Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers and Chemical Engineering 28 (2004) 1249–1266
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

Towards integrated information models for data and documents

Birgit Bayer[1], Wolfgang Marquardt*

Lehrstuhl für Prozesstechnik, RWTH Aachen, Turmstraße 46, D-52056 Aachen, Germany

Received 17 June 2002; received in revised form 26 August 2003; accepted 27 August 2003

Abstract

Numerous approaches to information modeling exist within chemical engineering representing product data, work processes, or other information. These models have a limited scope and were developed independently from each other. Thus, extended and general information models for chemical engineering are still missing as they are needed for the efficient support of work processes and for the development of domain-specific software tools. In this paper, open issues of information modeling are discussed. These are the integrated representation of information and work processes, the description of documents as carriers of data, and the integration of existing data models. The conceptual model framework CLiP is presented, which holds solution approaches for these three issues. Further, it can serve as an integration basis for existing information models. The paper further presents an overall architecture of a tool supporting the development and integration of information models.

Keywords: Information models; Product data; Documents; Workflow; Model integration

1. Introduction

Due to the increasingly competitive and global market, there is a need for the improvement of the work processes in chemical engineering. Current tool support is mainly characterized by separate or loosely linked software packages. Within these tools, growing amounts of data, documents, and all other kinds of information are handled. These different pieces of information need to be managed since they are valuable resources of knowledge. While they are created and handled within different tools, there are often dependencies and overlaps between them. Thus, automated information exchange has been recognized to be of major importance for improving and enhancing engineering work (Beßling, Lohe, Schoenmakers, Scholl, & Staatz, 1997). Empirical studies have shown, that companies are working towards integrated solutions for the management of information, but that there are still unsolved problems. These include the data exchange between heterogeneous tools or the integration of different lifecycle phases (Hameri & Nihtilä, 1998).

Most of the chemical engineering application tools have been developed for specific purposes; they have reached a high level of maturity. Therefore, a recognizable improvement of the work processes can only be achieved by an integration of these existing application tools into an environment combined with common services like document management, access to common databases or the support of work processes (Marquardt & Nagl, 1998; Nagl & Marquardt, 2001). During the last years, several proprietary software environments were developed for chemical engineering design like Aspen Zyqad or Comos PT by Innotec. In these commercial approaches, mainly tools of one vendor are tightly linked together; extensions with new tools and adaptations to the peculiarities of the work processes within a specific company are rarely supported.

A thorough understanding of the application domain is necessary for the development of open and flexible design environments that allow the integration of existing tools and provide central services and support functionalities. The tools, the information handled within these tools, and the work processes using that information need to be understood together with their dependencies.

Information modeling is a commonly used method for the analysis and formalization of information structures as

* Corresponding author. Tel.: +49-241-8096712; fax: +49-241-8092326.
E-mail addresses: birgit-bayer@web.de (B. Bayer), marquardt@lfpt.rwth-aachen.de (W. Marquardt).
[1] Present address: BASF-AG, WLF-RK, D-67056 Ludwigshafen, Germany.

0098-1354/$ – see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2003.08.008

the basis for software design (Mylopoulos, 1998). Within an information model, data and documents used within an application domain can be captured and described in a formal manner together with the work processes where they are handled. This paper will focus on open problems related to information modeling that need to be solved as a prerequisite for an efficient and flexible support of engineering work and the development of open software environments. Thus, we will look at information modeling as a step towards the support of the work processes in chemical engineering. Open problems in information modeling will be discussed together with solution approaches. The main contribution of this work is the model framework CLiP, which has been developed as a conceptual information model covering all relevant information and work processes of the chemical engineering domain. CLiP can build the basis for the integration of existing information models and tools and for the development of new models and support tools based on these models. During all these modeling activities, the different practices, tools, and perspectives of application experts participating in the chemical engineering work processes need to be considered together with the rapid changes in the markets, products, and processes (Reich et al., 1999).

Three major open problems in information modeling will be sketched in the following section. Possible approaches for the solution of these problems will be discussed in Section 3. Here, we will take the perspective of application experts developing information models as a means to understand and support their work processes and the information that is handled in their domain rather than that of computer scientists developing information models as a specification for software implementation. In Section 4, the conceptual model framework CLiP is presented, which offers solution approaches to the discussed problems. CLiP can also build the basis for the integration of existing information models and future modeling activities. A vision of a tool to support systematic and collaborative information modeling is introduced in Section 5. Section 6 concludes this paper with a discussion of the contribution of CLiP to the discussed problems and its potential use for the development of support tools for chemical engineering work processes.

2. Three open problems in information modeling

There are different and often contradictory conceptions of the term information and thus also about the focus and content of information models. Here, the terms knowledge, information, and data need to be clarified, which are of major importance in the context of information modeling. Knowledge can be defined as the fact or condition of knowing something with familiarity gained through experience or association. Thus, *knowledge* implies a knower, a person who has the knowledge. The acquisition of knowledge refers to the complex task of learning, which requires the knower's understanding and some degree of commitment (Brown & Duguid, 2000). Since knowledge is bound to a person, it cannot be transferred and communicated as such—it needs to be transformed into information.

Accordingly, *information* is defined as a—usually incomplete—transformation of knowledge as a means to communicate and exchange this knowledge (Kappe, 1999). A person can explain her knowledge to make it accessible to others; this explanation corresponds to the transformation of knowledge into information. Information is characterized by its content and its context. The content of information can be coded as *data* as it is handled in software tools and stored within data bases. Since data often refers to a technical product and its characterizing properties, the term *product data* is often used in literature. Examples for data in the domain of chemical engineering are the size of a plant equipment (e.g. volume, diameter), the operating conditions of a process step (e.g. pressure, temperature) or the physical properties of a chemical compound (e.g. density, boiling temperature). Data can be aggregated to *documents*, like reports, data sheets or flowsheets. Physically, documents can be a stack of paper or a file within a computer system. They are used as carriers for the data they hold (Rosman, van der Meer, & Sol, 1996).

Besides formal data, documents can also hold more informal information like comments or drawings. Data and documents are created and used within work processes. Besides the organizational structure within a company, the work processes provide the context of information (Abecker, Bernadi, Hinkelmann, Kühn, & Sintek, 1998). Therefore, data and documents cannot be analyzed and described completely without considering the work processes in which they are created and used. For the development of specific software tools, not only knowledge about the information structures is needed but also knowledge about the workflow and the individual activities. Within most existing information models the work processes are not considered. Thus, an *integrated description of information in the workflow* is missing.

Documentation and documents are needed and created during most work processes within a company (Zantout & Marir, 1999). Documents are major carriers of data and other, informal information. Despite of the obvious relation between documents and data, many existing information models are focusing on data as single instances of information but not on documents. Documents and data can represent the same information, but they do this in different ways. In order to be able to integrate them, their relations and the role of *documents as carriers of data* need to be clarified.

Numerous information models for chemical engineering have been developed in the past, mostly dealing with product data (e.g. Batres, Naka, & Lu, 1999; ISO 10303, 1998). An overview and discussion of chemical engineering information models is given by Bayer & Marquardt (2002). All of these models have been developed for a specific purpose. Thus, their focus and scope is restricted. Still, these models are—more or less valuable—sources of domain knowledge. The *integration of existing data models* is a first step

towards the reuse of that knowledge. Further, for the integration of software tools, the integration of their underlying data models is needed.

3. Solution approaches

For solving the three problems introduced above, it is not necessary to start from scratch. Similar problems do arise in other application domains and approaches to their solution exist. These need to be reviewed thoroughly and adapted to chemical engineering with its characteristic information, tools, and problems. In the following sections, we attempt to give an overview of existing approaches and to show how they can be applied to chemical engineering.

3.1. Information in the workflow

Activities and work processes provide the context of information. Documents and data are created and used within activities; activities are based and rely on information, its quality and availability. Thus, interdependencies between documents, data, and the work processes exist. These need to be identified and described in order to reach complete information models.

Different workflow models have been developed in the past within different companies in the chemical industries as well as in academia (e.g. Mannarino, Henning, & Leone, 1997; PIEBASE, 1998). For most of these models, a persistent integration with information models is not given. We will discuss the possibilities for such an integration on two levels of detail: by describing the workflow and the information used on a coarse level and by describing and integrating individual activities and the data they are working in a detailed way.

3.1.1. Workflow models

A workflow model should include not only the individual activities that are performed but also the information that is created and used (Allen, 2000). Besides activities and information and the associated control and information flows, workflow models should also cover the actors or their different roles within the workflow and the tools and resources that are used (Schneider & Marquardt, 2002). Further, the dynamics of the work processes need to be taken into consideration (Westfechtel, 1998). Especially, design processes are highly dynamic and cannot be predicted and modeled completely in advance. Therefore, there is a need for an evolution of workflow models over time, including for example the introduction or deletion of activities or the restructuring of the workflow towards novel engineering approaches like concurrent engineering (Cleetus, 1992).

One modeling language that allows to represent all required concepts is the C3 formalism (Foltz, Killich, Wolf, Schmidt, & Luczak, 2001). C3 is based on the activity diagrams of the Unified Modeling Language UML (Rumbaugh, Jacobson, & Booch, 1999); modeling elements to describe ill-structured work processes, communication, and tools have been added. Other languages that can be used for modeling workflows and the occurring information are the activity diagrams of the UML themselves and IDEF0 (KBSI, 2000). The document-oriented workflow model of a chemical engineering project given by (Misander, 2000) is one example for a workflow model with a focus on documents. In this model, the creation of major documents is described together with needed input documents and other relevant data.

