A Process-Integrated Conceptual Design Environment for Chemical Engineering

Matthias Jarke, Thomas List, Klaus Weidenhaupt

Lehrstuhl für Informatik V, RWTH Aachen
Ahornstr. 55, 52072 Aachen, Germany
jarke@informatik.rwth-aachen.de

Abstract. The process industries (chemicals, food, oil, ...) are characterized by -- continuous or batch -- processes of material transformation. The design of such processes, and their mapping to the available equipment (plants composed of production units in which reactions take place), is a complex process that determines the competitiveness of these industries, as well as their environmental impact. In cooperation with researchers and industry from chemical engineering, we have developed the idea to capture and evaluate the experiences gained about process designs in so-called *process data warehouses*. The data sources for such process data warehouses are highly heterogeneous tools, e.g. for conceptual design (termed flowsheeting in chemical engineering), for mathematical simulations of large non-linear differential equation systems, for measurements gained with experimental usage of equipment at small scale or in original size, or even from molecular modeling. The clients of a data warehouse are interested in operational data transfer as well as experience analysis (pattern detection, process mining) and reuse.
Starting from an empirical analysis of the requirements for a process data warehouse, the paper describes the solution architecture we are pursuing, the models driving the approach, and the status of a prototypical implementation we are undertaking. The prototype links commercial components operationally through advanced wrapping techniques, where the workflow is determined by constraint analysis in a logic-based meta model and executed through a process-integrated modeling environment. In the conclusions, we point out what can be learned from this work for conceptual modeling in general.

1 Introduction

Chemical engineering is the combination of physical, chemical, biological, and informational operations on a chemical plant with the aim of transforming input materials in a manner, that a material product with desirable properties concerning type, behavior, and composition results. The chemical engineering process is, besides this main stream analysis, heavily influenced by considerations of cost and time, but also by environmental side effects, such as energy consumption, detrimental side products, water heating or pollution, and the like.

Since 1997, the German national science foundation (DFG) is funding a Collaborative Research Center at RWTH Aachen in order to address a coherent solution to the issue of collaborative computer-supported chemical engineering [NaWe99]. This center, called IMPROVE after the German title acronym, focuses on the early phases of the development and re-engineering of chemical processes, the so-called conceptual design and the basic engineering. Decisions made at this stage are known to fix already 80% of the overall costs for investment and overall operation which typically run to hundreds or thousands of millions of EUROs per plant. The improved interoperability of mathematical simulation tools for this sector alone has been estimated to result in savings of about 5bn EUROs for the European chemical industries. The European process industries have therefore, in parallel to the start of IMPROVE, embarked on a standardization effort for process simulation in the BRITE-EURAM project CAPE-OPEN [Bra*99].

As stated, IMPROVE aims at linking the existing island solutions for specific chemical engineering tasks into a coherent engineering process. It combines, on the one hand, different specialist areas such as chemical industries and their users e.g. in the plastics construction sector, and – on the other hand – different tasks such as methods for individual workplaces with those for management tasks. Figure 1 summarizes the overall research design of IMPROVE, a brief overview follows.

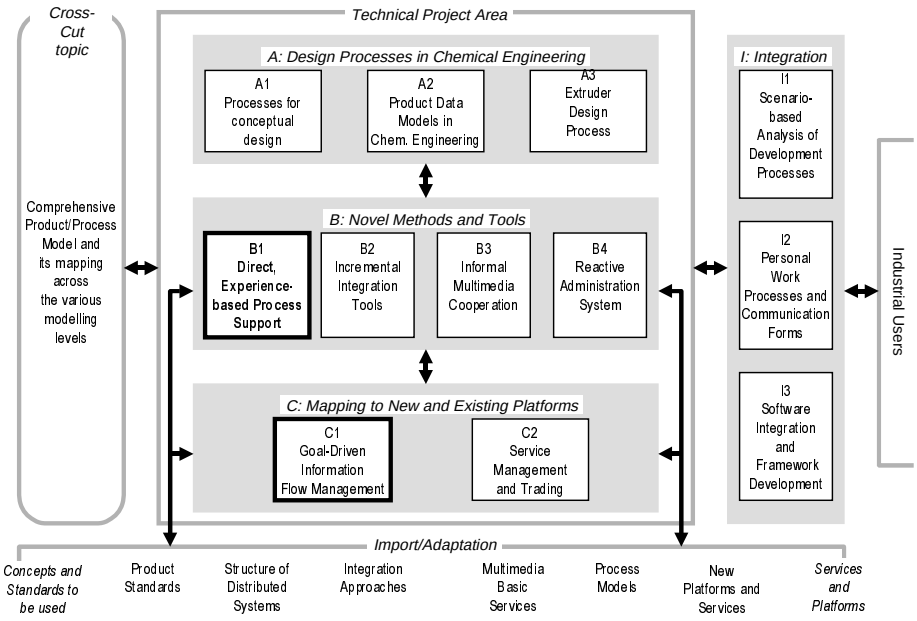

Fig. 1. Structure of the Collaborative Research Center IMPROVE

A pre-requisite is detailed knowledge of individual development processes and their interplay, including interdependencies between product and process. Within IMPROVE, this is the task of the chemical engineering groups (project area A).

This research results in an "idealized" product and process model for cooperative engineering which, however, is neither globally valid nor static but must be customizable to the variety and dynamics of the engineering processes. It must also take into account existing (legacy and new) design and simulation tools. Within project area C, this problem is addressed by developing a so-called process data warehouse and some interoperation service management facilities based on CORBA. Here, we also find the linkage to the above-mentioned CAPE-OPEN effort.

Between these two areas, project area B investigates novel computer science concepts and tools for the different tasks in individual work and cooperation for chemical engineering: direct process support by tracing and reusing developers work expertise; indirect process support by ensuring structure and consistency constraints between the different products; informal, multimedia cooperation to assist awareness and decision making; and project coordination by a flexible and reactive administration system.

The tools supporting these new functionalities, typically on top of wrapped legacy software, must be closely interwoven with each other, without falling into the trap of monolithic integration that has already prevented the success of the CIM movement. The linkage between the A, B, and C layers is instead provided, on the one hand, through metadata management concerning the many aspects of the product and process models (shown on the left of figure 1), on the other by research on an overall framework consisting of actual development scenarios for validation at the level of chemical engineering reality (I1), social science and user interface evaluation (I2), and software architecture (I3).

