Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers and Chemical Engineering 29 (2004) 93–111
Computers & Chemical Engineering
www.elsevier.com/locate/compchemeng

A management system for dynamic and interorganizational design processes in chemical engineering☆

Markus Heller[a1], Dirk Jäger[a2], Marcus Schlüter[b], Ralph Schneider[c], Bernhard Westfechtel[a]*

[a] *RWTH Aachen, Lehrstuhl für Informatik III, D-52056 Aachen, Germany*
[b] *RWTH Aachen, Lehrstuhl für Informatik V, D-52056 Aachen, Germany*
[c] *RWTH Aachen, Lehrstuhl für Prozesstechnik, D-52056 Aachen, Germany*

Received 22 August 2003; received in revised form 17 February 2004; accepted 20 July 2004
Available online 11 September 2004

Abstract

Design processes in chemical engineering are hard to manage. The design process is highly creative, many design alternatives are explored, and both unexpected and planned feedback occurs frequently. Thus, it is inherently difficult to manage the workflow in design processes, i.e., to coordinate the effort of experts working on tasks such as creation of flow diagrams, steady-state and dynamic simulations, etc. Conventional project and workflow management systems support the management of design processes only to a limited extent. In contrast, the management system AHEAD is designed specifically for dynamic design processes. In addition, AHEAD supports the management of interorganizational design processes. A subprocess may be delegated to a contractor, which receives information only about those parts of the overall process that are relevant for the contract. The management systems of the client and the contractor are coupled at run time such that both systems maintain up-to-date state information. The contract may be changed at any time according to the rules of a pre-defined change protocol. To validate this approach to the management of dynamic and interorganizational design processes, the AHEAD system has been applied to a comprehensive case study, namely the conceptual design and basic engineering of a chemical plant for producing polyamide 6.

Keywords: Design process; Project management; Workflow management; Delegation; Polyamide 6

1. Introduction

Design processes in chemical engineering are hard to manage. Since design processes are highly creative, they can rarely be planned completely in advance. Rather, planning and execution may have to be interleaved seamlessly. In the course of the design process, many design alternatives are explored which are mutually dependent. Furthermore, design proceeds iteratively, starting from sketchy, coarse-level designs to detailed designs which are eventually needed for building the respective chemical plant. Iterations may cause feedback to earlier steps of the design process; it may be necessary to revoke inadequate design decisions. Finally, design involves cooperation among team members from different disciplines and potentially multiple enterprises, causing additional difficulties concerning the coordination of the overall design process.

Technical tools such as flowsheet editors, simulators for steady-state and dynamic simulations, etc. are crucial aids for effectively and efficiently performing design tasks. In addition, *managerial tools* are required which address the coordination of design processes. In fact, such tools are crucial for supporting *business decision making* (Heller and Westfechtel, 2004). In the course of the design process, many decisions

☆ This work was carried out in the Collaborative Research Center 476 IMPROVE, which is funded by the Deutsche Forschungsgemeinschaft (DFG).

* Corresponding author.
[1] Tel.: 49 2418021311; fax: 49 2418022218.
[2] Present address: Bayer AG, Cologne, Germany.
E-mail addresses: heller@i3.informatik.rwth-aachen.de (Markus Heller), jaeger@i3.informatik.rwth-aachen.de (Dirk Jäger), schlueter@i5.informatik.rwth-aachen.de (Marcus Schlüter), schneider@lpt.rwth-aachen.de (Ralph Schneider), bernhard@i3.informatik.rwth-aachen.de (Bernhard Westfechtel)

0098-1354/$ – see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2004.07.025

have to be made concerning the steps of the chemical process, the relationships among these steps, the realization of chemical process steps by devices, etc. To perform these decisions, design alternatives have to be identified and elaborated, and the respective design tasks have to be coordinated regarding their mutual interfaces and dependencies. To support business decision making, managerial tools must provide chief designers with accurate views of the design process at an adequate level of granularity, offer tools for planning, controlling, and coordinating design tasks, thereby taking care of the dynamics of design processes.

Currently, many management tools are available which, unfortunately, often have not been developed for supporting complex and dynamic design processes. In fact, most of these tools are generic in the sense that they can be applied to any kind of business process (there are a few process support tools designed for chemical engineering such as, e.g., n-dim (Subrahmanian, Konda, Reich, Westerberg, & the N-dim group, 1997) and KBDS (Bañares-Alcántara & Lababidi, 1995). For example, *project management systems* (Kerzner, 1998) assist managers in project planning and control, assuming that the project can be represented by a partially ordered set of activities. However, iteration and feedback cannot be represented in project plans. Furthermore, important information is missing concerning, e.g., the products of design tasks such as, e.g., different kinds of flow sheets and simulation models. To some extent, project management systems are useful for high-level planning and control at the level of milestones. But to support decision making effectively, other sources of information have to be exploited, as well.

