an elegant way of integrating the persistent storage of declarative as well as procedural knowledge by encapsulating data and methods in an object. A first quite promising

attempt in applying such a tool for model base implementation has been reported recently by Maffezoni *et al.* (1994) for mechatronics applications. Despite the attractivity of object-oriented DBMS from a data modeling point of view, they may show insufficient performance due to the complexity of database schema and transactions. Hybrid solutions building on data repository ideas (Bernstein and Dayal, 1994) may be more favorable.

5. CONCLUSIONS

Considerable effort has been undertaken in recent years towards advanced modeling tools to be integrated in computer-aided process engineering environments. A common paradigm is the complete decoupling of model representation and its application. That way models may be used for different purposes if a proper integration of a variety of tools in an open system architecture becomes feasible. Currently, two major lines of development may be distinguished, which are expected to provide modeling support of a different quality. Novel developments based on general modeling languages are in a quite mature state and an introduction at the market is expected in the near future. Process modeling languages and their integration in knowledge based modeling tools are ambitious goals. General modeling languages may serve as a useful starting point for their development by adding application specific layers to the language definition. Many fundamental research issues in the area of modeling methodologies, data or knowledge engineering and human–computer interaction seem necessary to be resolved before mature knowledge based modeling tools can be expected. Even these tools will first of all address expert users to achieve higher productivity and better quality control during modeling projects. In the longer run, such systems are expected to allow dissemination of modeling and hence model-based process engineering technology on the desktops of a much larger group of engineers as it is possible today.

Acknowledgements—Part of this work has been carried out when the author was with the University of Stuttgart. The support of the Stuttgart "Forschergruppe Modellierungsmethoden" as well as of our research group at DWTH Hochen by DFG (Deutsche Forschungsgemeinschaft) is gratefully acknowledged. Further, many fruitful discussions, in particular with R. Bogusch, A. Gerstlauer, E.D. Gilles, M. Jarke, S. Lohmann, B. Nilsson, S.E. Mattson, C.C. Pantelides, K. Pahl, S. Räumschüssel, Ch. Rathke, G. Stephanopoulos and A.W. Westerberg, which have contributed significantly to a solidification of the ideas presented in this paper, are highly appreciated.

REFERENCES

Ahmed S., A. Wong, D. Sriram and R. Logcher, Object-oriented database management systems for engineering: a comparison. *JOOP* **Jun** 27–44 (1992).

Andersson M., Object-oriented modeling and simulation of hybrid systems. Doctoral Dissertation. Department of Automatic Control, Lund Institute of Technology, Sweden (1994).

Aris R., *Mathematical Modeling Techniques*. Pitman, London (1978).

Åsbjørnsen O.A., B. Meyssami and C. Sørlie, Structuring the knowledge for process modeling from first principles. Paper *Pres. at IAKE '89*, University of Maryland, College Park (1989).

AspenTech, *ASPEN PLUS User Guide*. Aspen Tech., Cambridge, MA (1988).

AspenTech, *SPEEDUP User Manual*. Aspen Tech., Cambridge, MA (1991a).

AspenTech, *Model Manager Reference Manual*. Aspen Tech., Cambridge, MA (1991b).

Augustin D. C., M. S. Fineberg, B. B. Johnson, R. N. Linebarger, F. J. Sansom and J. C. Strauss, The SCi continuous system simulation language (CSSL). *Simulation* **9**, 281–303 (1967).

Bär M. and M. Zeitz, A knowledge-based flowsheet-oriented user interface for a dynamic process simulator. *Computers chem. Engng* **14**, 1275–1283 (1990).

Barker H. A., M. Chen, P. W. Grant, C. P. Jobling and P. Townsend, Open architecture for computer-aided control engineering. *IEEE Control Systems*, **Apr**, 17–27 (1993).

Barton P. I., The modeling and simulation of combined discrete/continuous processes. Ph.D. Thesis, Department of Chemical Engineering, Imperial College, London (1992).

Barton P. I. and C. C. Pantelides: the modeling of combined discrete/continuous processes. *AIChE Jl* **40**, 966–979 (1994).

Barnstein, P. A. and U. Dayal, An overview of repository technology. Proc. 20th VLDB Conference, Santiago, Chile, 1994.

Biegler L. T., Chemical process simulation. *Chem. Engng Progr.*, **Oct**, 50–61 (1989).

Bogusch, R. and Marquardt, W., A formal representation of process model equations. Proc. ESCAPE-5, Bled, Slovenia, June 1995, *Computers chem. Engng (Suppl.)* **19**, S211–S216 (1995).