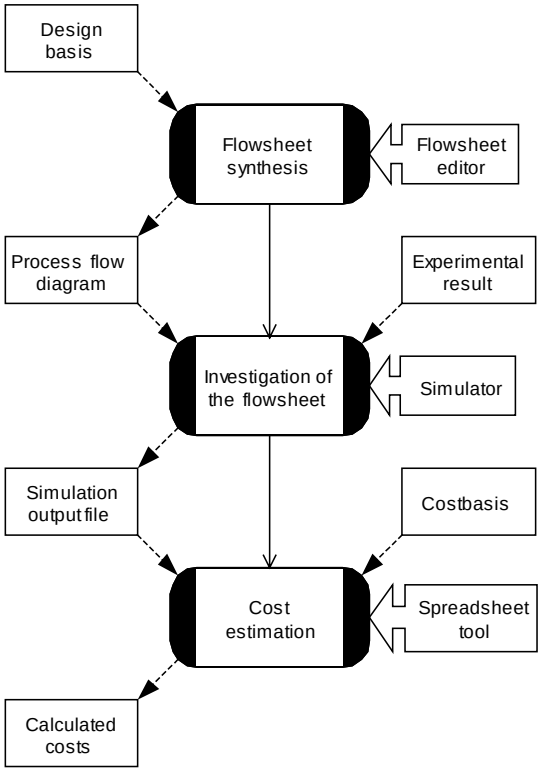

Fig. 1. Example for a workflow model in C3 notation.

Fig. 1 shows a simple workflow notated in C3, which is a typical example of a work process within chemical engineering design. Based on the design basis, a flowsheet representing a chemical process is synthesized, for example within a flowsheet editor. The process flow diagram resulting from this activity is the input information for the investigation of the flowsheet within a simulator. Here, experimental results are also used. The simulation output file, which covers the simulation results, and the cost basis, holding costs and cost factors, are used for cost estimation. This activity can be performed for example in a spreadsheet tool.

Workflow models like the one shown in Fig. 1 can be used to analyze the role of documents within a work process and to derive requirements for the support of document handling and storage. They provide the information, at what stage in the workflow specific documents are created and used and where new versions and alternatives are created. Also, information about the tools applied for document

handling can be obtained. Thus, an integrated description of the work processes, its products, and the resources is obtained. Such models are of major importance for the development of workflow management support and for the identification of required interfaces for tool integration.

Within the workflow model in Fig. 1, the activities are given on a coarse level and the information shown refers to documents and not to single data instances. The C3 formalism and other workflow modeling languages allow the hierarchical refinement of the activities. If such a refinement is realized, a parallel refinement of the information is also needed. Since workflow modeling languages are focusing on the description of activities and not on the modeling of information, such a refinement of information is often not supported. Complex data structures or dependencies between information items cannot be represented. IDEF0 (KBSI, 2000) supports the refinement of information, but most workflow modeling languages are not suitable for the description of data and activities together with their interdependencies on a more detailed level.

3.1.2. Formal integration of data and workflow models

There are several possibilities for the integration of concepts representing data and work processes into one model. One possibility leading to a weak integration is to develop a product data model and a workflow model and structure them within common partial models. Such a structuring can for example be done according to the different phases of the lifecycle of a chemical plant. An example of such a partial model could be “synthesis of process structure”, where all related activities and all data are held. The problem of such an approach is, that some data instances are used between different lifecycle phases. Thus, the product data models that need to be included in the different lifecycle-oriented partial models overlap to a significant degree. A non-ambiguous correlation of product data to such partial models is not possible.

A very stiff integration can be achieved by the definition of activities as methods or operations acting on data. Within such a product-centered view, the characteristics of the work processes are neglected. Accordingly, products can be defined as results of work processes. The internal structure of the product data cannot be represented with such a process-centered view.

None of these approaches leads to an integrated model of product data and work process information, where both are represented in an equal and balanced manner. To obtain such an integrated model, the product data and the work processes should be modeled independently in a first step. Then, mutual links should be introduced to represent dependencies. This will lead to a high number of links between the product data and the work process model making model development and maintenance an elaborate and costly task. But still, this approach seems to be the most favorable since it takes both structural and behavioral aspects into consideration in a balanced and transparent manner. Further, it provides a high flexibility for the development and usage of the models.

Fig. 2 shows a small example where a work process model describing the activities of specifying and running a flowsheet simulation is integrated via links with a small product data model. The latter represents parts of the data about the chemical process that is used within these activities. For drawing the simulation flowsheet, information about the process, the individual process steps, and connecting material streams is needed. For the specification of the reaction and the feed stream, further data is needed, which is given in the classes *reaction* and *material stream*.

A formal approach to the development of independent product data and work process models that are integrated afterwards is presented by McKay and de Pennington (2001). They introduce a general model that is applied for the representation of three aspects: the design processes, the data that is used within these processes, and the supply chain,

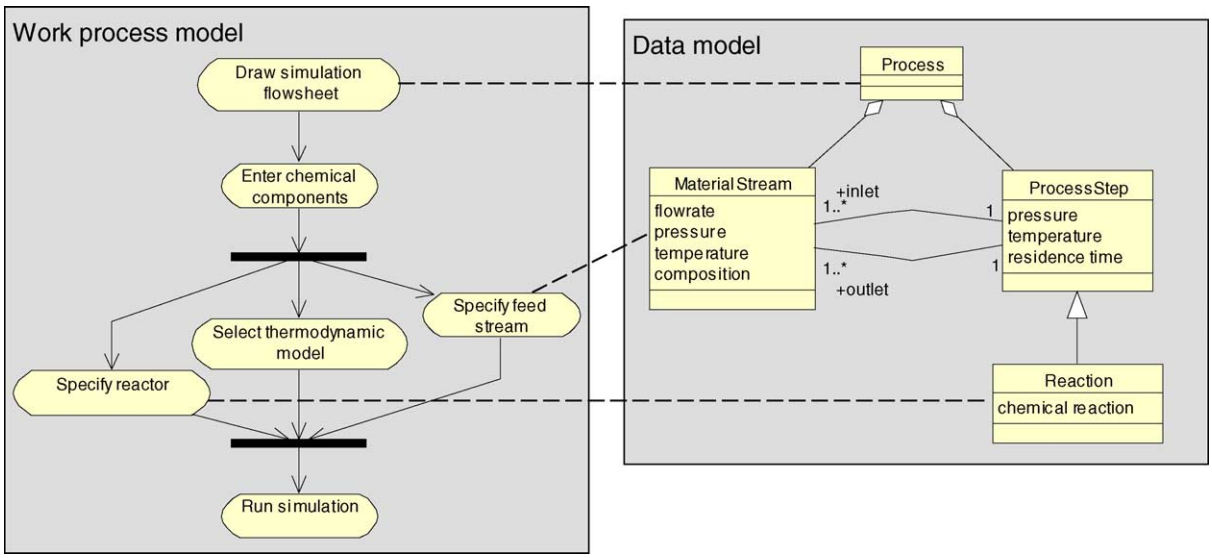

Fig. 2. Example for links between work process and product data models (notated as UML activity and object diagram).

representing the organizational structures. For the integration of these three models, interface classes are used instead of simple links. McKay and de Pennington (2001) stress that the definition of these integration classes is still an open issue. It is unclear, what is the best or the most general way to integrate the process models and the data models.

3.2. Documents as carriers of data

Documents play a major role in chemical engineering work processes: data are often handled not as single instances but in the form of documents, which can contain various data items and which build a unit of information within the work process (Salminen, Lyytikäinen, & Tiitinen, 2000). The exchange of information between different designers and business divisions and often also the exchange of data between tools is characterized by the exchange of documents (e.g. flowsheets, simulation files, equipment specification sheets). The format and the configuration of the information held in documents are usually fixed, either by the tool working on the document or by standardized document templates as they are common in the process industries for example for plant documentation. Documents are a very useful and user friendly way for the representation of information. Within chemical engineering, flowsheets and equipment specification sheets are the human perspectives on the plant and the plant data (Preece, Ingersoll, & Tong, 1994).

In the literature, different definitions of the term document do exist. All these definitions have in common, that documents are artifacts that are created and used within work processes. Thus, a document can be defined on the most general level as a logical unit of information describing the result of some activity (Westfechtel, 1998).

Documents differ widely in form and content, ranging from informal documents containing natural language text, over more formal ones like tables or simulation files with well-defined syntax and semantics to documents containing graphical data. They are created and used within different tools. Static documents, like equipment specification sheets, and executable documents, like simulation specifications, can be distinguished. Further, it needs to be considered that documents and their contents evolve over time. Each document has a lifecycle of its own, consisting of creation, editing, review, approval, and release processes. During such a document lifecycle, different versions of a document are created that need to be managed. Also, several alternatives of a document might exist representing different alternatives of the product (Peltonen, Pitkänen, & Sulonen, 1996).

For a complete description of documents, different types of information models are needed (Salminen et al., 2000): object models representing the documents themselves and their relations among each other in the sense of a *taxonomy*; models describing the *dynamic behavior of the documents* over time, i.e. their versions and status; and finally models focusing on the *contents of the different documents* with their structure and dependencies. In the remainder of this section, these different document models are presented.

3.2.1. Taxonomy of documents

A taxonomy of documents describes the different types of individual documents together with their dependencies. For the development of such a model, the documents occurring within a work process need to be identified. This can be done on the basis of workflow models (e.g. Misander, 2000) or project execution manuals as they are available in various chemical engineering companies. Another possibility to identify documents are case studies developed on the basis of empirical studies, actual work processes, and literature knowledge (e.g. Bayer, Eggersmann, Gani, & Schneider, 2002; Eggersmann, Krobb, & Marquardt, 2002).

Based on these sources, a taxonomy of documents has been developed as shown in Fig. 3. The collection of the given documents is not necessarily complete. The modeling language UML is used here to describe the different types of documents and there dependencies (indicated by dashed arrows); the full potential of the object-orientation of the UML (i.e. the definition of methods) has not been employed. Within the *design basis*, the requirements and product specifications are given. Based on this, the layout of the process and plant are developed, which are represented in *flowsheets*. *Process flow diagrams* and *piping and instrumentation diagrams* can be distinguished. For the different items represented on these flowsheets, specifications are developed, i.e. *equipment specifications*, *piping specifications*, *instrument specifications*, and *safety valve specifications*. Within the *medium list*, all occurring components and process media are listed. Mathematical *models* are used to describe the process behavior. The models are adjusted to *experimental results*. They are implemented within some simulators; these tools are working on different documents including *simulation input files* and *simulation output files* covering the *simulation results*. Based on the *cost basis* and the simulation results, the *calculated costs* can be obtained. Further, there are *reference documents* (i.e. *books*, *articles*, and *patents*) and different *schedules* used for project administration.