This paper focuses on *direct process support* for the cooperative development of chemical processes based on *recorded experiences* (project B1 in fig. 1), and on its underlying support by a *process data warehouse* and its meta database, managing the product and process model (project C1). In section 2, we describe these aspects in more detail and point out the central role of flowsheets as a kind of master document in chemical engineering environments. In section 3, the data model of the process data warehouse is presented and linked to recent work in data warehouse and traceability research. Section 4 discusses how engineering environments can be realized on top of such a data model, based on advanced wrapping techniques required for full process integration. These wrapping techniques are demonstrated with the most important new tool for direct process support, a process-integrated flowsheet editor we have developed on top of the widely used commercial tool, VISIO. In section 5, we illustrate how conceptual knowledge, accessible via the process data warehouse, can enhance process-integrated engineering work by method consulting. Section 6 concludes the paper by positioning our work in the conceptual modeling field.

2 Process Integration and Process Data Warehousing

In this section, we first give an overview of our approach to process integration and process data warehousing. Based on an empirical study in a large chemical engineering department, the so-called flowsheet has been identified as the focal concept in such development processes; we therefore briefly describe the state-of-the-art and then point out the advantages of extending flowsheeting with our approach.

2.1 Process-Integrated Design Environments

In contrast to coarse-grained project coordination support at the management level, direct process support aims at fine-grained method guidance for developers at the technical workplace, typically across multiple heterogeneous software tools. Figure 2 gives an overview of the resulting interaction between process-integrated tools (on the right) and process data warehouse (on the left) in our approach.

Direct process support is based on explicit and formal models of the design process. They are composed of working and decision steps performed in the various technical software tools. Just like the early phases in software development, the design of a chemical plant is a highly creative task which cannot be fully and coherently prescribed. Nevertheless, this does not necessarily mean that no process support is possible at all. There exist certain well-understood activities [JaMa96; Döm*96; Lohm98] which occur in many development processes and, therefore, can be generally defined as so- called *process fragments*.

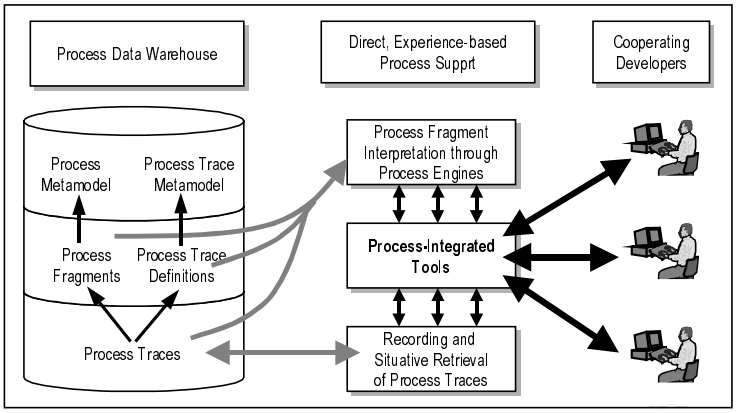

Fig. 2. Basic structure of direct, experience-based process support

Direct process support becomes visible to the user by *process-integrated behavior* of his/her interactive design tools. By process integration [Poh*99] we mean the dynamic adaptation of tools according to the enactment of process fragments. Such process fragements may include: *process automation* by the automated invocation of single tool actions through the process engine, but also *process enforcement and guidance* by restricting user access to certain tool functionality (e.g. by deactivating corresponding menu points) or by highlighting the currently relevant design objects in the user interface of the tool according to the current state of process enactment.

2.2 The Process Data Warehouse

Currently, development processes in chemical engineering are ill-documented, let alone standardized. This complicates the identification and formalization of generally definable, fine-grained process knowledge significantly. Process support has, there-

fore, to start from concrete experiences within and across development processes. A prerequisite is the recording and situated retrieval of experience data as so-called *process traces*.

Process-integrated tools trace their use in the process data warehouse, e.g. by storing the results of tool actions or by recording the user choices among a set of design alternatives. When performing similar processes, the experience knowledge stored in the process traces can be visualized and reused according to the current process situation.

Physically, the three layers of the process data warehouse shown on the left of figure 2 distributed across several subsystems. The upper two layers constitute a *meta database* of knowledge about process definitions and generalized experiences which can be re-used as method advice or consistency control, when activated from the process-integrated tools. We call this the method advisor function of the process data warehouse. In IMPROVE, it is implemented using the ConceptBase system [JGJ*95].

The lower two layers (overlapping in the middle with the meta database) involve massive amounts of trace and product data, often according to task-specific or legacy data models which do not necessarily match the upper-level integrative models perfectly. These data are typically managed in files or, at best, in relational or object-oriented data models.

Direct process support requires a tight integration of the process model interpretation with the design tools used to perform the processes. This poses high demands on the openness of the tools to be process-integrated and causes certain additional effort for the development of suitable wrappers. Our research has, therefore, focused on those parts of the IMPROVE-scenario which can profit most from direct process support.

2.3 The Flowsheet as a Cornerstone of Chemical Engineering Processes

During the development of a chemical process, many information pieces are created which have to be maintained and kept easily accessible. Among these documents, *flowsheets* (graphical representations of function, physics, and structure of a plant), play a prominent role. The importance of flowsheets is not only stressed in various text books about chemical engineering (e.g. [Doug88; Blas97]), but can also be observed in chemical engineering practice.

Indeed, one of the key results of a workshop we conducted in early 1998 with developers and managers from a large chemical engineering department was that the flowsheet reflects in a natural manner the assumptions made by the various stakeholders (chemical engineers, material

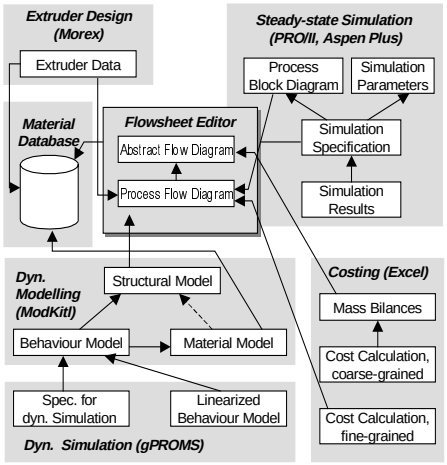

Fig. 3. Role of the flowsheet in the overall IMPROVE scenario

engineers, costing people, safety engineers, managers etc.) about the current state of plant design. The flowsheet hence acts as the main communication medium across several organizational units, and throughout the often decade-long lifecycle of a chemical plant or chemical production process. Moreover, the flowsheet is used as an anchoring point and structuring device for information pieces such as simulation specifications and results, cost calculations, design rationales, safety considerations etc. Managers assess the work progress by the development state of the flowsheet.