Workflow management systems (Lawrence, 1997) have been developed for supporting routine processes performed, e.g., in banks, insurance companies, administrations, etc. A workflow management system manages the flow of work between participants, according to a defined procedure consisting of a number of tasks. It coordinates user and system participants to achieve defined objectives by set deadlines. To this end, tasks and documents are passed from participant to participant in a correct order. Moreover, a workflow management system may offer an interface to invoke a tool on a document either interactively or automatically. Workflow management systems differ from project management systems since they address fairly detailed execution support rather than only high-level planning. Their most important restriction is limited support for dynamic design processes. Many workflow management systems assume a statically defined workflow that cannot be changed during execution. This assumption does not match the characteristics of design processes, which are highly creative, dynamic, and iterative, and therefore cannot be controlled by a static workflow defined in advance. Though this problem has been recognized in the workflow community Georgakopoulos, Prinz, & Wolf, 1999, so-called adaptive workflow management systems provide at best partial solutions (e.g., based on exceptions, which, however, have to be pre-defined). Still, many of the commercial tools hardly address the problem of evolving workflows.

The management system AHEAD (adapatable and human-centered environment for the management of design processes) (Jäger, Schleicher, & Westfechtel, 1999, Nagl, Westfechtel, & Schneider, 2003) addresses the challenge of supporting dynamic engineering design processes. It has been developed in the context of the long-term research project IMPROVE (Nagl & Marquardt, 1997) which is concerned with models and tools for design processes in chemical engineering. AHEAD equally covers products, activities, and resources and therefore offers more comprehensive support than project or workflow management systems. Moreover, AHEAD supports seamless interleaving of planning and execution—a crucial requirement which workflow management systems usually do not meet. Design processes are represented by *dynamic task nets*, which may evolve continuously throughout the execution of a design process. Dynamic task nets include modeling elements specifically introduced for design processes, e.g., feedback relationships for iterations in the design process which cannot be represented in project plans. This way, AHEAD improves business decision making since it offers a more natural, realistic, and adequate representation of design processes.

In a previous paper (Nagl et al., 2003), we focused on the management of design processes within one organization. In particular, we assumed that all management data are stored in a central database which can be accessed by all users. This assumption breaks down in the case of *interorganizational design processes*, which we will investigate in this paper. Each of the participating organizations requires a view on the overall design process which is tailored to its needs. In particular, it is crucial to provide for information hiding such that sensitive data are not propagated outside the organization. To support the management of interorganizational design processes, we have developed an approach which is based on *delegation*. A subprocess may be delegated to a contractor, passing only those data which are relevant for the contract. Both the client and the contractor use their own instances of the management system, which maintain their data in local databases. The management systems are coupled at run time by exchanging state information. In this way, each of the participating organizations maintains an up-to-date-view of the relevant part of the overall design process. Contracts may be changed at any time according to a pre-defined change protocol which requires mutual agreement to make the change valid. To validate our approach, we have applied the AHEAD system to a comprehensive case study, namely the conceptual design and basic engineering of a chemical plant for producing polyamide 6.

The rest of this paper is organized as follows. Section 2 presents the case study and describes the features of design processes from which requirements to their management are derived. Section 3 introduces the AHEAD system and provides an overview of our delegation-based approach to the management of dynamic and interorganizational design processes. Section 4 elaborates on this approach, which is introduced step by step by an example drawn from the case study.

Section 5 describes the realization of the AHEAD system. Section 6 discusses related work, and Section 7 concludes the paper.

2. Interorganizational design processes

2.1. Design processes in chemical engineering

The life cycle of a chemical design process ranges from the definition of a design objective to the decommissioning of the plant as shown in Fig. 1 (developed within the Global CAPE-OPEN project (Braschweig & Gani, 2002)).

The performed activities (e.g., conceptual process design) are assigned to roles, departments, or companies (resources). The temporal order of the activities is given by the control flow (solid lines). The characteristics of design processes in (chemical) engineering are described in more detail in Nagl et al. (2003). In this paper we will focus on the aspect of interorganizational collaboration within a design project. From this organizational point of view, different models of collaboration can be classified. In most cases, the business unit and the operating company will belong to the same enterprise. Depending on the license owner of the chemical process, the following cases can be distinguished:

(1) All roles are part of one company.
(2) The chemical process is owned by the operating company. Detailed process design and the construction of the plant is carried out by external contractors.
(3) The chemical process is owned by the construction company. Most of the activities during the basic and detail engineering are performed by the construction company.

Besides these cases, many other mixed forms of interorganizational collaboration can occur during such a project. There are not only barriers between different phases of the life cycle but also within one life cycle phase. This gets especially more and more important due to the increasing trend of outsourcing in the chemical process industries (Canning, 2000).