On a more general level, Zantout and Marir (1999) identify three main types of documents: *reference documents* which represent a static resource of information, *collaboration documents* handled within a project by different users and evolving over time, and *transaction documents* used for communication and documentation. Reference documents within chemical engineering are for example material data sheets, patents, and technical articles. The different types of flowsheets are used for collaboration as well as for transaction throughout the entire lifecycle of a chemical plant.

3.2.2. Dynamic behavior of documents

Documents do evolve over time; they are changed in their content and used within different contexts and by different users. The dynamic behavior of a document over its lifecycle can be represented using state transition diagrams. Fig. 4

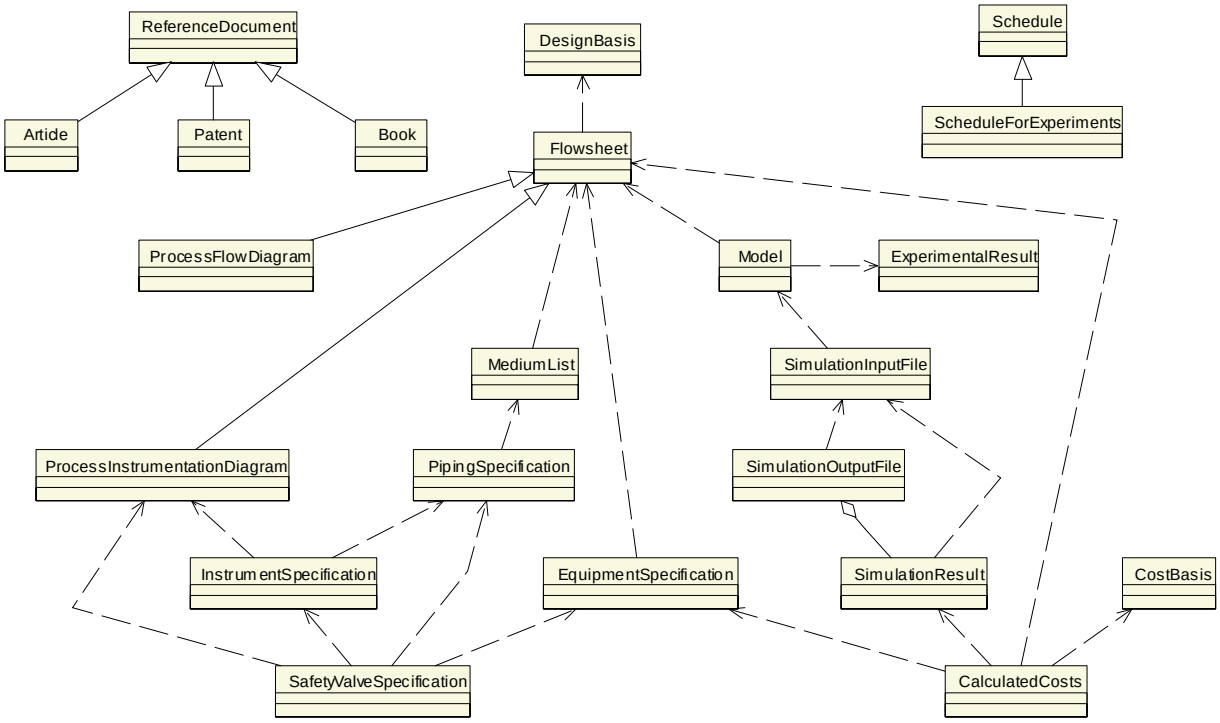

Fig. 3. (Incomplete) taxonomy of documents used in chemical engineering design (class diagram of the UML).

shows exemplarily some states of a process and instrumentation diagram (P&ID).

The P&ID comes into existence in its first pass. It is developed on the basis of the process flow diagram and the design basis (Misander, 2000). This first pass can be modified repeatedly, as long as design modifications are required. This is indicated by a transition in Fig. 4. When there are no further modifications required and after the completion of the equipment specifications (which are given in other documents), the second pass of the P&ID can be created. The P&ID will be released after the equipment specifications have been finally adjusted.

Models like this can form a basis to derive requirements for the support of document handling and storage; they provide information about document creation and use including document versions and alternatives. For the development of

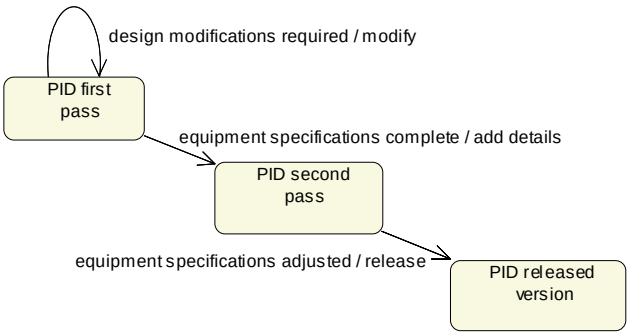

Fig. 4. Different states of a process and instrumentation diagram P&ID (state transition diagram of the UML).

document management facilities, it is necessary to consider also dependencies between the different document types as they are shown in Fig. 3. The single versions of the individual documents and their alternatives depend on each other leading to complex configurations of documents that need to be managed. Also, changes in one document might induce changes in other documents in order to maintain consistency between them and will thus lead to changes in the actual work process. Information about such consequences can be obtained from workflow models as they are discussed in Section 3.1.1.

3.2.3. Model of document contents

Documents depend on other documents. These dependencies can be refined to dependencies between the document contents. As defined in Section 2, a document is an aggregation of data. For documents that have an internal structure—as it is the case for most technical documents—a representation of the document content can be achieved on various levels of detail. On the most detailed level, each single data instance contained within the document is described.

One possibility to represent such models of document contents is the use of the eXtensible Markup Language (XML; W3C, 1997) and its Document Type Definitions (DTDs). XML deals with the representation of the logical, internal document structures. Within a DTD, the structure of a specific document type can be described. Fig. 5 shows the (incomplete) DTD of a specification sheet for vessels and reactors (which is a special type of the document *equipment specification* in Fig. 3). It is possible to relate single

```
<!ELEMENT VesselSpecificationSheet (Header,EquipmentSpecification,Footer)>

<!ELEMENT Header (Site,Project,Plant,Version)>

<!ELEMENT EquipmentSpecification (Title,ProcessData,MechanicalLayout,Comments)>
    <!ELEMENT Title (#PCDATA)>
    <!ELEMENT ProcessData (MaterialInformation,EquipmentData,Temperature,Pressure)+>
        <!ELEMENT MaterialInformation (Material,LiquidPhase?,VaporPhase?)>
            <!ELEMENT Material (Name,WaterHazardClass)>
            <!ELEMENT LiquidPhase (LiquidDensity,LiquidMassFlow)>
            <!ELEMENT VaporPhase (VaporDensity,VaporMassFlow)>
        <!ELEMENT EquipmentData (Volume,InsideDiameter,HeatingCooling?, ConstructionMaterial)>
            <!ELEMENT HeatingCooling (Type,Area)>
    <!ELEMENT MechanicalLayout (MaximumPressure,MaximumTemperature,MaximumLiquidLevel)+>
    <!ELEMENT Comments (Comment*)>
        <!ELEMENT Comment (#PCDATA)>

<!ELEMENT Footer (Company,OrderNumber,PersonInCharge,InspectionNote,ReleaseNote)>
```

Fig. 5. DTD of a vessel specification sheet (incomplete).

elements of this DTD to classes and attributes of a chemical engineering data model. Thus, a document type can be interpreted as a view on a data model and a specific document as a view on the data represented by the data model at a distinct point in time. It is possible to include unstructured and semi-structured information elements within a DTD, that are not necessarily part of a data model but hold important information for the users (for example, the DTD element *comments* in Fig. 5).

The expressiveness of such document type definitions is rather restricted. Only the syntax of a document and of its contents can be given. No information about the semantics of the contents, i.e. about their meanings, can be provided. Consequently, no further information about the behavior of executable documents like simulation specifications can be supplied. In order to be able to handle these aspects of documents, more expressive document specifications than XML DTDs are needed, like for example the specification language PROGRES (Schürr & Zündorf, 1996). Another possibility to provide the single document elements with meaning and context is to link them to the concepts of a data model.

3.3. Integration of data models

In recent years, numerous data models have been developed in the area of chemical engineering and in related disciplines. Bayer and Marquardt (2002) give an overview and a comparison of different chemical engineering data models. Each of these data models has a specific and original scope and is built for a distinct purpose. Thus, their coverage is limited compared to the requirements that can be stated for a support of the work with data over the lifecycle of chemical plants. Still, within these data models, knowledge about the domain of chemical engineering is captured. Therefore, there is need for the integration of the already existing approaches in order to reuse the knowledge they hold.

This integrated knowledge can then be employed for the development of integrated services to support the engineers' work with data. Another aspect of the integration of data models is the fact that many software tools are based on a—more or less formalized—data model. Thus, the first step towards the integration of tools is the integration of their underlying data models.

If there were no relations between the different existing data models, their integration would simply lead to an overall data model obtained from the concatenation of the single models. But models developed independently by different researchers and application experts do overlap in scope and content. Often the same concepts are represented differently so that conflicts occur during integration. Integration of data models is a well-known problem in the areas of database and data warehouse construction. Different mechanisms and techniques have been developed to integrate and to deal with several information sources. All these approaches can be classified into two main categories (Calvanese, de Giacomo, Lenzerini, Nardi, & Rosati, 2000): the definition of *mappings* between different schemata and the construction of *one integrated schema*.

Before these two categories will be summarized briefly, a more detailed overview of possible conflicts during data model integration will be given. This section will be concluded with a discussion on a global data model as the basis for the integration of existing data models.