The manifold relationships to other information units are illustrated in fig. 3, which depicts the documents and tools used in the IMPROVE demo scenario. They closely correspond to the (unfortunately confidential) ones found in the industrial case study.

2.3.1 Flowsheet tools in chemical engineering practice

In current practice, flowsheets are frequently created using drawing tools or CAD systems. These tools provide no specific support for chemical engineering, nad often confront the user with superfluous functionality.

In competition to these pure drawing tools, dedicated tools for chemical engineering, such as block-oriented simulators (Aspen Plus, PRO/II), have been augmented by flowsheet user interfaces. This pragmatic trend reflects the close relationship between flowsheet design and mathematical models, and provides the user with considerable support – as long as he/she does not have to leave the boundaries of the tool. The enrichment of simulation programs with flowsheet functionality has indeed led to monolithic, hardly maintainable software systems, which rarely provide open interfaces for extensions. As a consequence of such islands of automation, flowsheet information is repeatedly entered manually, e.g. if different simulators are used within one project. Moreover, as flowsheet editors of common simulators do not allow to annotate individual flowsheet elements with, e.g., cost calculations or safety remarks, isolated flowsheet documents emerge in these working environments, too.

2.3.2 The CAPE-OPEN Initiative

These problems have gotten to the point where the monolithic solutions have also led to commercial monopolies : a few vendors control access by chemical companies to progress in the field of chemical engineering research. The CAPE-OPEN project has been a first answer of the chemical industries to these issues. In CAPE-OPEN, the chemical companies BASF, Bayer, DuPont, and ICI, and the oil companies BP and Elf joined to define interoperability standards for chemical engineering software (esp. Simulators) in order to allow interoperation between the tools of the three market-leading vendors as well as to enable the addition of novel methods and tools from small vendors, research groups, and in-house developments.

Due to uncertainties in the underlying software standards, the CAPE-OPEN standard had to be developed in two parallel versions, one in DCOM, the other in CORBA. To manage coherence between the two versions, we had to resort to UML as a higher-level specification, which was developed following a use-case approach [Jar*99]. The interoperation use cases are a nice example of the richness found in the chemical engineering application; we therefore reproduce the hierarchy by which we structured the roughly 160 use cases, in figure 4.

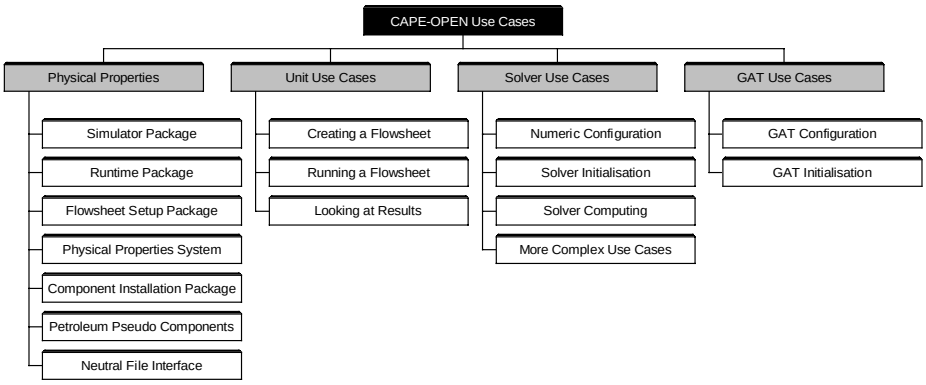

Fig. 4: CAPE-OPEN use case hierarchy for chemical engineering simulation

2.3.3 Flexibility through process integration

The fact, that the flowsheet is involved in manifold development activities, leads to conflicting requirements: from the user's perspective, the flowsheet should be seamlessly integrated into the various process chains (simulation, costing , safety engineering ...) and the corresponding tools. But from a software architecture perspective, tight coupling between tool functionality of different working domains should be avoided in order to remain sufficiently flexible.

Process integration offers the potential to couple different tools more flexibly and to provide high quality support for the user at the same time. Explicit, easily modifiable process fragments guide the user during activities across multiple tools, while the tools themselves only cover a limited scope. In this way, process integration complements data integration mechanisms, which maintain structural consistency between documents of different tools, and component-based approaches such as CAPE-OPEN.

3 The Process Data Warehouse Data Model

The design of the process data warehouse builds on ideas found in data warehousing and software engineering. A major influence has been the separation of logical and conceptual perspectives in source integration for data warehouses, proposed in [CDL*98] and adapted for metadata management in [JJQV99]. For our present purposes, the key argument is that, in highly heterogeneous environments such as chemical engineering, it makes little sense to go for a direct integration of the information sources at the logical level. Instead, as proposed in the Information Manifold project [LeSK95], the information content of these sources should be captured at the level of conceptual modeling and related to an "idealized" *enterprise conceptual model* in order to establish the coverage of the sources, as well as their inter-relationships. IMPROVE follows this approach by adapting a conceptual modeling language proposed for chemical engineering, for the purposes of meta data management and knowledge representation in the process data warehouse.

3.1 The IMPROVE Conceptual "Enterprise" Model

The definition of IMPROVE's conceptual product data model pursues a strategy which is complementary to the idea of process integration. Based on the chemical engineering data model VeDa [MaGG93], the product data model is divided into logical clusters, which represent one distinct activity within the design process [BaSM99]. They can be interpreted as the perspectives of different designers from several technical disciplines on the product data. A meta model expresses commonalties between the individual partial models. Relationships between objects of different partial models are expressed by dependency links. Our current ConceptBase implementation of this "conceptual enterprise model" comprises roughly 150 metaclass definitions. In terms of [CDL*98], this could be considered the conceptual source models. The "logical source models", then, correspond to the data models of individual chemical engineering tools. Note, however, that when re-using legacy tools, their data models will usually cover aspects of more than one conceptual source model.