As shown in Fig. 2, three different types of barriers between information storage and exchange can be distinguished. Following the concept and illustration of Probst in Probst, Raub, and Romhardt (2000), there are specific barriers within the analyzed design processes which can be seen in the form of gaps. Horizontal gaps indicate hierarchical barriers, vertical gaps point up the regional barriers, e.g., between departments or geographically diverted companies. The temporal aspects are shown as different layers in z-direction. During the life cycle shown in Fig. 1, brisk information exchange takes place among the enclosed information blocks

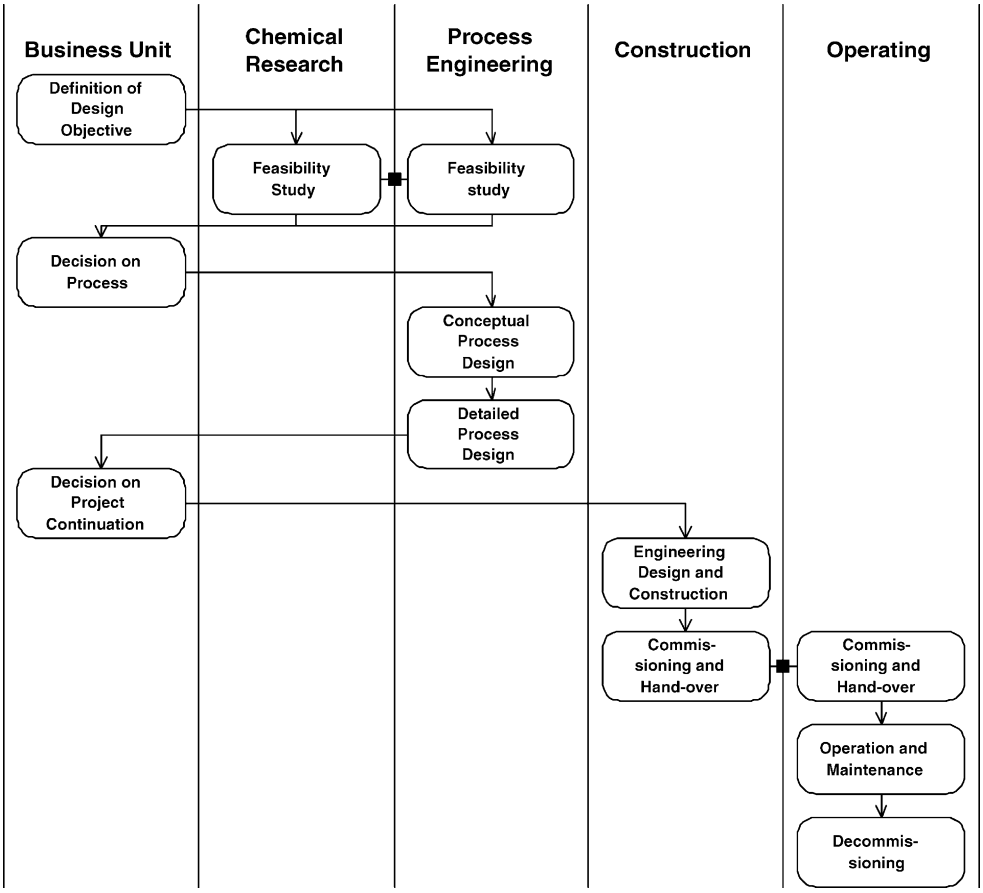

Fig. 1. Life cycle of a chemical process plant.

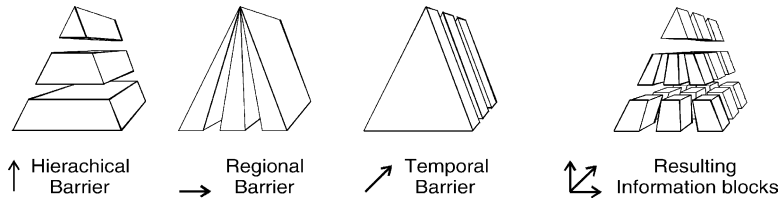

Fig. 2. Information blocks in design processes.

which result by adding the different gap-dimensions (right part of Fig. 2).

In the following, we will focus on the phases of basic engineering (conceptual process design in Fig. 1), and partially on detail design (detailed process design in Fig. 1). In these phases, concepts and basic specifications for the chemical plant are elaborated and specifications for the apparatus are defined.

In addition to historical competence segregation in different departments, the above mentioned outsourcing enforces interorganizational collaboration. Often, several partners are working concurrently on single plant-parts. In particular, the chemical engineering works on the design and dimensioning of typical chemical plant devices like reactors or separation columns. At the same time, plastics engineers are engaged in the configuration of compounding extruders. In these machines, the high-viscosity bulk polymer, which comes from the separation apparatus, is modified, e.g., with fibres and fillers or chemical additives in order to adapt its properties to complex requirements profiles for the construction of plastics parts.

By means of this organizational view on the design process, a case study for the design of a polymerization plant for polyamide 6 will be introduced in Section 2.2. This case study targets on some special aspects of the cooperation between chemical and plastics engineering. An illustration of a possible hierarchical structure in an enterprise with some information flows is shown in Fig. 3, where both the horizontal and vertical information flow between chemical and plastics engineering are displayed.

From a management point of view the administration of *many different companies* in one project is not a trivial but necessary task. Each company has its *own organizational structure* and *working procedures*. Furthermore, the issue of *privacy* is crucial. For example, the licensor of the chemical process is not interested in sharing too much information with the other parties. All these problems have to be properly addressed by a management system.