3.3.1. Model diversity and conflicts

Any data model is developed with a certain intention and for one or a small number of specific applications. Therefore, data models do differ, even if they are focusing on a similar data scope; they can represent different viewpoints on the same objects. Data models can differ in their modeling language and syntax, in the use of modeling constructs, in their level of detail, and in their concepts and semantics. Differences arise from naming conventions, classifications, and assumptions. When data models are going to be integrated, these differences cause a high specification effort: concepts with identical and equivalent meanings in the

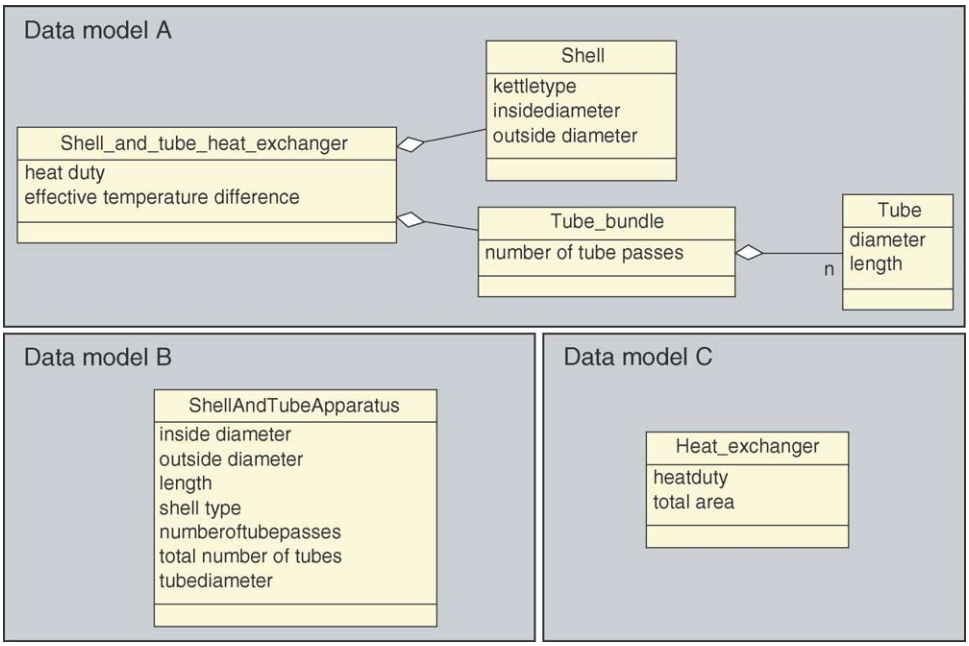

Fig. 6. Data models of heat exchangers.

different models have to be identified and relations between them must be formulated. Further, conflicts need to be detected and compensated.

Overviews over possible conflicts between data models are for example given by Batini, Lenzerini, and Navathe (1986) and Pitoura, Bukhres, and Elmagarmid (1995). In general, schema, semantic, and data conflicts can be distinguished, where schema conflicts can be further discriminated in naming conflicts (homonyms and synonyms) and structural conflicts. Further, relations may exist between the data models to be integrated that are not formalized in either one of the models. These relations are called interschema relations (Catarci & Lenzerini, 1993).

We want to illustrate the different conflicts and relations with a small example from chemical engineering. In Fig. 6, parts of three (fictive) chemical engineering data models are given that represent plant equipment that can be used for heat exchange. Data model A is the most detailed one, while data model C is on the coarsest level of detail. Comparing model A and model B, first of all *structural conflicts* can be identified: while the shell, the tube bundle, and the tubes are modeled as elements of the shell and tube heat exchanger in A, they are described with their properties as attributes of the shell and tube apparatus in B. Other examples for structural conflicts are different taxonomies used in different data models for the representation of class hierarchies (not shown in the given example). The attributes kettle type in A and shell type in B are synonyms, i.e. there is a *naming conflict* with model elements named differently that represent the same concept. The attributes length in A and length in B are homonyms: the same name is used for different concepts (length of the single tube compared to the length of the overall apparatus). There is also a *semantic difference* between the models A and B: while shell and tube heat exchanger in A represents explicitly some heat transfer equipment, an instance of the shell and tube apparatus in B is not automatically a heat exchanger. It can also be an apparatus fulfilling a different function within the plant while having the shell and tube geometry (e.g. a tubular reactor). This difference is not as obvious as the structural and naming differences, since it is constituted in the model interpretation and not in the model itself. One indicator of this semantic difference is the absence of attributes representing duty and heat exchange area in model B.

When tools are integrated that work on these different data models, additionally to the conflicts discussed so far, *data conflicts* may occur due to inconsistent data values (e.g. the heat duty is given as 0.951 kW in tool A and as 948 W in tool C) and different expressions used for the same data (e.g. different units, different precisions, or different reference states).

Some *interschema relations* exist between data model A and C. The class heat exchanger of C is a superclass of the shell and tube heat exchanger of A. This relation is not formalized in any way within one of the two models and can only be identified by comparing the concepts and their interpretation. The characteristic properties of these two concepts given by their elements and attributes can also be related to each other. The attribute heat duty represents the same concept in both models. A more complex relation can be found for the heat exchange area: the attribute total area of the heat exchanger in model C can be derived from the diameter and length of the tube in model A and the number of tubes given by the multiplicity at the aggregation association between tube bundle and tube. This derivation

works only in one direction: the calculation of the number of tubes and their geometry simply from the total area is not possible.

A lot of work has been done on the (semi-)automatic detection of such conflicts as well as on their resolution (see, for example, Castano, de Antonellis, & de Capitani di Vimercanti, 2001; Reddy, Prasad, Reddy, & Gupta, 1994). But still a lot of this work needs to be done manually.

3.3.2. Definition of mappings between schemata

If the data models overlap in their scope, *mappings* can be defined to connect equivalent concepts within the different models. These mappings or transformations can be used for a unified presentation of heterogeneous data models. This mechanism is called schema translation (Pitoura et al., 1995).

A quite simple mapping was introduced by Mariño, Rechenmann, and Uvietta (1990): *bridges* are defined to link classes describing the same concept within different class hierarchies. In this particular approach, the class hierarchies represent the perspectives of different experts of one domain. They are modeled using one formalism and integrated afterwards by means of the bridges. In this *a priori approach*, concepts within the different hierarchies can refer to exactly the same object and conflicts do not occur.

In contrast, there is usually a mismatch between the content and semantics when existing data models are to be integrated in an *a posteriori approach*. This mismatch has to be overcome by means of a *transformation*. Therefore, the bridges need to be enriched by some transformation rules and constraints that reconcile the semantic differences between the hierarchies. Transformation rules can for example be defined using description logics (Catarci & Lenzerini, 1993). Another possibility are graph grammars that can be used as a formal basis for the integration of data models and application schemata (Claypool & Rundensteiner, 2001). It is thus possible to formalize interschema knowledge that can be used for checking consistency between different data models or for querying multiple sources. This knowledge can be used later on to integrate the tools based on the integrated data models (Becker, Haase, Westfechtel, & Wilhelms, 2002; Gruner, Nagl, & Schürr, 1998).

Model integration via transformations is not restricted to any number of models. But an integration of N models requires the definition of up to $N(N-1)$ transformations to map all models. Therefore, it is useful to have one neutral or global model to which all other models can be related. This reduces the number of necessary transformations to a maximum of $2N$, if all models are indirectly integrated with each other (Book et al., 1994). The introduction of a neutral data model reduces the quantity of transformations to be defined, but not their quality. Thus, there is still a significant specification effort related to their development. In Section 3.3.4, it will be discussed how a neutral data model for model integration should look like.

3.3.3. Construction of an integrated schema

Another possibility to integrate data models besides the definition of mappings is to *merge the different modeling schemata into an integrated schema*. In this area, research has been done for a long time in the context of database construction. Batini et al. (1986) give an early review; new approaches and results are still published.

All approaches of schema integration follow a sequence of different methodological steps including: *preintegration*, where the schemata are analyzed and an integration policy is set; *schema comparison*, which deals with the detection of correspondences and conflicts; *schema conforming*, where the conflicts are resolved; and finally *schema merging and restructuring*, where the conformed schemata are superimposed to an integrated schema. Different data models serve as the input, the output consists usually of one integrated schema and mappings between this integrated schema and each input model.

During schema conforming, some modeling work needs to be done: parts of a model that are incomplete compared to corresponding parts in other models might be left out; unification between differing models is needed; and the introduction of novel models might be necessary. The merging of different data models into one schema is a laborious and difficult task. Again, interschema knowledge, that is not captured within the models to be integrated, is necessary in order to find analogous and related modeling concepts to create the integrated schema.

Another major drawback of the merging approach for integrating data models is obvious: for each new data model to be integrated, a new integrated schema as well as new mappings or modified queries for all input data models have to be created. Even though work has been done in order to support and automate the merging of data models, this integration remains laborious and inflexible.

3.3.4. A global model as the basis for data model integration

This brief overview shows the difficulties related to the integration of existing data models. But some model integration is necessary as a basis for tool integration and in order to utilize knowledge already captured in existing data models.

The development of an integrated schema by merging existing data models is a reasonable strategy if the major goal of data model integration is the reuse of domain knowledge captured in existing data models. Such an integrated schema needs to be changed when a new data model is to be introduced. Thus, this approach is unfavorable, when the integration of data models shall be employed mainly for the integration of existing application tools with new tools might introduced frequently—as it is the current situation in chemical engineering. Then, the approach of integrating the data models via mappings is more promising. A *global model* is viable for such an integration approach when a large number of models or, respectively, of tools is going to be integrated.

Despite the tremendous efforts spent in the development of data models, none of them is established as a global model for integration or as a basis for the development of new models. Standards like the models of the STEP initiative (e.g. ISO 10303, 1998) have been developed where product data are described on a detailed level. These have not gained acceptance, neither in the chemical process industries nor with the software vendors working on chemical engineering software. Some of the models developed in the context of STEP have been applied during software development as a basis for data exchange and central data storage. But in all these applications, the data models have been modified, changed, or extended. This lack of acceptance must be interpreted as an indication that it is not possible to define a detailed data model that serves all potential purposes—even within one application domain.