Boston J. F., H. I. Britt and M. T. Tayyabkhan, Software: tackling tougher tasks. *Chem. Engng Progr.* **Nov**, 38–49 (1993).

Bunge M., *Treatise on Basic Philosophy. Vol. 3: Ontology I: The Furniture of the World*. Reidel, Dordrecht, The Netherlands (1977).

Bunge M., *Treatise on Basic Philosophy. Vol. 4: Ontology II: A World of Systems*. Reidel, Dordrecht, The Netherlands (1979).

Cellier F. E. and H. Elmqvist, Automated formula manipulation supports object-oriented continuous-system modeling. *IEEE Control Syst.* **Apr**, 28–38 (1993).

Christaller Th., F. di Primio, U. Schnepf and A. Voß (Eds) *The AI Workbench BABYLON*. Academic Press, New York (1992).

Denn M. M., *Process Modeling*. Longman, New York (1985).

Elmqvist H., A structured model language for large continuous systems. Ph.D. Thesis. Department of Automatic Control, Lund (1978).

Elmqvist H., F. E. Cellier and M. Otter, Object-oriented modeling of hybrid systems. *ESS'93 European Simulation Symp.*, Delft, The Netherlands, 25–28 October (1993).

Fikes R. and T. Kehler, The role of frame-based representation in reasoning. *ACM Commun.* **28**, 904–920 (1985).
Fischer G. and A. Lemke, Construction kits and design environments: steps toward human problem-domain communication. *Human-Comput. Interact.* **3**, 179–222 (1988).
Gensym Corp., G2 Version 4.0 Beta Release Notes, 1995.
Gerstlauer A., M. Hierlemann and W. Marquardt, On the representation of balance equations in a knowledge based process modeling tool. *CHISA '93*, Prague, Czech Republic (1993).
Goldberg A., *SMALLTALK-80: The Interactive Programming Environment*. Addison-Wesley, Reading, MA (1983).
Haase R., *Thermodynamik irreversibler Prozesse*. Dr Dietrich Steinkopf Verlag, Darmstadt (1963). English translation: *Thermodynamics of Irreversible Processes*. Dover Publications, New York (1990).
Hayes P. J., The logic of frames. In *Frame Conceptions and Language Understanding* (D. Metzing, Ed.), pp. 46–61. W. de Gruyter, Berlin (1979).
Hayes-Roth F., D. A. Waterman and D. B. Lenat, *Building Expert Systems*. Addison-Wesley, Reading, MA (1983).
Jarke M. and W. Marquardt, Design and evaluation of computer-aided modeling tools. ISPE '95, Snowmass Colorada, June 1995. To be published in AIChE Symp. Ser.
Katz R. H., Toward a unified framework for version modeling in engineering databases. *ACM Comput. Surv.* **22**, 375–408 (1990).
Kim W., *Introduction to Object-oriented Databases*. MIT Press, Cambridge (1990).
King R., My cat is object-oriented. In *Object-Oriented Concepts, Databases, and Applications* (W. Kim and F. H. Lochovsky. Eds) pp. 23–30. Addison-Wesley, Reading, MA (1989).
King R. and R. Hull, Semantic database modeling: survey, applications, and research issues. *ACM Comput. Surv.* **19**, 201–260 (1987).
Klir G. J., *Architecture of Systems Problem Solving*. Plenum Press, New York (1985).
Kröner A., P. Holl, W. Marquardt and E. D. Gilles, DIVA—An open architecture for dynamic simulation. *Computers Chem. Engng* **14**, 1289–1295 (1990).
Kumar V., Algorithms for constraint-satisfaction problems: a survey. *AI Magz.* **13**, 32–44 (1992).
Lefkopoulos A. and M. A. Stadtherr, Index analysis of unsteady-state chemical process systems – I. An algorithm for problem formulation. *Computers chem. Engng* **17**, 399–413 (1993).
Ljung L., *System Identification*. Prentice Hall, Englewood Cliffs, NJ (1987).
Lohmann, B. and W. Marquardt, On the systematization of the process of model development. Submitted to *ESCAPE-6*, Rhodes, GR, 1996.
Lockemann P. C., G. Moerkotte, A. Neufeld, K. Radermacher and N. Runge, Database design with user-definable modelling concepts. *Data & Knowledge Engng* **10**, 229–257 (1993).
Lund P. C., An object-oriented environment for process modeling and simulation. Doctoral dissertation. Laboratory of Chemical Engineering, Norwegian Institute of Technology, Trondheim (1992).