2.2. Case study polyamide 6

The case study describes the design process of a chemical plant for the production of 40 000 tons of polyamide 6 per year including the polymer compounding. It focuses on the workflow, the roles involved and the tools used together with their interactions. In Fig. 4, a simplified overview on this case study is given. Besides the performed activities and the involved roles the tools used during the design process are represented. More detailed information on the case study can be found in Eggersmann, Hackenberg, Marquardt, and Cameron (2002) and Bayer, Eggersmann, and Schneider (2002).

Two companies are involved in the considered scenario in this paper. The roles manager, reaction expert, separation expert, and laboratory expert belong to the same (chemical engineering) company. The compounding processing expert and the compounding simulation expert belong to the extruder

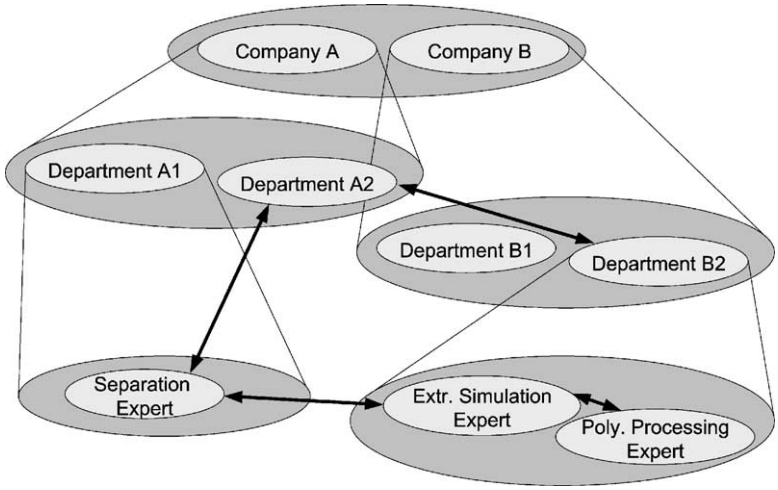

Fig. 3. Hierarchical organizational structures and information exchange.

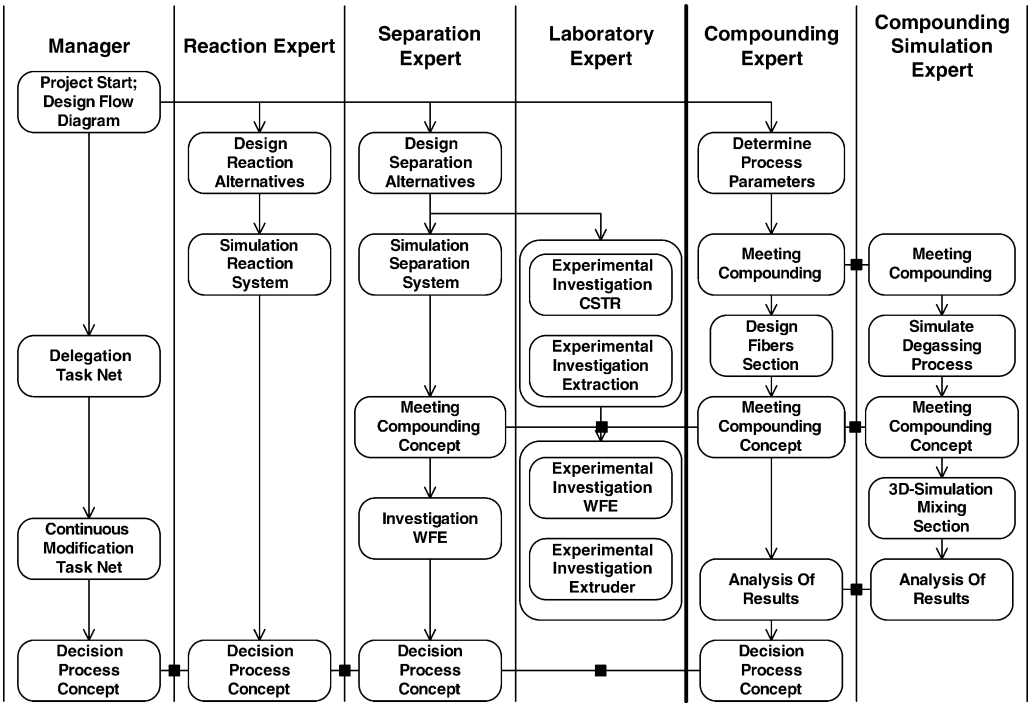

Fig. 4. Design process of a polyamid6 plant.

manufacturer. This corresponds to the classical disjunction of the two domains of chemical engineering and plastics engineering. On the chemical engineering side, the chemical process (including reaction, separation, and experiments) for the production of the raw material is designed. These activities are part of the early basic engineering. On the plastics engineering side the detailed compounding extruder configuration is designed. Typically, these activities are performed in sequential order. In our scenario they are performed simultaneously. Dependencies causing minor offset between the activities cannot be shown at this coarse-grained diagram level in Fig. 4.