The partial models *Chemical Process* and *Plant*, which are closely related to the concept of flowsheet, represent two key aspects of the product data model. In the following, we discuss two important requirements which arise from these partial models: the treatment of complex refinement structures and a rich type system of flowsheet components.

3.2 Complex Refinement Structures

Flowsheets are refined across various abstraction levels and depicted in specific representation formats. In the context of the IMPROVE project, we are mainly interested in two of the three commonly used flowsheet variants[1]: the abstract flow diagram (AFD) and the process flow diagram (PFD).

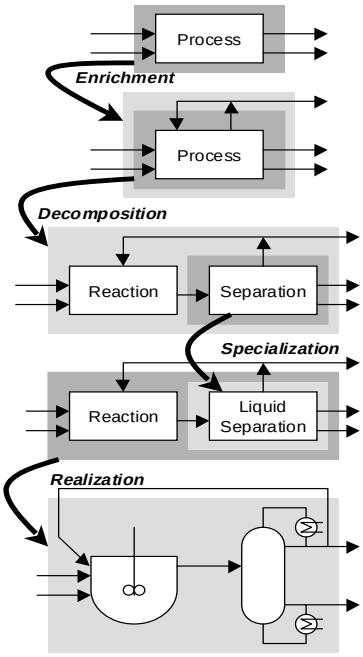

Fig. 5. Types of Flowsheet Refinement

The AFD specifies the overall function of the chemical process and decomposes it into subfunctions and connections between the subfunctions. The AFD does not yet determine the physical realization of a functional element, but only refines to the level of so-called unit operations, e.g. material conversion, separation, or merging. Functional elements are graphically depicted as boxes, material streams as directed arrows. The PFD is used to represent the plant at the equipment level where certain apparatuses and machines realize the unit operations of the functional structure defined in the AFD. In the PFD, abstracted symbols of the equipments are used in graphical representation.

---

[1] The pipe-and-instrumentation flow diagram is mainly used for the detailed dimensioning of plant equipment which is beyond the early phases of process design, i.e. the scope of the IMPROVE project.

Fig. 5 illustrates the stepwise refinement of the process structure in the AFD and its rough physical realization in the PFD. Even this small example gives an impression of the emerging complex refinement structures, which are characterized by manifold refinement relationships (enrichment, decomposition, specialization, realization) between flowsheet parts.

A „correct“ refinement has to consider a set of consistency constraints, e.g. balancing rules for the in- and out-streams or type consistency between refined and the refining flowsheet parts. A flowsheet tool supporting the refinement of flowsheets has to assure such consistency constraints. To support this combination of complex refinement tasks with knowledge capture in the meta database implementation, we added a part-of abstraction to the conceptual object meta meta model, combined with the elementary distinction between process steps and streams (physically corresponding to chemical plant devices and connections between them) which is typical for the flowsheet. The resulting meta meta model (which is formally represented in logic) is graphically depicted in figure 6.

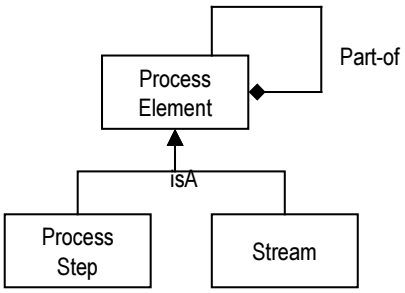

Fig. 6. Meta meta model for refinable flowsheet

3.3 Rich and Extensible Type System for Flowsheet Components

The organization of flowsheet components in an expressive type system is an indispensible prerequisite for guiding consistent flowsheet refinement. However, the richness and the rapid growth of knowledge goes beyond the abilities of typical object-oriented type systems : there are ten thousands of conceivable process step types and millions of possible stream types. Examples of refinement rules include:

- “if a process element has the same materials in its input and output streams, a feedback loop should be designed for them”;
- “if you add a reaction component, and the output of this reaction is a mixture of different materials, a separation unit should be added behind the reactor”;
- “if the boiling temperature of the ingredients within a mixture differ pairwise by more than 10°C, the separation is best implemented by a destillation column”.

The type system has to specify compatibility between flowsheet components along several dimensions (e.g. „U can be used as a *specialization* of V“ or „X can occur as part of a *decomposition* of Y“). It has to provide a semantically rich characterization of individual flowsheet components (e.g. the third rule above).

The type system must be extensible since knowledge about available flowsheet components and their properties is still rapidly evolving. Furthermore, the chemical engineer should be empowered to define his/her own flowsheet components and characterize them semantically in order to promote their reuse in future projects. User-defined flowsheet components emerge, e.g., from parametrizing generic flowsheet components or from aggregating complex components from simpler ones (e.g. by sequencing a set of destillation columns and heat exchangers).

Clearly, this richness cannot be hardcoded in the flowsheet editor. Instead, we encode declarative knowledge in the meta database of the process data warehouse. The data model of the flowsheet editor just knows about the most obvious and stable refinements of the meta meta model from figure 7, and about the existence of subtypes with different (user-definable) graphical symbols. The semantics of these subtypes, as well as rules and methods for their usage, is delegated via calls to the evolving amount of knowledge in the meta database of the process data warehouse, where it is accessed via querying and constraint analysis techniques. Considerable effort has been spent to find effective mechanisms for handling such large hierarchies in meta-data repositories based on deductive object-oriented [BaJa99] and/or description logic data models [Satt98]. The present implementation only covers some of these results.

4 Process Integration of the Flowsheet Editor

While the creation of wrappers for simple tool invocation/stoppage and basic data transfer is relatively simple, it only constitutes a rough approximation to the kind of process integration we envisioned in section 2. Fortunately, full process integration is only necessary for those tools in the chemical engineering process which have either complex tool-internal user interaction, or complex process interaction with several other tools. From the previous discussions, it should be obvious that the flowsheet editor is a prime example of such a tool. We therefore focused our efforts on process-integrating a flowsheet editor first, and to demonstrate process integration with other widely used tools (such as Excel spreadsheets) which are known from our practice surveys to interact a lot with the flowsheet.