Maffezzoni C. R. Girelli and P. Lluka, Object-oriented database support for modeling and simulation. *Proc. ESM'94*, Barcelona, June (1994).
Mariño O., F. Rechenmann and P. Uvietta, Multiple perspectives and classification mechanism in object-oriented representation. *Proc. ECAI-90*, pp. 425–430 (1990).
Marquardt W., Dynamic process simulation—recent trends and future challenges. In *Chemical Process Control CPC-IV* (Y. Arkun and W.H. Ray, Eds), pp. 131–180, CACHE, Austin; AICHE, New York (1991).
Marquardt W., Rechnergestützte Erstellung verfahrenstechnischer Prozeßmodelle. *Chem.-Ing.-Tech.* **64**, 25–40 (1992a). English translation in *Int. Chem. Engng* **34**, 28–46 (1994).
Marquardt, W., An object-oriented representation of structured process models. *Computers Chem. engng* **16S**, S329–S336 (1992b).
Marquardt W., A. Gerstlauer and E. D. Gilles, Modeling and representation of complex objects: a chemical engineering perspective. *Proc. 6th Int. Conf. IEA/AIE '93*, Edinburgh, Scotland, pp. 219–228 (1993).
Marquardt W., Towards a process modeling methodology. In *Model-based Process Control* (R. Berber, Ed.). Kluwer, Netherlands, 3–40 (1995).
Mattson S. E. and M. Andersson, OMOLA — an object-oriented modeling language. In *Recent Advances in Computer Aided Control Engineering* (M. Jamshidi and C.J. Herget, Eds), pp. 291–310. Elsevier, Amsterdam (1993).
Mattson S. E., M. Ericsson and P. Östberg, An object-oriented model of a heat-exchanger unit. *Proc. ESM'94*, Barcelona (1994).
Meyssami B. and O. A. Åsbjørnsen, Process modeling from first principles — method and automation. *Proc. 1989 Summer Computer Simulation Conf.* Austin, TX, pp. 292–299 (1989).
Müller I., *Thermodynamik — Die Grundlagen der Materialtheorie*. Bertelsmann Universitätsverlag, Düsseldorf (1973).
Newel A., The knowledge level. *Artifl Intell.* **18**, 87–127 (1982).
Nilsson B., Object-oriented modeling of chemical processes. Doctoral Dissertation, Department of Automatic Control, Lund Institute of Technology, Sweden (1993).
Nilsson, B. Experiences of developing process mdel libraries in OMOLA. ISPE '95 Snowmass, Colorado, June 1995. To be published in AIChE Symp. Ser.
Oh M. and C. C. Pantelides, A modeling and simulation language for combined lumped and distributed parameter systems. *Proc. 5th Int. Conf. Process Systems Engineering PSE'94*, Vol. 1, Kyongju, Korea, pp. 37–44. Submitted to *Computers chem. Engng* (1994).
Orchard R. A. and M. R. Tausner, General systems: a basis for knowledge engineering. *Sus. Practice* **1**, 165–179 (1988).
Pantelides C. C. and P. I. Barton, Equation-oriented dynamic simulation — current status and future perspectives. *Computers Chem. Engng* **17S**, S263–S285 (1993).
Pantelides C. C. and H. I. Britt, Multipurpose process modeling environments. *Foundations Computer Aided Design Conf. FOCAPD'94*, Snowmass, CO (1994).
Pearson R. K., Nonlinear input/output modeling. *Proc. Int. IFAC Symp. Advanced Control of Chemical Processes ADCHEM'94*, Kyoto, Japan, pp. 1–15 (1994).
Perkins J. D., R. W. H. Sargent, and R. Vásquez-Román, Computer generation of process models. *Proc. 5th Int. Conf. Process Systems Engineering PSE'94*, Vol. 1, Kyongju, Korea, pp. 123–125 (1994).
Pfeiffer B. M. and W. Marquardt, Symbolic semi-discretization of partial differential equation systems. *Proc. SC'93, Int. IMAC" Symp on Symbolic Computation*, Villeneuve d'Ascq, France, 23–25 February (1993).
Piela P. C., ASCEND: An object-oriented computer environment for modeling and analysis. Ph.D. Thesis, Carnegie-Mellon-University, Pittsburgh, PA (1989).
Piela P. C., T. G. Epperly, K. M. Westerberg and A. W. Westerberg, ASCEND: an object-oriented environment for modeling and analysis: the modeling language. *Computers chem. Engng* **15**, 53–72 (1991).