The first activity in our scenario is the project start which belongs to the manager role. Here basic flow diagrams for the process are designed and human resources are assigned to corresponding roles. The manager role includes all the administration and management activities in the chemical engineering domain. The process manager defines the required process steps like reaction, separation and the compounding steps like degassing of monomers, mixing of fibers or homogenization. The reaction expert, the separation expert and the compounding expert start their activities almost concurrently. Exemplarily in Fig. 5 a detailed extruder configuration including the polymer feeding, a mixing section, a degassing section followed by the fiber adding and the degassing of air can be seen.

Further on we will concentrate on the delegation of the activities out of the chemical engineering into the plastics engineering domain. The compounding expert receives the compounding steps and information about process boundary condition such as mass flow, estimated viscosity and thermophysical polymer properties (e.g., heat capacity, thermal conductivity). Afterwards he estimates compounding specific process parameters like the machine size, the extruder's rotational speed, the mass flow in every extruder and the number of needed extruders.

Because the degassing process in the extruder can be quantified only with high experimental effort or analytically by a simulation program (Haberstroh & Schlüter, 2002), at first the degassing section is investigated by the compound-

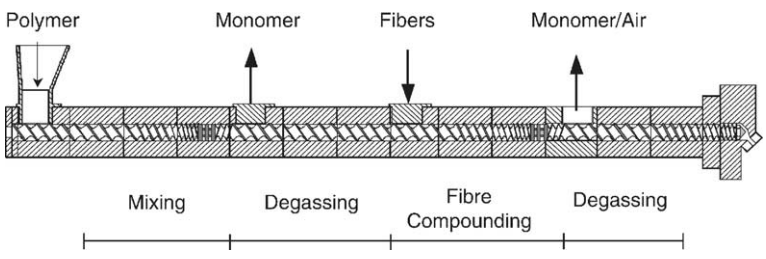

Fig. 5. Functional sections in a compounding extruder.

ing simulation expert. In a meeting, all necessary tasks are discussed and afterwards the compounding simulation expert starts a calculation to quantify the amount of degassed monomer while the compounding expert estimates the process behavior for the fibre adding section by his experience based knowledge. In the following meeting, first design results are discussed with the separation expert representing the chemical engineering company. This collaboration for the design of the separation process is necessary, because the separation of volatile components like monomers and solvents from the polymer is possible both in, e.g., a wiped film evaporator and the compounding extruders as mentioned above.

As a result of the interdisciplinary meeting, the members decide to make a detailed analysis of the homogenization processes in the mixing section by use of 3D-CFD tools (computational fluid dynamics). Afterwards the results are discussed among the plastics engineers in a second meeting to prepare a report for the chemical engineering client.

The parallel sequences of activities in chemical and plastics engineering require powerful and smart management tools which can handle the highly dynamic concurrent processes. If any of the analyzed process steps turns out to be not feasible or not economically reasonable, various activities can be affected and a large part of the complete project has to be reorganized or in the worst case cancelled.

3. A management system for interorganizational design processes

In order to support design processes in chemical engineering as described in the previous section in an adequate way, two key requirements need to be addressed. Firstly, tool support must take the *dynamics* of design processes into account. Secondly, management tools must be capable of handling *interorganizational* design processes.

The management system *AHEAD* (Jäger et al., 1999) has been designed to meet both requirements. In Nagl et al. (2003), we demonstrated how AHEAD supports dynamic design processes in chemical engineering. However, tool support was confined to the management of design processes within one organization. In the sequel, we will describe an extension to the AHEAD system which is dedicated to the management of interorganizational design processes. This extension has been elaborated in a Ph.D. thesis (Becker, Jäger, Schleicher, & Westfechtel, 2001; Jäger, 2003) reports on an early stage of this work applied in the software engineering domain.

Fig. 6 illustrates the key components of the *distributed AHEAD system*. The AHEAD system provides tools for different kinds of users. The *management environment* is used by project managers to plan and control design projects. This includes the management of products (versioned design documents), activities (design tasks), and resources (project