4.1 Architecture and Basic Functionality

The development of a completely new flowsheet editor is a formidable task, and the result is unlikely to be able to compete with the rapid advances in graphical design tools. Our process-integrated flowsheet editor has therefore been realized according to the a-posteriori philosophy of IMPROVE, on top of an existing tool.

None of the commercially available flowsheet tools fulfills the requirements concerning flowsheet refinement and extensible type systems for flowsheet components. Hence, the existence of open interfaces, which allow the addition of the above-mentioned functionality (besides the process integration), became the most important criterion during the choice for a tool to be integrated. We finally selected VISIO [Visi98], a widely used tool for creating technical drawings. VISIO's strengths lie in its comprehensive and extensible symbol libraries and its add-on mechanisms based

on COM interfaces. These interfaces allow external extensions a fine-grained access to VISIO‘s internal object model.

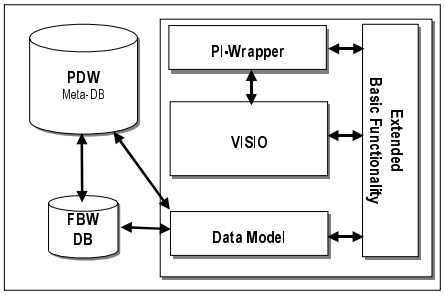

Fig. 7. Architecture of the Flowsheet Editor

Fig. 7 depicts the coarse-grained architecture of the flowsheet editor. Visio is the center of the architecture and provides the essential GUI-functionality for constructing flowsheets. The data model layer extends VISIO’s object model by refinement structures and stores the flowsheet persistently in a (relational) DBMS. VISIO’s GUI functionality and the data model layer are used by the extended basic functions which realize operations such as creating a new refinement level or navigating between different refinement levels. Figure 9 shows a a refinement within the same drawing pane. This is done by expanding the right flowsheet node into a sub-drawing, thus automatically forcing the user to consider the balancing rules for inputs and outputs.

4.2 Wrapper for Process Integration

For integrating the flowsheet editor with the process model interpretation, we have adapted the PRIME approach [Poh*99]. Process integration in PRIME is based on explicit models of both the design processes and the tools to perform the processes. Process and tool models are integrated within the so-called environment model.

The interpretation of environment models enables the tools to adapt their behavior to the process definition and the current process state. A generic implementation framework defines the basic structure of a process-integrated tool and provides components for environment model interpretation and interaction with the process engine.

Originally meant for the development of *new* process-integrated tools, the implementation framework can also be used as a *wrapper* for *existing* tools. To be process-integratable, a tool has to offer a number of application programming interfaces (APIs): (1) *service invocation and feedback*: invoke individual services in a (running) tool instance and report the service results back to the process engine; (2) *command insertion*: enhance the user interface of the tool by new commands and bind them to externally defined process fragments; (3) *product display*: highlight currently relevant design objects in the user interface of the tool; (4) *selectability*: restrict the selection of product parts and activation of commands; (5) *selection notification*: notify the process engine about product and command selections in the tool.

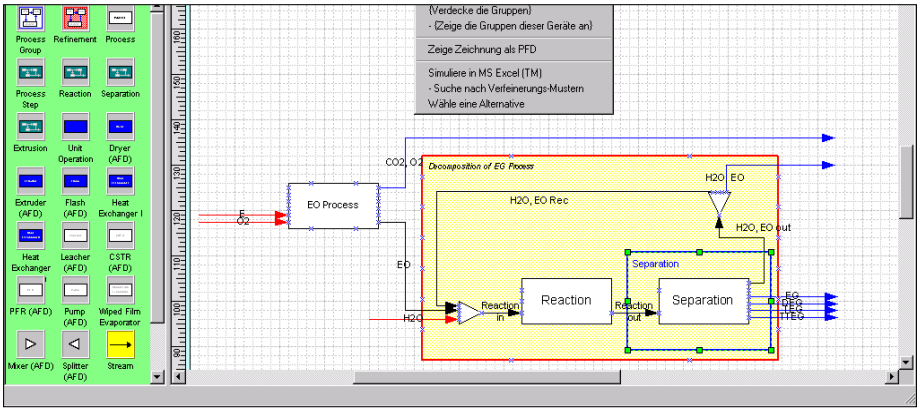

Fig. 8. Screenshot from flowsheet editor, demonstrating refinement on top of VISIO

The PRIME implementation framework uses these APIs when interpreting the relevant parts of the integrated tool and process models. It thereby brokers the interaction between the process engine and the tool. In our experiments, we achieved an almost complete process integration of VISIO. Only the process-sensitive restriction of tool functionality caused some problems since VISIO offers a large number of menu sets which have to be maintained by the PRIME-Framework.

4.3 Tool Data Model

The data model of the flowsheet editor has been designed in UML and implemented in C++ (Fig. 9). It is closely related to the partial models *Chemical Process* and *Plant* of the conceptual IMPROVE product data model. Its design was also influenced by emerging data exchange standards in chemical engineering (e.g. the process data exchange interface standard PDXI in the STEP context [DaGo95]).

The connectivity between flowsheet elements is defined at the level of abstract classes (*ProcessDevice*, *Port*, *Connector*, *Stream)*, since it shares the same structure in both ADFs and PFDs. Flowsheet elements can be aggregated to *ProcessGroup*s. ProcessGroups can be refined by other ProcessGroups. This enables the uniform definition of *n:m* relationships of different refinement types (decomposition, enrichment, specialization, realization) between flowsheet elements. As an essential prerequisite for extending the type system of flowsheet elements, only its abstract classes are directly defined in the data models (*Process*, *ProcessStep*, *Unit-Operation*, *ProcessEquipment*). As mentioned earlier, information about concrete flowsheet elements and their properties as well as compatibility constraints between flowsheet elements is maintained in the meta database of the process data warehouse which is consulted by the flowsheet editor at runtime.