Piela P. C., R. D. McKelvey and A. W. Westerberg, An introduction to ASCEND: its language and interactive environment. *J. Man. Info. Sci.* **9**, 91–121 (1992).

Ponton J. W. and P. J. Gawthrop, Systematic construction of dynamic models for phase equilibrium processes. *Computers chem. Engng* **15**, 803–808 (1991).

Preisig H. A., On computer-aided modelling for design. *AIChE Ann. Meeting*, Los Angeles (1991).

Preisig H. A., An object-oriented approach to computer-aided modeling. *Proc. CHEMECA-1992*, Australia (1992).

Preisig H. A., Computer-aided modeling: species topology. *Proc. Int. IFAC Symp. Advanced Control of Chemical Processes ADCHEM'94*, Kyoto, Japan, pp. 143–148 (1994a).

Preisig H. A., Modeler — an object-oriented computer-aided modeling tool. *Foundations Computer Aided Design Conf. FOCAPD'94*, Snowmass, CO (1994b).

Räumschüssel S., A Gerstlauer, E. D. Gilles, B. Raichle, M. Zeitz and W. Marquardt, An architecture of a knowledge-based process modeling and simulation tool. *Proc. IMACS/IFAC 2nd Int. Symp. on Mathematical and Intelligent Models in System Simulation*, 12–16 April, Brussels (1993).

Ramkrishna D., The status of population balances. *Rev Chem. Engng* **3**, 49–95 (1985).

Rathke Ch., Object-oriented programming and frame-based knowledge representation. *Proc. 5th IEEE Int. Conf. on Tools with Artificial Intelligence*, November, Boston, MA, pp. 95–98 (1993).

Schuler H. (Ed.) *Prozeßsimulation* Verlag Chemie Weinheim (1994).

Simon H. A., *The Sciences of the Artificial.* MIT Press, Cambridge (1981).

Sørlie C. F., A computer environment for process modeling. Doctoral Dissertation Laboratory of Chemical Engineering, Norwegian Institute of Technology, Trondheim (1990).

Steele G. L., *Common Lisp. The Language* 2nd Edn. Digital Press (1990).

Stefik M. and D. G. Bobrow, Object-oriented programming: themes and variations. *AI Mag.* **6**, 40–62 (1986).

Stein L. A., H. Lieberman and D. Ungar, A shared view of sharing: the treaty of Orlando. In *Object-Oriented Concepts, Databases, and Applications* (W. Kim and F. H. Lochovsky, Eds), pp. 31–48. Addison-Wesley, Reading, MA (1989).

Stephanopoulos G., J. Johnston, T. Kriticos, R. Lakshmanan, M. Mavrovouniotis and M. Siletti, DESIGN-KIT: an object-oriented environment for process engineering. *Computers Chem. Engng* **11**, 655–674 (1987).

Stephanopoulos G., G. Henning and H. Leone, MODEL.LA. A language for process engineering. Part I and II. *Computers chem. Engng* **14**, 813–869 (1990).

Stroustrup B., *The C++ Programming Language.* Addison-Wesley, Reading, MA (1986).

Telnes K., Computer-aided modeling of dynamic processes based on elementary physics. Doctoral Dissertation, Division of Engineering Cybernetics, Norwegian Institute of Technology, Trondheim (1992).

Unger J., A. Kröner and W. Marquardt, Structural analysis of differential-algebraic equation systems — theory and applications. *Computers chem. Engng* 867–882 (1995).

Van Gigch J. P., *System Design Modeling and Metamodeling.* Plenum Press, New York (1991).

Wand Y., A proposal for a formal model of objects, In *Object-Oriented Concepts, Databases, and Applications* (W. Kim and F. H. Lochovsky, Eds), pp. 537–559. Addison-Wesley, Reading, MA (1989).

Westerberg A. W., H. P. Hutchinson, R. L. Motard and P. Winter, *Process Flowsheeting.* Cambridge University Press (1979).

Westerberg A. W., K. Abbott and B. Allan, Plans for ASCEND IV: our next generation equational-based modeling environment. Paper Presented at *AspenWorld '94*, Boston, MA (1994).

Woods E. A., The hybrid phenomena theory. Doctoral Dissertation, Division of Engineering Cybernetics, Norwegian Institute of Technology, Trondheim (1993).

Zeigler B. P., *The Theory of Modeling and Simulation.* Wiley, New York (1976).

Zeigler B. P., System-theoretic representation of simulation models. *IIE Trans.* **16**, 19–34 (1984).