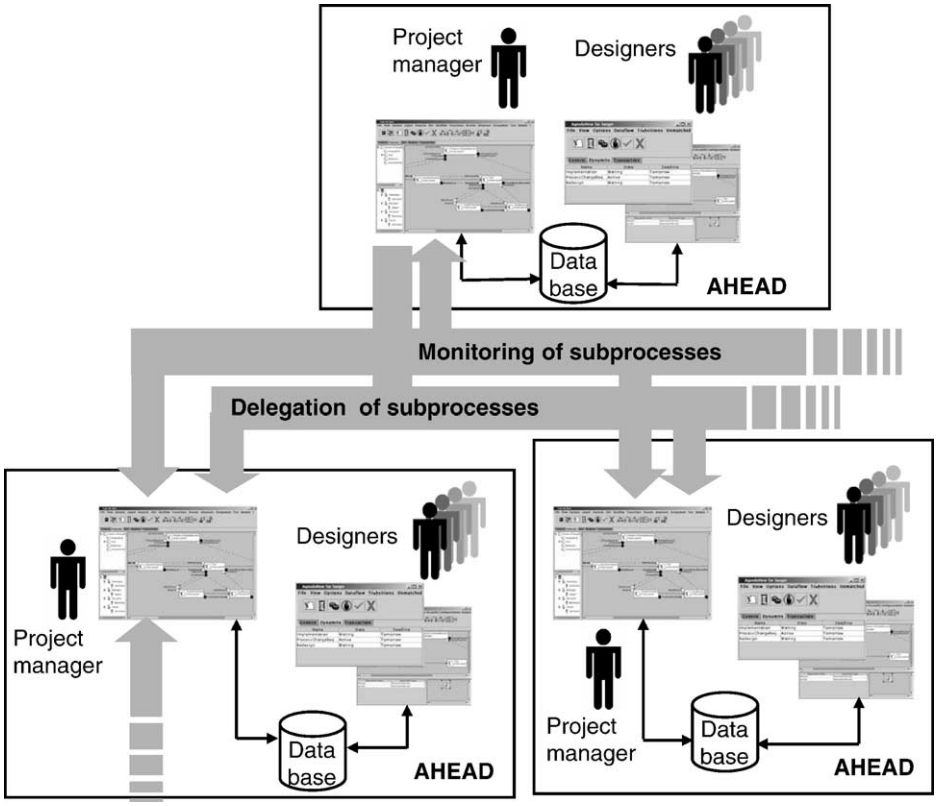

Fig. 6. Distributed AHEAD system.

team); in this paper, the main focus lies on the management of activities. The *work environment* is dedicated to designers, who are supported by *agendas* and *work contexts*. An agenda displays a set of assigned design tasks. For each task selected from the agenda, the work environment prepares a work context comprising all documents relevant for executing this task. When the designer selects a document, the work environment offers a menu of available tools which the user may activate to work on the document. Management and work environment are coupled by a shared *management database*.

The extension of AHEAD to a distributed system is illustrated by the arrows connecting different instances of the AHEAD system. AHEAD may be used to *delegate* a subprocess to a contractor. In general, a delegated subprocess consists of a connected set of subtasks; delegation is not confined to a single task. When the contractor accepts the delegation, a database is created which contains a copy of the delegated subprocess. Subsequently, execution of the subprocess is *monitored* such that the *client* may control the progress of work performed by the *contractor*.

The *delegation model* underlying the AHEAD system meets the following requirements:

Delegation of subprocesses. A delegated subprocess consists of a connected set of subtasks. This way, the client may define *milestones* for controlling the work of the contractor.

Delegation as a contract. The delegated subprocess serves as a contract between client and contractor. The client is obliged to provide the required inputs, based on which the contractor has to deliver the outputs fixed in the contract.

Autonomy of client and contractor. The autonomy of both parties is retained as far as possible; it is restricted only to the extent required by the contract.

Need-to-know principle. The parties engaged in a contract share only those data which are needed for the contract. This includes the respective subprocess as well as its context, i.e., its embedding into the overall process. Other parts of the process are hidden.

Refinement of delegated subprocesses. The contractor may refine delegated subprocesses if this is required for managing the local work assignments. Since these refinements are not part of the contract, they are not visible to the client.

Monitoring of process execution. The client is informed continuously about the state of execution of the subprocess delegated to the contractor. In this way, the client may monitor execution and control whether set deadlines are met.

Support of dynamic design processes. Support for process dynamics is extended to interorganizational design processes. In particular, contracts can be changed dynamically. However, this requires conformance to a *change protocol* because cooperation among different enterprises requires precisely defined formal rules. The change protocol ensures that the contract may be changed only when both involved parties agree.

Delegation is performed in the following steps:

(1) *Export.* The client exports the delegated subprocess into a file (an XML document). A copy of the delegated subprocess is retained in the database of the client.
(2) *Import.* The contractor imports the delegated subprocess, i.e., the file is read, and the local database is initialized with a copy of the delegated subprocess.
(3) *Run time coupling.* The AHEAD systems of client and contractor are coupled by exchanging events. Coupling is performed in both directions. This way, the client is informed about the progress achieved by the contractor. Vice versa, the contractor is informed about operations relevant for the delegation (e.g., creation of new versions of input documents).
(4) *Changing the contract.* The contract established between client and contractor may be changed according to a predefined change protocol. The change is initiated by the client, who issues a change request. In a first step, the proposed change is propagated to the contractor. In a second step, the contractor either accepts the change—which makes the changes valid—or rejects it, implying that the propagated change is undone.

Please note that steps (1)–(3) are ordered sequentially. Step (4) may be executed at any time after the run time coupling has been established.

4. Example

In this section our delegation-based approach to the management of dynamic and interorganizational design processes is demonstrated by a simplified step-by-step example session.

The scenario, drawn from the case study of Section 2.2, covers the cooperation of two companies in order to design a chemical plant for the production of polyamide 6. While the chemical engineering company is responsible for designing the chemical process (reaction, separation) for the production of the polyamide 6 material, the plastics engineering company has to design the extruder which is used for the compounding of the bulk polymer. This scenario resembles the second alternative of possible scenario settings described in Section 2. It is assumed that both companies use the AHEAD system for the management of this shared process.