Still, a global data model is needed as a basis for model integration. But instead of a detailed data model, a conceptual model should be developed that can serve as a framework for the integration of existing and the development of new data models (McKay, Bloor, & de Pennington, 1996). Such a conceptual model should provide structural integrity and extensibility as they are needed for sound extensions of the model to support new or user specific data requirements.

4. The conceptual information model CLiP

In the last section, the description of dependencies between information models and work process models, the relations between data and documents, and the integration of existing data models have been discussed. These three tasks need to be solved in order to obtain *integrated information models for chemical engineering*. Within such models, knowledge about the domain of chemical engineering can be provided as the basis for the development of specific support tools.

The solution of these three tasks requires the integration of different models: the integration of product data and workflow models, the integration of different document and product data models, and of course the integration of existing data models. Therefore, a similar basis can be used to address them: a conceptual model or model framework that allows the representation of data, documents, work processes, and their relations. Together with appropriate methodologies and support tools, such a model framework can serve as a basis for the integration of existing information models and for the sound development of new ones. The model framework should cover the basic concepts of the domain and describe them in the sense of an ontology (Uschold & Gruninger, 1996). It can then serve as a common vocabulary in the sense of shared understanding (Konda, Monarch, Sargent, & Subrahmanian, 1992).

Due to the increasing complexity and to dynamic changes in the domain of chemical engineering, it is not possible to determine all information items, work processes, and their relations for all projects and situations a priori in an information model (Reich et al., 1999). Still, we believe that a conceptual framework for information modeling can be defined capturing all aspects of information on a level of high generality. It is necessary to enrich and complete such a framework with more detailed concepts as a basis for tool development and information management in a specific context. These detailed concepts can be defined in the form of hierarchical models as the ones defined for example in the STEP application protocol 231 (ISO 10303, 1998) or as a flat space of information objects as proposed by Reich et al. (1999). The way they are defined depends on the specific application.

In this section, an overview of the *C*onceptual *Li*fecycle *P*rocess model CLiP will be given, which has been developed within the Collaborative Research Center IMPROVE (Marquardt & Nagl, 1998; Nagl & Marquardt, 2001) as a basis for the understanding of design processes in chemical engineering and for the development of specific computer-based support tools. It is a conceptual information model and thus independent from specific applications. Its structure is well-defined and open. The model comprises concepts for the description of product data and documents as well as for work processes in an integrated manner. Thus, CLiP can be a starting point for the development of a common model framework for information model integration and subsequent enrichment. A detailed description of CLiP is given in an extensive report by Bayer, Krobb, and Marquardt (2001).

4.1. The overall structure of CLiP

The conceptual model framework of CLiP covers three layers of meta model classes, which represent modeling concepts on different degrees of abstraction (see Fig. 7): the meta meta classes describe general systems; different technical systems and social systems are defined with their relations using meta classes; and simple classes represent the chemical process system with its subsystems and other, related systems.

As the root concept of the model framework, the *system* is introduced as a meta meta class according to the ideas of systems engineering (Patzak, 1982). A system can be characterized by its properties and their different values (not shown in Fig. 7). It can be decomposed into one or several parts, which are systems themselves as indicated by the *contains*-association. Systems can also *refer to* other systems, i.e. they can be interrelated or dependent. One special kind of reference between different systems is the *models*-association: one system can serve as a model for another one.

Different kinds of systems can be distinguished on the meta class level. *Technical systems* represent all kinds of technical artifacts. They can be decomposed and do interact with other technical systems. Technical systems are either *devices* or *connections*. Devices have the major functionality and are linked by connections. Furthermore, *material* and *social systems* are introduced as instances of system.

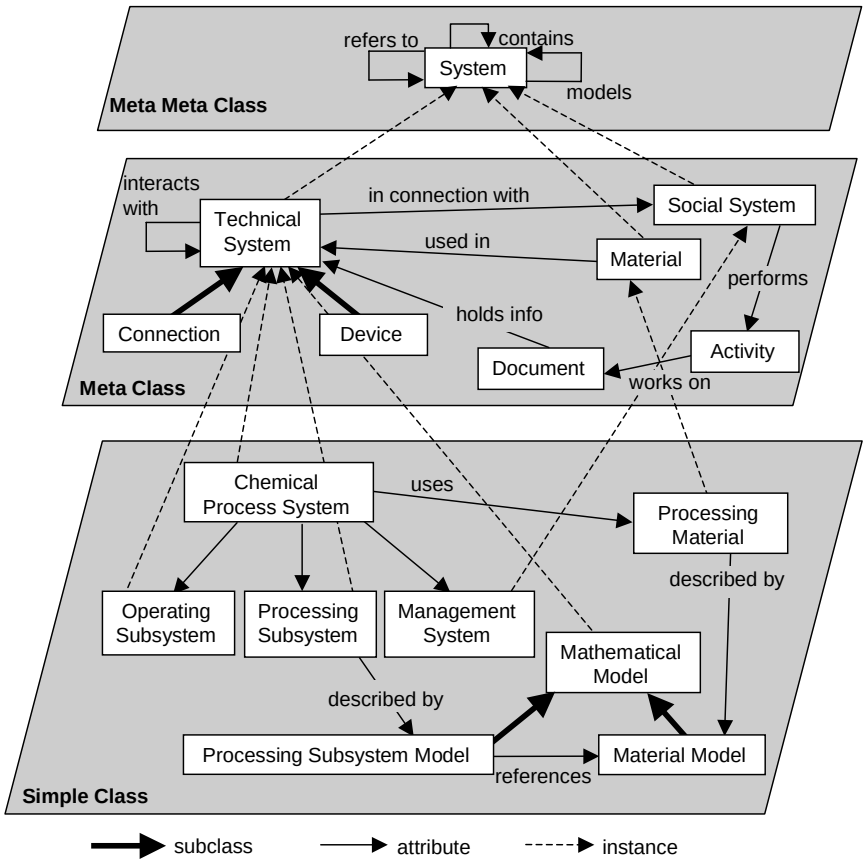

Fig. 7. Model structure of CLiP.

Material abstracts matter and substances, that can be used in various manners by technical systems. A social system can be a group of persons or a single person. The most important aspect of social systems are the performed *activities*. Activities are working on *documents* that hold information, for example, about the technical system under consideration.

Within the simple classes, the concept of technical systems is further refined to *chemical process systems* which consist of three distinguished parts: the *processing subsystem*, the *operating subsystem*, and the *management system*. The processing subsystem holds functionalities of material processing, the operating subsystem comprises the technology for controlling the processing subsystem, and finally, the management system refers to the personnel working on the chemical plant. Processing and operating subsystem are instances of technical system, whereas the management system is an instance of social systems. There are two different instantiations of material on this degree of abstraction: the *processing material* which is processed in order to get a specified product and the construction material (not shown in Fig. 7) used to build the chemical process system. The behavior of these materials can be described by *material models*. These are referenced by *processing subsystem models* describing the processing subsystem. Material models and processing subsystem models are *mathematical models*, which are instances of technical system.

4.2. Partial model structure of CLiP

CLiP is divided into partial models holding concepts that belong logically together. Fig. 8 shows the partial models related to the chemical process system; these partial models correspond to some of the concepts introduced in Fig. 7: the main concepts of CLiP introduced on the different meta levels provide the overall structure of the model. Within the partial models of the chemical process system and its three (sub-)systems, all information about these systems is given. This information comprises the different properties of the systems. The concepts related to these properties are again grouped to partial models that are nested within the partial model of the system itself. In Fig. 8, three partial models are shown used to model different properties of the chemical process system: function, realization, and behavior.

The partial models can be modeled and used largely independently from each other. Still, there are a lot of interdependencies between them. These are modeled explicitly in CLiP with associations between concepts belonging to different partial models. By dividing the model into parts and reintegrating these with associations, an open and extensible model structure is obtained. New partial models can be introduced by developing them independently and then describing their relations to the existing ones.

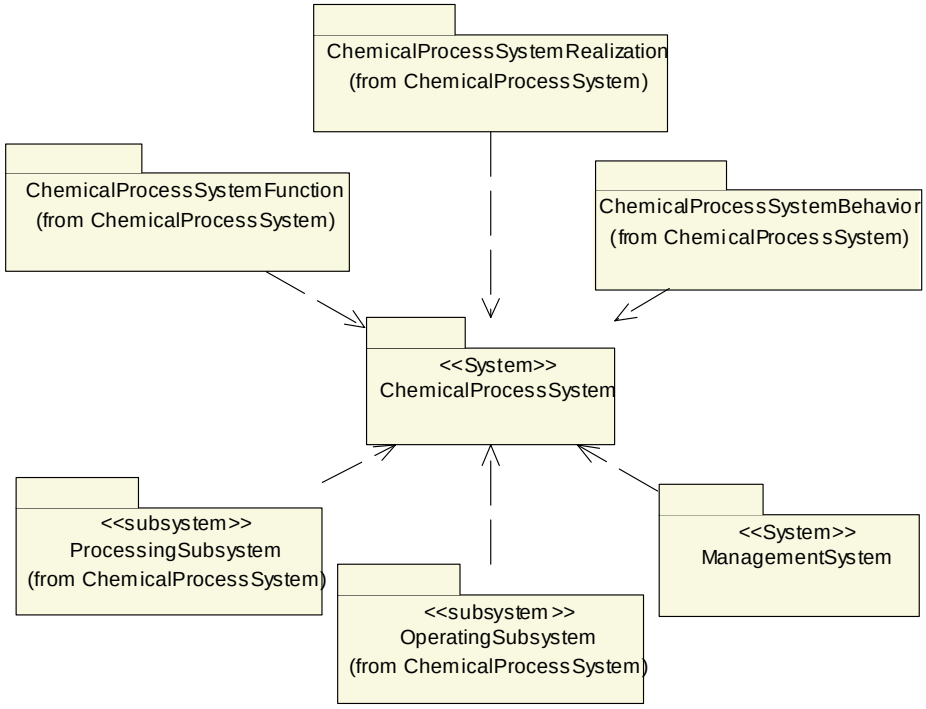

Fig. 8. Partial models on the simple class level of CLiP (represented by UML packages).