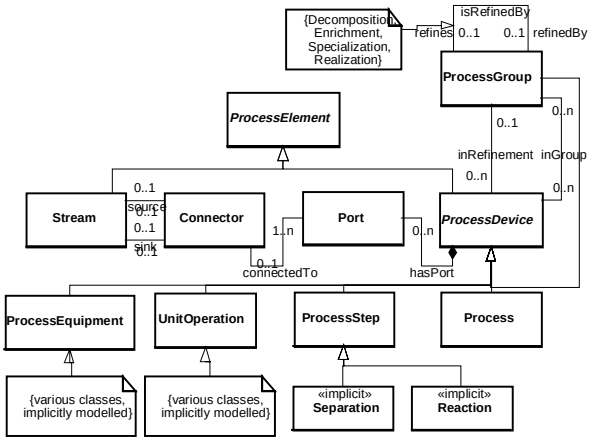

Fig. 9. Data model of the flowsheet editor

4.4 Example of a Tool-spanning Process Fragment

Fig. 10 shows the enactment of a process fragment involving multiple tools. In the VISIO-based flowsheet editor (left), the user has selected the refinement of a process group (step 1). The menu shows all tool actions as well as process fragments, which can be applied to the selected process group according to its type and the process definition. The user chooses the menu item *Simulation in MS Excel* (2). As this menu item is not related to an action provided by the flowsheet editor, a corresponding process fragment is enacted by the process engine (3). According to the definition of the process fragment, the process engine invokes Excel (right) and creates a spreadsheet which calculates the mass flow in the streams of the selected process group.

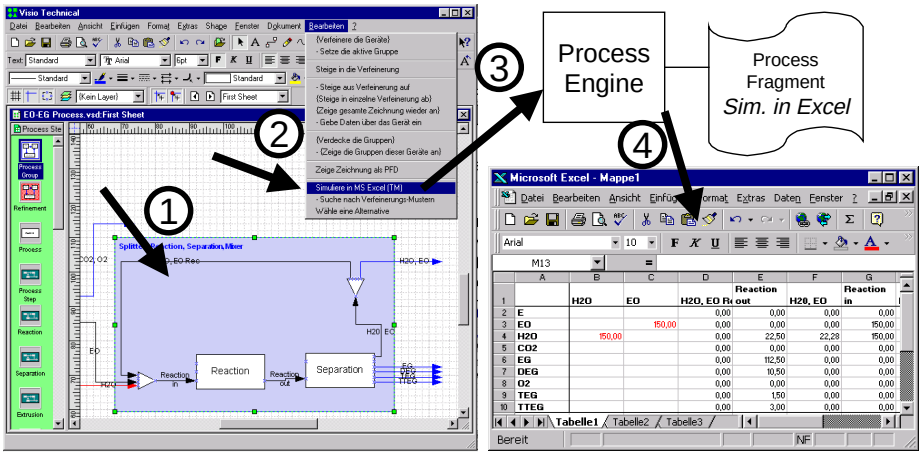

Fig. 10. Process fragment-guided tool integration

5 The Process Data Warehouse as Method Advisor

The method advisor functionality gives engineering tools such as the flowsheet editor access to conceptual knowledge about the chemical engineering domain stored in the metadata repository of the Process Data Warehouse.

To use the method advisor functionality, a tool has to ask the Process Data Warehouse for guidance in a specific situation given through an object and a method the tool wants to perform on it. The Process Data Warehouse now has to collect the necessary information to answer the request, i.e. further data found in the calling tool or in other tools and databases, e.g. the material database. After calculating the request based on the logical rules stored in the meta database, the Process Data Warehouse has to present the result in a suitable way. In the case of the flowsheet editor the request would consist of a flowsheet element selected by the user and a function like "refine" or "specialize". As shown in section 3, additional information like the input and output streams of a process element, found in the flowsheet editor itself, or like the boiling temperature of the participant substances, found in the material database, could be needed. For the flowsheet editor a suitable result would be a set of process fragments, or one choice context combining several fragments.

Fig. 11 shows the method advisory part of the Process Data Warehouse in combination with the flowsheet editor and a material database. The meta database stores the partial models described in section 3, a set of queries defined on these partial models to answer the requests, and a set of meta-queries that represent the additional information needed to answer a specific query. The database trader stores what data can be retrieved from which tool or database and describes how to do that. To capture process context of knowledge reuse, a representation of the different process fragments known to the process engine is also contained in the data warehouse.

To collect the additional flowsheet elements and substance data we have developed CORBA wrappers for the participant tools. These wrappers are compliant with the standards defined in the CAPE-OPEN project.

We illustrate the interplay between the method advisor functionality of the process data warehouse and the enactment of a process fragment involving multiple tools by a short example. In Fig. 12 the chemical engineer has selected the process step „Glycole Separation" (step 1) whose purpose it is to separate a mixture of different glycole types (EG[2], DEG[3], TEG[4], TTEG)[5] produced in a process step before. The chemical engineer wants to realize the „Glycole Separation" by a set of separation apparatuses, but does not know which ones to use and how to connect them. Instead of using the generic, but unspecific refinement function provided by flowsheet editor itself, the chemical engineer invokes (via the interface of the flowsheet editor) the method advisor component of the Process Data Warehouse (2).

[2] Ethylene glycol
[3] Diethylen glycol
[4] Triethylene glycol
[5] Tetraethylene glycol