4.1. Initial situation

The management environment of AHEAD (Fig. 7) is presented to the manager of the polyamide 6 design process in the chemical engineering company. The management environment supports the manager in planning, executing, analyzing and controlling design processes. In the upper region on the left-hand side a tree view representation can be seen which

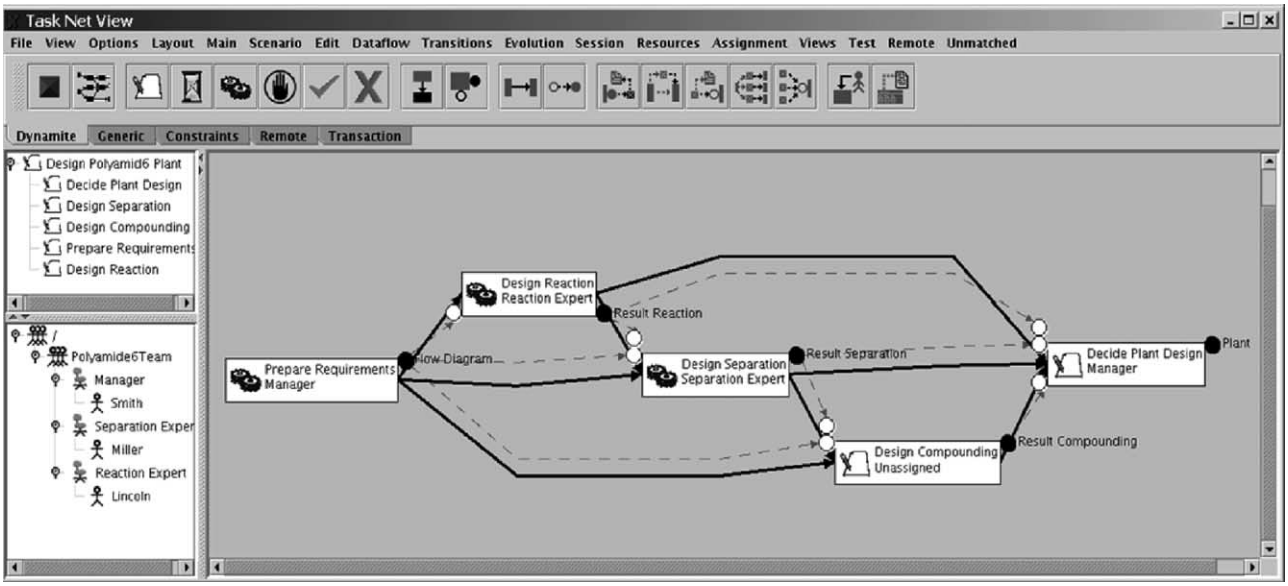

Fig. 7. Management environment of the AHEAD system.

shows the managed tasks of the overall design process. The lower region on the left-hand side displays a view onto the resources which are available for task assignments in the chemical engineering company. Three different roles, Manager, Reaction Expert and Separation Expert, together with the currently assigned human resources are listed. In the graph view on the right-hand side a task net is displayed which represents the already enacted polyamide 6 design process.

Task nets consist of tasks which are shown as rectangles and may have both inputs and outputs represented as empty and filled circles. Tasks can be linked by control flows (solid lines) defining the temporal order of the tasks. Data flows between outputs and inputs are shown as dashed lines. Tasks can be refined by another task net introducing a hierarchy relation between tasks. This hierarchy relation is not expressed directly in the representation, and the refining task net is drawn beneath its parent task instead.

To enable the execution of a task net, its standard dynamic behavior is defined by a set of rules. For example, at every time each task is in exactly one of the following execution states: In Definition, Waiting, Active, Suspended, Done and Failed. State transitions can be currently triggered either manually or by the system itself. The possible state transitions of a task are restricted by a set of formal rules defining the execution semantics of dynamic task nets. For example, a task has to be activated before any task in a refining subnet can be activated. A source of a control flow has to terminate before its target, and so on.

Design processes often cannot be planned with every detail in advance. Therefore, the corresponding task net does not remain static during execution of the design process. Rather, the task net evolves and can be edited even during its enaction. A number of restrictions apply to the modification of an already enacted task. For example, a task has to be in state In Definition or Active when its refining task net shall be modified. Likewise, adding an input or output to a task is only possible when the task is in state In Definition or Active.

In the task net of Fig. 7 the task Design Polyamide 6 Plant is decomposed on the top level into five child tasks, Prepare Requirements, Design Reaction, Design Separation, Design Compounding and Decide Plant Design. The design process has already been started and some of the tasks are in the state Active. In a first preparation phase, the requirements and specifications for the plant to be designed have been gathered and these specifications have been passed on to the three design tasks. The task Decide Plant Design will later receive and evaluate all design alternatives which have been elaborated in the design tasks. In this task one of the design alternatives for the construction of the polyamide 6 plant must finally be chosen.

In Fig. 8 the *work environment* is shown. After the login a user can see an *agenda* of all tasks which are currently assigned to his various roles. If the user selects one of the tasks from the agenda, the *work context* for this task is opened. In the upper region a graphical view shows the selected task and all tasks of the overall task net which are directly connected to it. This contextual information helps the user to understand from which tasks some input is provided and to which tasks the output is propagated. In the lower region on the left-hand side a document view displays all documents needed to execute this task. All documents received as input from previous tasks are listed here as well as the documents which are produced during execution of this task. Documents can be versioned in AHEAD. If a document is selected, its version information is displayed on the right-hand side in the lower region. If the user needs to start some tools, they are started from within this work environment.