4.3. Integration of existing data models with CLiP

Due to its structure and its independence from a specific application, CLiP can serve as a basis for the integration of existing data models. This conjecture has been checked with the extension of CLiP with concepts for the description of process control functions and their realization within process control systems (Bayer, Schneider, & Marquardt, 2001). Originally, the focus of CLiP was set on the processing subsystem, its function and realization. The entire meta and partial model structure was developed with this focus. Afterwards the model was extended with concepts describing parts of the operating subsystem and its properties. This was done by integrating existing data models covering control loops as well as sensor functions and actuator functions as they are given by Polke (1994). Also, data models describing elements of process control systems (i.e. a possible realization of process control functions) have been integrated. There was no need to change the overall structure of CLiP for this integration, which can be seen as an indication for the generality of that structure.

Further, some integration studies were performed looking at the possibilities to integrate existing data models of a similar scope with CLiP (Bayer, Schneider, & Marquardt, 2000). This was not done with the goal to extend the data model but to integrate existing models as a possible basis for the integration of software tools. These studies have shown that CLiP can serve as a global schema for data integration via mappings as discussed in Section 3.3.2.

4.4. CLiP and documents

Within CLiP, documents and their contents are modeled explicitly on the same level as the different systems. This allows the integrated description of the properties and the data characterizing a system and of the documents where this data is aggregated. The main concepts for the description of documents are given on the meta level (see Fig. 9).

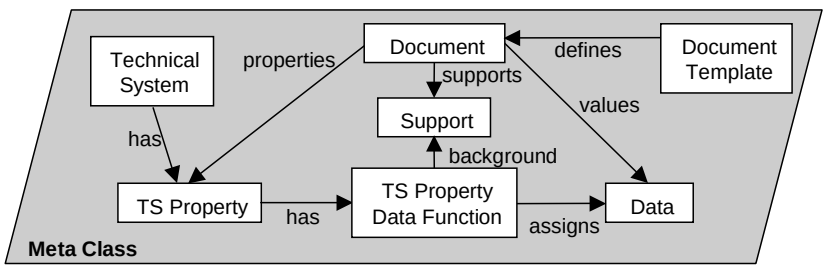

Fig. 9. Concepts describing data and documents within CLiP.

Like any other system, a technical system has some properties (*TSProperty* in Fig. 9) that are used to characterize and describe it. Every property has multiple possible appearances given as *data*. At one specific observation, i.e. at a specific point in time or for one measurement, each property can be described only by one specific data. For the identification of that observation the concept of the *support* has been introduced, in which the background of an observation can be given (Klir, 1985). The support contains the data, which build the background for an observation (i.e. the exact point in time or the location and procedure of the measurement); typical supports are time, space, and population coordinates. The complex relation between technical system property, data, and support is described explicitly by the *technical system property data function*. This property data function assigns a property to that data out of a set of possible data, that is actually observed at a specific support. It can refer to a data log in the case of measurements where the data of a property are recorded for different supports, usually over some time interval. In the case of mathematical simulations, the property data function can refer to a mathematical function.

Documents contain information, i.e. they contain some properties together with their data and the support of that data. Many documents that are used in technical design processes have a (partially) predefined content. The structure of the documents is often specified, either through the software tool that is working on this document or by some standards. Therefore, *document templates* can be defined for many documents, that refer to the properties that have to be given in the specific document, but that do not contain any data.

The relations between properties, data, support, documents, and document templates shown in Fig. 9 for the technical system are the same for the other systems on the meta level of CLiP. It is also possible, that a document contains information about several different systems. The modeling concepts introduced here are instantiated on the simple class level. Here, the class *document* is refined to *chemical engineering document*. Different document types, as they are for example given in Fig. 3, are modeled as subclasses of this chemical engineering document class. Thus, taxonomies of documents as they were discussed in Section 3.2.1 can be integrated formally within CLiP.

Within Fig. 10, the refinement of the relations between a document or, more exactly, a document template and the data model describing the processing subsystem is shown. The XML document type definition from Fig. 5 is an example for a document template of a chemical engineering document. It is given together with two classes from CLiP representing a vessel and a material stream within a plant. The classes and their attributes (i.e. properties of the processing subsystem) can be related to the elements of the DTD as it is indicated by the arrows.

Such relations can form the basis for an automatic generation of documents from a database. They can also be used for the exchange of information between documents and databases, i.e. between tools that work on documents (e.g. simulators running with input files) and databases (e.g. design databases, where information about the chemical process is stored).

4.5. CLiP and work processes

Activities are steps within a work process, in which some input information is modified or deleted and thereby some output information is produced. According to Westfechtel (1998), the logical unit of information that serves as input or output for an activity can be defined as a document. These interrelations have been used within CLiP for the integration of models describing the activities within a workflow with document models and detailed data models (see Fig. 11). The document class can be interpreted as an interface class between the product data and the work process models (cf. McKay & de Pennington, 2001).

According to the definition of a system's function as its ability to transform some input into a required output, which

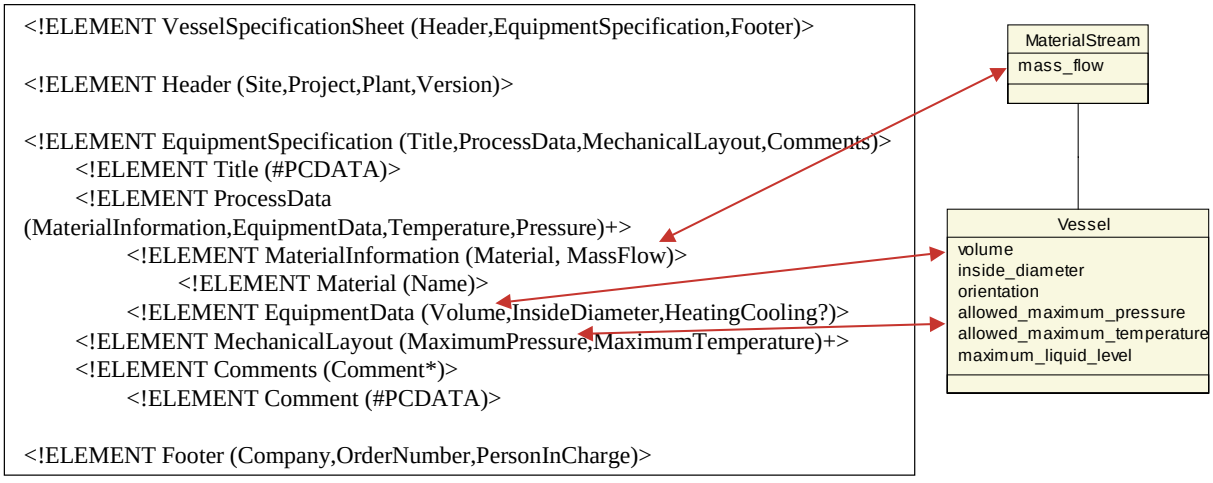

Fig. 10. Relation between a specification sheet (Fig. 5) and CLiP.

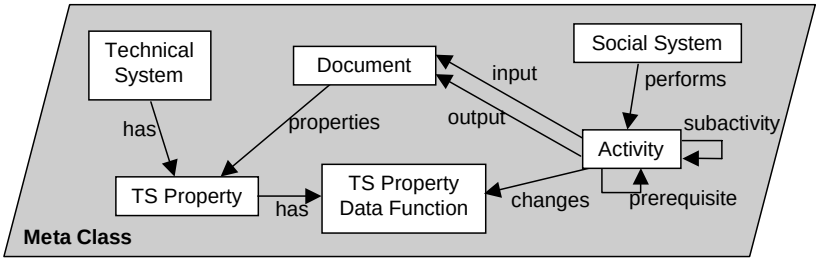

Fig. 11. Concepts describing activities within CLiP.

has been used throughout the development of CLiP, *activities* are introduced as the function of social systems. They can be decomposed hierarchically into subactivities. The order of activities can be defined by prerequisites, which are activities that need to be executed before the activity they belong to. A more detailed description of activities and their characteristics is given by Eggersmann, Krobb, and Marquardt (2000).

The input and output documents of an activity can be specified further to classify the quality of the information used within that activity. It can be specified if the information within the referenced document is exact or inexact, must or may not exist, and is complete or incomplete. Activities represent manipulations of information performed by an engineer or some software. This is described within CLiP by the *changes*-association between activity and technical system property appearance function (and other property appearance functions not shown in Fig. 11); the activity does change not the property of system itself but its assignment to data and support.

5. Tool support for information model development

In Section 3, three open problems in information modeling and previous solution approaches have been discussed. It has been stressed that their successful solution requires the integration of different information models. The conceptual information model CLiP has been introduced as a framework for such integrations. Model integration is a complex and laborious process, which should be supported by some appropriate methodology and software. No information model can cover all aspects of information that are needed; probably no standard will be accepted in all its details by all stakeholders. Therefore, it is also necessary to support the processes of model adaptation and extension as well as the development of further information models in order to meet specific application needs (e.g. for the development of a specific software tool or for the representation of an internal standard within a company). Here, we want to present a vision of some tool support for the development and integration of information models based on an ontological framework like CLiP.

Our suggestion is grounded on the *analogy* between *mathematical models* used for the description and prediction of the behavior of chemical processes and plants and *information models* used to describe the data, documents, and their dependencies. Within both kinds of models, knowledge about a system is represented in a formal manner. Mathematical models are used for tasks like simulation and optimization, while information models can be used for the reengineering and optimization of work processes as well as for the development of information management facilities or design and manufacturing support tools.