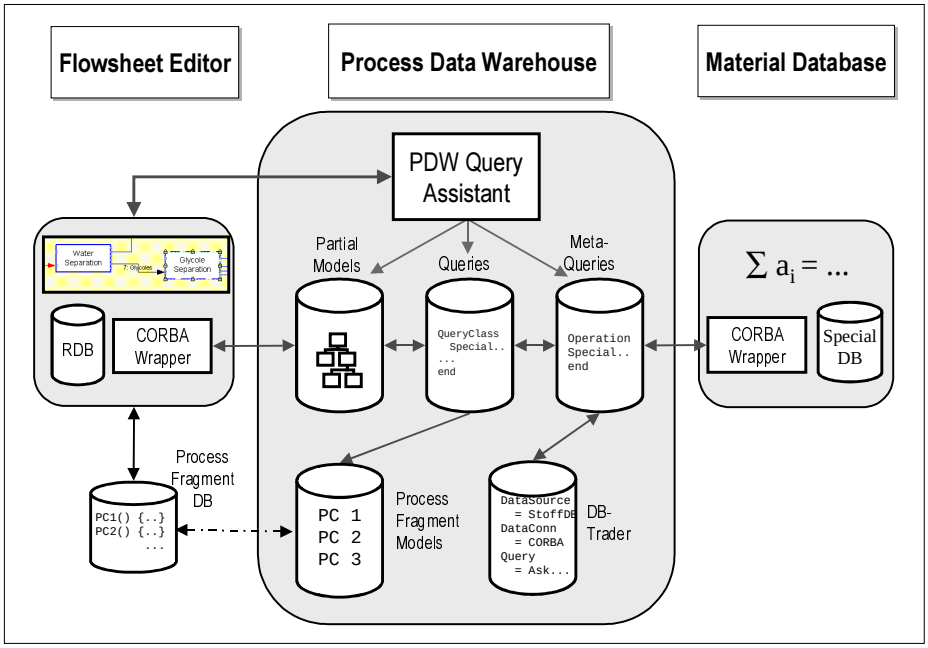

Fig. 11. Architecture of the method advisory function

For answering the query, the query assistant of the method advisor first retrieves all additional information required for analysing the current situation at hand from various data sources. In this case, the query assistant must first determine all input streams of the process step to be refined from the flowsheet database and then retrieve the relevant material data (here: boiling points) of the chemical components contained in the input streams from a material database (4). In our example, the current situation is evaluated to „Separation of a component mixture with sufficiently different boiling points“. The query assistant then retrieves possible process fragments which provide more specific procedural support for handling the current situation.

The resulting process fragment is activated by the process engine (5). According to the process fragment definition, the process engine suggests the refinement of the „Glycole Separation“ by a sequence of three destillation columns (6). However, for determining the sequence in which the destillation columns are connected, it is important to assess the fraction of each component in the input stream. For this purpose, the process engine automatically creates an Excel spreadsheet which calculates the mass flow of each component in the input stream of the separation unit to be refined (7). According to the knowledge gained from this simulation, the chemical engineer can decide how to connect the different destillation columns. The resulting flowsheet is shown at the bottom of Fig. 12 (8).

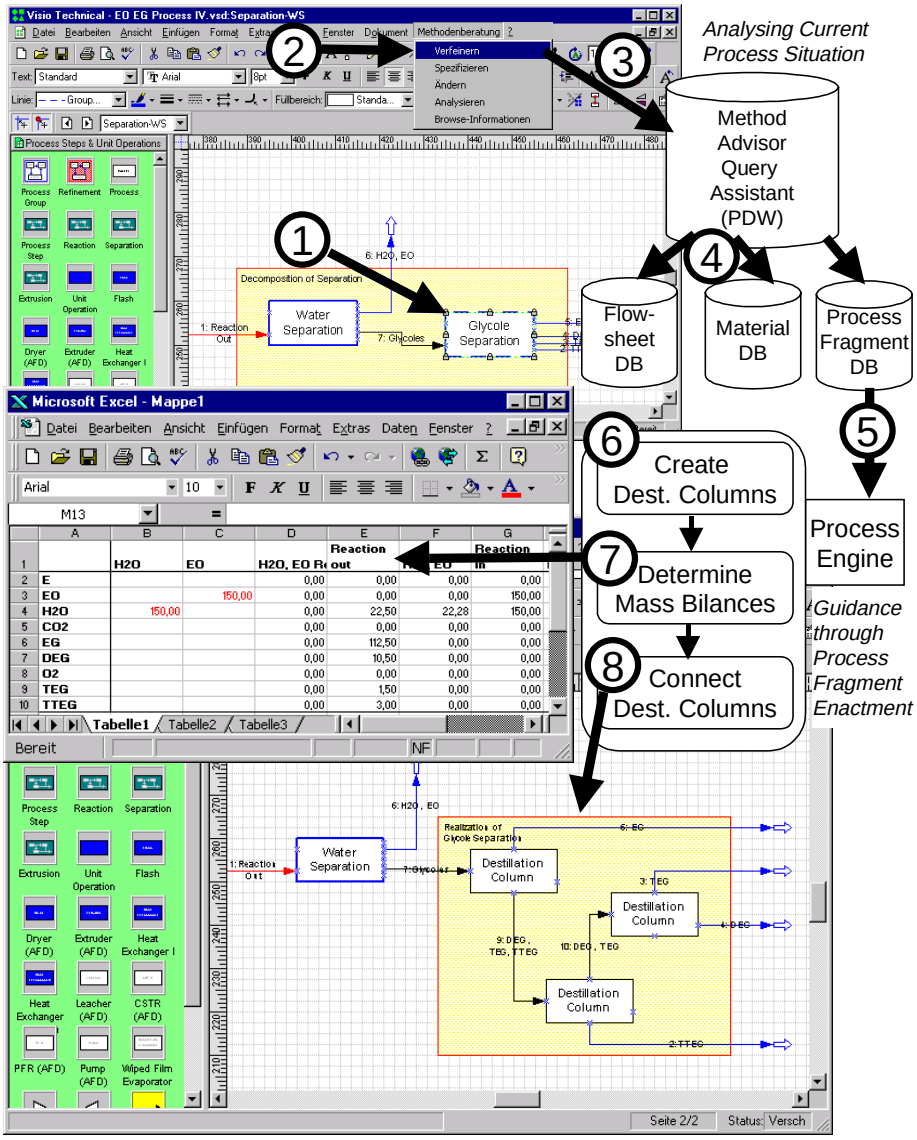

Fig. 12. Method Advisory enacting a tool spanning process fragment

6 Conclusion and Outlook

The chemical engineering domain brings out some problems in large-scale conceptual modeling which are perhaps less clear in the more widely discrete domains

such as mechanical or software engineering. In this paper, we focused on the question how to bring knowledge to the engineering workplace as directly as possible (i.e. in a process-integrated manner) even if this knowledge is complex and undergoes continuous change. Our solution to this problem was the combination of a framework for process integration of engineering tools, with a knowledge-based repository. Simplified versions of the data model and flowsheet editor reported here are being implemented for production usage in the company where we conducted the initial requirements workshop. From their usage, we are expecting experimental data concerning the actual potential of our approach.

The metadata provided by the process data warehouse also allow the control of data exchange and transformation between different tools, and the targeted capture and visualization of trace data as a basis for creating the kind of method knowledge which has been used in our examples. This explicit handling of traces is the major focus of our present work; it requires, in particular, the design of a process query language in order to selectively visualize and abstract traces captured by the PRIME environment across multiple tools [DöPo98].

Moreover, there is an additional complication in process data warehouses. An empirical study of requirements traceability in the American software industries [Rame98] showed that process capture is not limited to conceptual objects. Equally important is the representation of how this conceptual knowledge is grouped in documents, as documents are the unit of management control in engineering processes; this is further augmented by the capture of stakeholder roles, both with respect to content (the stakeholder as a source of requirements or responsible for a content) and documents (the stakeholder as owner, reader, ... of documents). The expanded meta meta model of the process data warehouse we aim to support is shown in fig. 13. This figure also shows how this meta meta model will be used to link the contributions of projects B1-B4 from figure 1 within the overall IMPROVE effort.

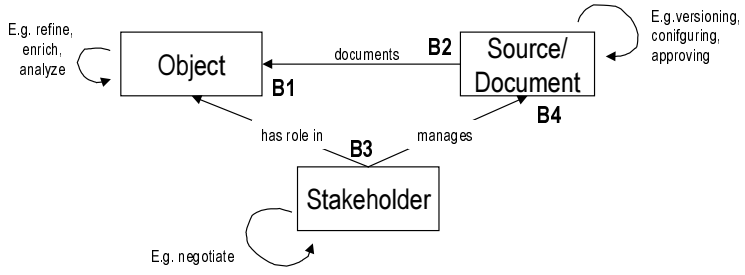

Fig. 13. Meta meta model of process data warehouse

References

[BaJa99] Baumeister, M., Jarke, M.: Compaction of large class hierarchies in databases for chemical engineering. Proc. BTW 99 (Freiburg, Germany), Springer-Verlag 1999, 343-361.

[BaSM99] Bayer, B., Schneider, R. und Marquardt, W.: Product Data Modeling for Chemical Process Design. Proc. European Concurrent Engineering Conference, Erlangen, Germany, April 1999.

[Blas97] Blass, E.: Entwicklung verfahrenstechnischer Prozesse - Methoden, Zielsuche, Lösungssuche, Lösungsauswahl. Springer Verlag Berlin Heidelberg, 1997.

[Bra*99] Braunschweig, Jarke, M., Köller, J., Marquardt, W., v.Wedel, L.: CAPE-OPEN – experiences from a standardization effort in chemical industries. Proc. Intl. Conf. Standardization and Integration in Information Technology (SIIT 99), Aachen, September 1999.

[CDL*98] Calvanese, D., deGiacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Information integration – conceptual modeling and reasoning support. Proc. CoopIS 98, New York 1998, 280-291.

[DaGo95] Dalton, C.M., Goldfarb, S.: PDXI, a Progress Report. In: Proc. CHEMPUTERS Europe II Conference, Noordwijk, Niederlande, Oct. 1995.

[Döm*96] Dömges, R., Pohl, K., Jarke, M., Lohmann, B. und Marquardt, W.: PRO-ART/CE - An Environment for Managing Chemical Process Simulation Models. In: Proc. 10th European Simulation Multiconference, Budapest, Hungary, 1996, S. 1012-1016.

[DöPo98] Dömges, R., Pohl, K.: Adapting traceability environments to project-specific needs. Comm. ACM 41, 12 (1998), 54-62.

[Doug88] Douglas, J.: Conceptual Design of Chemical Processes. McGraw-Hill, 1988.

[JGJ*95] Jarke, M., Gallersdörfer, R., Jeusfeld, M.A., Staudt, M., and Ehrer, S.: ConceptBase- a deductive object base for meta data management. J. Intelligent Information Systems, 4(2):167-192, 1995.

[JJQV89] Jarke, M., Jeusfeld, M.A., Quix, C., Vassiladis, P.: Architecture and quality of data warehouses. Proc. CAiSE 98, Pisa, Italy.

[JaMa96] Jarke, M. und Marquardt, W.: Design and Evaluation of Computer-Aided Process-Modeling Tools. Proc. Intl. Symp. Intelligent Systems in Process Engineering (ISPE 95), Snowmass, Co, 1995, AIChE Press 1996..

[Jar*99] Jarke, M., Tresp, Ch., Becks, A., Köller, J. und Braunschweig, B.: Designing Standards for Open Simulation Environments in the Chemical Industries: A Computer-Supported Use-Case Approach. Proc. 4th Intl. Conf. Systems Engineering (INCOSE 99), Brighton, UK, 1999.

[LeSK95] Levy, A., Srivastava, D., Kirkk, T.: Data model and query evaluation in global information systems. J. Intelligent Information Systems 5, 2 (1995), 121-143.

[Lohm98] Lohmann, B.: Verfahrenstechnische Modellierungsabläufe. Dissertation, RWTH Aachen, VDI Verlag Düsseldorf, Fortschritts-Berichte VDI, Reihe 3, Nr. 531, 1998.

[MaGG93] Marquardt, W., Gerstlauer, A. und Gilles, E.D.: Modeling and Representation of Complex Objects: A Chemical Engineering Perspective. In: Proc. 6th Intl. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Edinburgh, Scotland, 1993, S. 219-228.

[McJo88] McGuire, M.L. und Jones, J.K.: Maximizing the Potential of Process Engineering Databases. In: AIChE Annual Technical Meeting, Houston, TX, USA, 1988.

[NaWe99] Nagl, M. und Westfechtel, B. (Hrsg.): Integration von Entwicklungssystemen in Ingenieuranwendungen. Springer-Verlag 1999.

[Poh*99] Pohl, K., Weidenhaupt, K., Dömges, R., Haumer, P., Jarke, M., Klamma, R.: PRIME: Towards Process-Integrated Environments. To appear in: ACM Transactions on Software Engineering and Methodology.

[Rame98] Ramesh, B.: Factors influencing requirements traceability practice. Comm. ACM 41, 12 (1998), 37-44.

[Satt98] Sattler, U.: Terminological Knowledge Representation Systems in a Process Engineering Application. Dissertation, RWTH Aachen, 1998.

[Visi98] Visio Corp: Developing VISIO solutions, version 5.0. Seattle, Wash., 1998.