For the development and usage of mathematical models, tools have been established that provide the user with a library of model blocks that can be customized for a specific application. The coverage of their libraries and their flexibility ranges from tools like Aspen Plus and Hysys, where a comprehensive library of model building blocks is given and only parameters have to be adjusted, over modeling and simulation tools like gPROMS or Aspen Custom Modeler, where the user can define its own models and only very generic model building blocks are provided that can be freely modified, to tools like Dymola that provide both, completely free modeling facilities and predefined model building blocks. From such a range of tools the appropriate one can be chosen, depending on the system to be modeled, the model requirements, and the user's experience.

We suggest to adapt this successful approach of tool support from the development of mathematical models to the development of information models: a *modeling tool for information models* for the domain of chemical engineering. A possible architecture of such a tool is sketched in Fig. 12; it consists of a model base and modules for the import, integration, development, use, and export of models. These modules should provide functionalities for different activities in information modeling indicated by arrows in Fig. 12.

Within the *model base*, a conceptual model framework such as CLiP is available together with information models originating from different sources that are integrated with the model framework via mappings. Thus, the model base provides the user with a library of existing information models and model building blocks. For an efficient model management, it is necessary to represent not only the different information models explicitly within the modeling base but also the mappings between them (Bernstein, Halevy, & Pottinger, 2000).

Existing data models as well as application documents and database schemata are sources of domain information, which are cheap, valuable, and easily available compared to the—

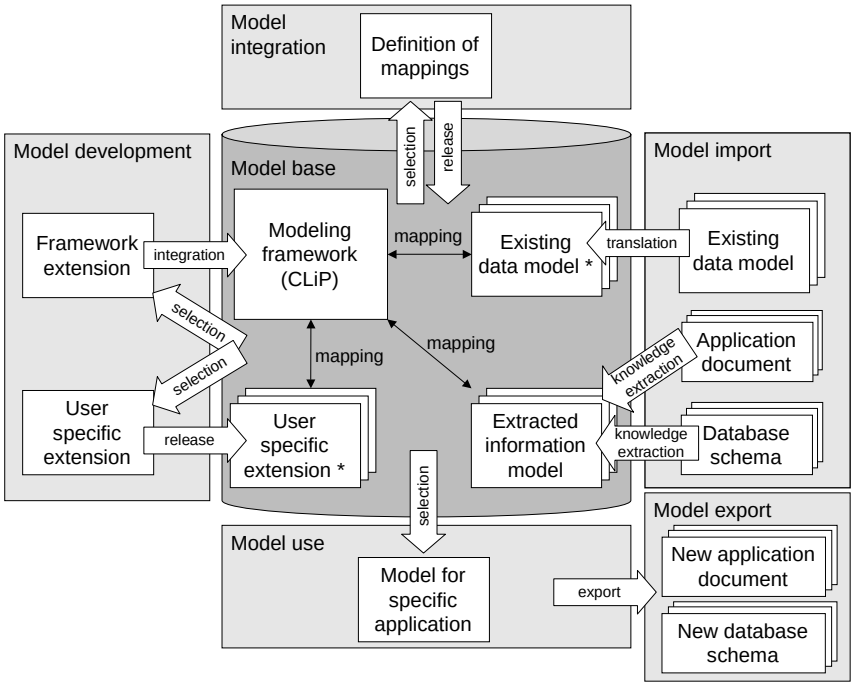

Fig. 12. Tool support for the development of information models.

often implicit and informal—knowledge of domain experts (Abecker et al., 1998). They should thus be considered for *model import*. For the integration of existing data models, a *translation* of their concepts to the ones used in the model base is needed. This translation only refers to the modeling language and not the contents of the model to be imported. The integration of information from application documents and database schemata requires some *knowledge extraction* to derive models of these application resources.

In order to relate the imported models to the model framework, a *model integration* facility is required. The user selects the models to be integrated and those parts of the framework, which are relevant for this integration. Then, a definition of the mappings between the model and the framework can be given. Here, correspondences need to be established between the different information models similar to the ones proposed by Gruner et al. (1998) for the description of relationships between contents of structured documents. The mappings are released to the model base, where they are available together with the framework and the imported and extracted models for further development and use.

The user can select parts from the model base and use them for further *model development*. Here, extensions of the framework and user-specific extensions can be distinguished. The *integration* of framework extensions needs to be done in a consistent manner. User specific extensions can simply be released into the model base. In order to relate them to the framework and the other models in the model base; they need to be integrated in a similar manner as imported models.

For *model use*, parts from the model base can be selected and used for the development and adaptation of an information model for a specific application. This allows a customizing of the models from the library. Often, a further specification of the (conceptual) information models towards implementation models is needed. Here, some *model export* into different modeling and programming languages can provide valuable support functionalities. The modified and exported models can be used as a the basis for the development of support tools or data bases for chemical engineering work.

The development of a modeling tool for information modeling as it is sketched here is an ambitious goal. Solutions and approaches for some of the modules and functionalities exist, others are subject of current research. The model base, model development, and model use can for example be realized on the basis of commercial CASE (computer-aided software engineering) tools. Functionalities are needed for browsing and retrieving existing models, for collaborative development (e.g. multi-user access, release mechanisms), and for model testing (Fikes & Farquhar, 1999). CASE tools provide some facilities for model export; probably, these need to be extended. For the translation of existing data models, description logics can be used since they provide a semantic richness that allows for an almost complete representation of the concepts of widely used information modeling languages (Calvanese, Lenzerini, & Nardi, 1998). Knowledge extraction from application resources is a more complicated task. But an automatic thesaurus generation as proposed by Abecker et al. (1998) can be the basis for a simplified and rather efficient model development based on such information sources. For model integration,

description logics or graph grammars as discussed in Section 3.3.2 can be applied. Still, model translation, import, and integration require high manual specification efforts for the identification of relevant concepts and dependencies. These tasks are modeling tasks that require domain knowledge and can thus not be fully automated.

6. Summary and conclusions

For the improvement of work processes in an application domain and for the development of support tools and software environments, a thorough understanding of these processes together with the handled information is needed. Such an understanding can be obtained by the analysis and formalization of the domain using information models. Numerous models have been developed in the past describing a more or less complete part of the domain of chemical engineering. Most of these models are dealing with product data, some with work processes or documents.

In this paper, some open research issues in information modeling have been discussed. These issues are the dependencies between information and the workflow where this information is used, the relation between data and documents as the main carriers of information, and the integration of the numerous existing data models. One major prerequisite for the analysis of the domain and for solving the discussed problems is a model framework or ontology.

The information model CLiP has been presented as a basis for the development of such a model framework. Here, we want to discuss the actual contribution of CLiP to the solution of the information modeling problems discussed in Section 3 and its potential use for the development of support software.

Within CLiP, modeling elements are included to allow an integrated description of product data, documents, and work processes together with their dependencies. Thus, a formal integration of workflow models and data models as discussed in Section 3.1.2 can be achieved. Another open issue in this context is the integration of work process models on different levels of detail. Here, CLiP does not offer yet a direct solution; this issue is subject of current research in our group.

Documents and data can be described together within CLiP, with relations that can be given between the contents of a document and the actual product data they are representing. Here, the focus is set on description of the document structure, i.e. its syntax. A description of the semantics of a document under different aspects is missing. Document models as they are included in CLiP can be used as a basis for the development of document management systems, but important aspects like the description of configurations of dependent documents with their versions and alternatives are currently missing.

CLiP is a conceptual information model, developed independently from any specific application. Also, its meta model layers and the partial models provide a well defined and extensible structure. Thus, CLiP fulfills the major requirements of a global model to be used as the basis for model integration according to McKay et al. (1996). Still, the integration of CLiP with existing data models is a complex and difficult task. But compared to the STEP approach, the complexity of this task is reduced by the high degree of conceptualization of CLiP. The integration of data models can be simplified by the meta models as it is discussed in Bayer et al. (2000). Experience has shown that standard models like the one of STEP have not gained acceptance in the domain of chemical engineering. Therefore, CLiP has been designed for adaptations and extensions.

In Section 5, we have presented a vision of a tool environment for the support of information modeling on the basis of a conceptual framework. Such an environment can be advantageous for future developments in information and work process modeling and the related software tools. The model framework can serve as a common basis, where all information models can be related to. The exchange of information between tools and between companies can be supported, and still the description of specific, user-dependent information structures is possible. It has been stressed that there is also need for a modeling methodology.

Information models are a prerequisite for the development of specific application tools and information management facilities and for the integration of tools into software environments. But for these software development tasks, information models are required on a different level of conceptualization and formalization than the one provided by CLiP. More detailed models are needed, for example to describe the internal data structures of the tools and their functionalities. These have to be consistent with conceptual information models like CLiP, which has been developed from the perspective of an application expert in order to describe his domain of interest. Therefore, further model development, refinement, and integration are needed on different levels. These modeling issues are currently investigated within the Collaborative Research Center IMPROVE (Marquardt & Nagl, 1998; Nagl & Marquardt, 2001).

Acknowledgements

This work has been supported in part by the DFG (Deutsche Forschungsgemeinschaft) in the Collaborative Research Center IMPROVE (SFB 476). The authors would like to thank M. Eggersmann, M. Nagl, R. Schneider, and L. von Wedel for many fruitful discussions.

References

Abecker, A., Bernadi, A., Hinkelmann, K., Kühn, O., & Sintek, M. (1998). Towards a technology for organizational memories. *IEEE Intelligent Systems, 13*(3), 40–48.

Allen, R. (2000). Workflow: An introduction. In L. Fischer (Ed.), *Workflow handbook 2001. Future strategies, lighthouse point* (pp. 15–38).

Batini, C., Lenzerini, M., & Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys, 18*(4), 323–364.

Batres, R., Naka, Y., & Lu, M. L. (1999). A multidimensional design framework and its implementation in an engineering design environment. *Concurrent Engineering: Research and Applications, 7*(1), 43–54.

Bayer, B., Eggersmann, M., Gani, R., & Schneider, R. (2002). Case studies in conceptual process design. In B. Braunschweig, R. Gani (Eds.), *Software architectures and tools for computer aided process engineering*. Amsterdam: Elsevier, in press.

Bayer, B., Krobb, C., & Marquardt, W. (2001). A data model for design data in chemical engineering—information models. Technical report LPT-2001-15, Lehrstuhl für Prozesstechnik, RWTH Aachen. Online available from: http://www.lfpt.rwth-aachen.de/Publication/Techreport/2001/LPT-2001-15.html.

Bayer, B., & Marquardt, W. (2002). A comparison of data models in chemical engineering. *Concurrent Engineering: Research and Applications*, submitted for publication.

Bayer, B., Schneider, R., & Marquardt, W. (2000). Integration of data models for process design—first steps and experiences. *Computers & Chemical Engineering, 24*(2–7), 599–605.

Bayer, B., Schneider, R., & Marquardt, W. (2001). CLiP—Ein konzeptuelles Rahmenwerk für Produtdatenmodelle. In *GMA-kongress 2001 on automatisierungstechnik im spannungsfeld neuer technologien* (Vol. 1608, pp. 681–688). VDI-Verlag: VDI-Berichte.

Becker, S., Haase, T., Westfechtel, B., & Wilhelms, J. (2002). Integration tools supporting cooperative development processes in chemical engineering. In H. Ehrig, B. J. Krämer, & A. Ertas (Eds.), *Proceedings of the sixth biennial world conference on integrated design and process technology* (IDPT-2002) [CD-ROM]. Pasadena: Society for Design and Process Science.

Bernstein, P. A., Halevy, A. Y., & Pottinger, R. A. (2000). A vision for management of complex models. *SIGMOD Record, 29*(4), 55–63.

Beßling, B., Lohe, B., Schoenmakers, H., Scholl, S., & Staatz, H. (1997). CAPE in process design—Potential and limitations. *Computers & Chemical Engineering, 21*(Suppl.), S17–S21.

Book, N. L., Sitton, O. C., Motard, R. L., Blaha, M., Maia-Goldstein, B., Hedrick, J., & Fielding, J. J. (1994). The road to a common byte. *Chemical Engineering, 101*(3), 98–111.

Brown, J. S., & Duguid, P. (2000). In *The social life of information*. Boston: Harvard Business School Press.

Calvanese, D., de Giacomo, G., Lenzerini, M., Nardi, D., & Rosati, R. (2000). Source integration. In M. Jarke, M. Lenzerini, Y. Vassiliou, & P. Vassiliadis (Eds.), *Fundamentals of data warehouses* (pp. 27–45). Berlin: Springer.

Calvanese, D., Lenzerini, M., & Nardi, D. (1998). Description logics for conceptual data modeling. In J. Chomicki, G. Saake (Eds.), *Logics for databases and information systems* (pp. 229–264). Boston: Kluwer Academic Publishers.

Castano, S., de Antonellis, V., & de Capitani di Vimercanti, S. (2001). Global viewing of heterogeneous data sources. *IEEE Transactions on Knowledge and Data Management, 13*(2), 277–297.

Catarci, T., & Lenzerini, M. (1993). Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems, 2*(4), 375–398.

Claypool, K. T., & Rundensteiner, E. A. (2001). Sangam: Modeling transformations for integrating now and tomorrow. Technical report WPI-CS-TR-01-04, Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA.

Cleetus, K. (1992). Definition of concurrent engineering. Technical report, CERC-TR-RN-92-003, Concurrent Engineering Research Center, West Virginia University. Online available from: http://www.cerc.wvu.edu/CERC-TR-INDEX.htm.

Eggersmann, M., Krobb, C., & Marquardt, W. (2002). Applications of modeling—a case study from process design. In B. Braunschweig, R. Gani (Eds.), *Software architectures and tools for computer aided process engineering*. Amsterdam: Elsevier, in press.

Eggersmann, M., Krobb, C., & Marquardt, W. (2000). A modeling language for design processes in chemical engineering. In A. H. F. Laender, S. W. Liddle, & V. S. Storey (Eds.), *Conceptual modeling—ER 2000. Lecture notes in computer science* (Vol. 1920, pp. 369–382). Berlin: Springer.

Fikes, R., & Farquhar, A. (1999). Distributed repositories of highly expressive reusable ontologies. *IEEE Intelligent Systems, 14*(2), 73–79.

Foltz, C., Killich, S., Wolf, M., Schmidt, L., & Luczak, H. (2001). Task and information modeling for cooperative work. In M. J. Smith, G. Salvendy (Eds.), *Systems, social and internationalisation design aspects of human-computer interaction. Proceedings of HCI international 2001* (Vol. 2, pp. 172–178). Mahwah, NJ: Lawrence Erlbaum.

Gruner, S., Nagl, M., & Schürr, A. (1998). Integration tools supporting development processes. In M. Broy, B. Rumpe (Eds.), *Requirements targeting software and systems engineering. Lecture notes in computer science* (Vol. 1526, pp. 235–256). Berlin: Springer.

Hameri, A.-P., & Nihtilä, J. (1998). Product data management—Exploratory study on state-of-the-art in one-of-a-kind industry. *Computers in Industry, 35*, 195–206.

ISO 10303 (1998). Process engineering data: Process design and process specifications of major equipment (Part 231). Committee Draft ISO TC184/SC4/WG3 N740.

Kappe, F. (1999). Knowledge management: Aufbau von wissenspotentialen in verteilten organisationen. In *Proceedings of the congress VII on workflow & knowledge management* (ONLINE'99).

KBSI (2000). IDEF—i-CAM definition. Online available from: http://www.idef.com/ (accessed April 4, 2002).

Klir, G. J. (1985). *Architecture of systems problem solving*. New York: Plenum Press.

Konda, S., Monarch, I., Sargent, P., & Subrahmanian, E. (1992). Shared memory in design: A unifying theme for research and practice. *Research in Engineering Design, 4*, 23–42.

Mannarino, G. S., Henning, G. P., & Leone, H. P. (1997). Process industry information systems: Modeling support tools. *Computers & Chemical Engineering, 21*(Suppl.), S667–S672.

Mariño, O., Rechenmann, F., & Uvietta, P. (1990). Multiple perspectives and classification mechanism in object-oriented representation. In *Actes 9th European Conference on artificial intelligence* (pp. 425–430). Stockholm.

Marquardt, W., Nagl, M. (1998). Tool integration via interface standardization? In *36. Tutzing symposium der DECHEMA e.V. on informationsverarbeitung in der prozeß-und anlagentechnik* (pp. 95–126). DECHEMA-Monographie.

McKay, A., Bloor, M. S., & de Pennington, A. (1996). A framework for product data. *IEEE Transactions on Knowledge and Data Engineering, 8*(5), 825–838.

McKay, A., & de Pennington, A. (2001). Towards an integrated description of product, process and supply chain. *International Journal of Technology Management, 21*(3–4), 203–220.

Misander, P. K. (2000). A document-oriented model of the workflow in an engineering project. CAPE-NET. Online available from: http://capenet.chemeng.ucl.ac.uk/Website/CAPE_Supersite_Frameset.html (accessed April 24, 2001).

Mylopoulos, J. (1998). Information modeling in the time of revolution. *Information Systems, 23*(3–4), 127–155.

Nagl, M., & Marquardt, W. (2001). Tool integration via cooperation functionality. In *Proceedings of the 3rd European congress of chemical engineering* (ECCE3) (Paper 6-5). Online proceedings available from: http://www.dechema.de/veranstaltung/ecce/cd/toc.htm (accessed April 10, 2002).

Patzak, G. (1982). *Systemtechnik—planung komplexer innovativer systeme*. Berlin: Springer.

Peltonen, H., Pitkänen, O., & Sulonen, R. (1996). Process-based view of product data management. *Computers in Industry, 31*, 195–203.

PIEBASE (1998). Process industries executive for achieving business advantage using standards for data exchange (PIEBASE). Online available from: http://www.posc.org/piebase/ (accessed 4 April 2002).

Pitoura, E., Bukhres, O., & Elmagarmid, A. (1995). Object orientation in multidatabase systems. *ACM Computing Surveys, 27*(2), 141–195.

Polke, M. (1994). *Process control engineering*. Weinheim: VCH.

Preece, P., Ingersoll, T., & Tong, J. (1994). Specification (data) sheets—Picture a database. In *Proceedings of the fourth European symposium on computer aided process engineering*. *IChemE symposium series* (Vol. 133, pp. 467–474).

Reddy, M. P., Prasad, B. E., Reddy, P. G., & Gupta, A. (1994). A methodology for integration of heterogeneous databases. *IEEE Transaction on Knowledge and Data Engineering, 6*(6), 920–933.

Reich, Y., Konda, S., Subrahmanian, E., Cunningham, D., Dutoit, A., Patrick, R., Thomas, M., Westerberg, A. W., & the n-dim group (1999). Building agility for developing agile design information systems. *Research in Engineering Design, 11*(2), 67–83.

Rosman, G., van der Meer, K., & Sol, H. G. (1996). The design of document information systems. *Journal of Information Science, 22*(4), 287–297.

Rumbaugh, J., Jacobson, I., & Booch, G. (1999). *The unified modeling language reference manual*. Reading, MA: Addison-Wesley.

Salminen, A., Lyytikäinen, V., & Tiitinen, P. (2000). Putting documents into their work context in document analysis. *Information Processing and Management, 36*, 623–641.

Schneider, R., & Marquardt, W. (2002). Information technology support in the chemical process design lifecycle. *Chemical Engineering Science, 57*(10), 1763–1792.

Schürr, A., & Zündorf, A. (1996). Specification of logical documents and tools. In M. Nagl (Ed.), *Building tightly integrated software development environments: The IPSEN approach*. *Lecture notes in computer science* (Vol. 1170, pp. 297–323). Berlin: Springer.

Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *The Knowledge Engineering Review, 11* (2), 93–136.

W3C (1997). Extensible markup language (XML). Online available from: http://www.w3c.org/XML/ (accessed 25 April 2002).

Westfechtel, B. (1998). *Models and tools for managing development processes. Lecture notes in computer science* (Vol. 1646). Berlin: Springer.

Zantout, H., & Marir, F. (1999). Document management systems from current capabilities towards intelligent information retrieval: An overview. *International Journal of Information Management, 19*(6), 471–484.