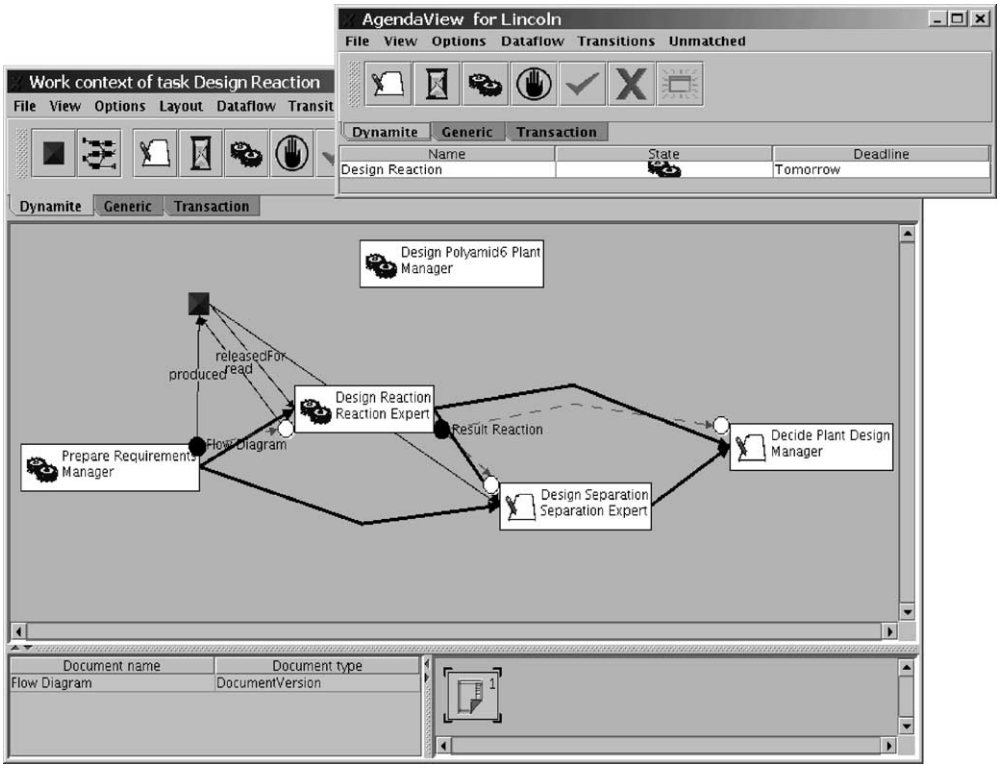

Fig. 8. Work environment of the AHEAD system.

The example session described here only deals with the part of the overall design process which is related to the design of the extruder. The chemical engineering company acts as a client and delegates the task of designing the extruder component to its contractor, the plastics engineering company.

The task net in Fig. 9 results after the manager of the chemical engineering company, acting as the client, has refined the extruder design task by a subnet (for better readability the task net is shown in a schematic representation instead of a screen snapshot of the management environment). The task Design Compounding and all subtasks will be delegated to the contractor. Their execution state In Definition denotes that the tasks are still under planning and cannot be activated at the moment. Furthermore these tasks are currently not assigned to a responsible role. Design Compounding is specified with requirement documents as input and an extruder design alternative as output. This task definition can be seen as a *contract* between both companies, where the contractor has to produce a certain output based on the inputs which are provided by the client.

As stated before, in this subprocess an extruder is developed according to a set of desired product properties. The subtask Determine Process Parameters receives a product quality specification and the extruder's properties as input and produces rough estimates for the extruder's parameters, e.g., operating temperature and capacity of the extruder. The content of fibres as well as the degassing of volatile components of the plastics are investigated in separate tasks. The subsequent investigation of the extruder's functional sections in task Investigate Extruder is based on the output of the previous tasks. The results are evaluated and if the desired properties are met, the extruder design is propagated as a preliminary result to the parent task Design Compounding.

4.2. Establishing the delegation

The delegated subprocess Design Compounding and its refining task net is exported to a file. Fig. 10 shows the view on the overall design process as it is presented to the manager on the client side after the export step. All delegated tasks are executed in a remote management system of a contractor, so these tasks are assigned to a newly created resource Remote: Extruder Company. They can be monitored from the client side because a copy of the subprocess is retained in the local database. The export file also contains contextual information about the delegated process, namely the part of the overall task net in which the delegated process is embedded. The context information is not part of the contract between client and contractor. For example, the tasks Prepare Requirements, Design Separation and Decide Plant Design have a direct control-flow to one of the delegated tasks and thus belong to the context. Of course, the context tasks are still executed on the client side, but they are monitored in a remote management system. In contrast to these monitored tasks, the task Design Reaction does not belong to the context and this private task cannot be seen elsewhere. Private tasks, monitored tasks, and re-

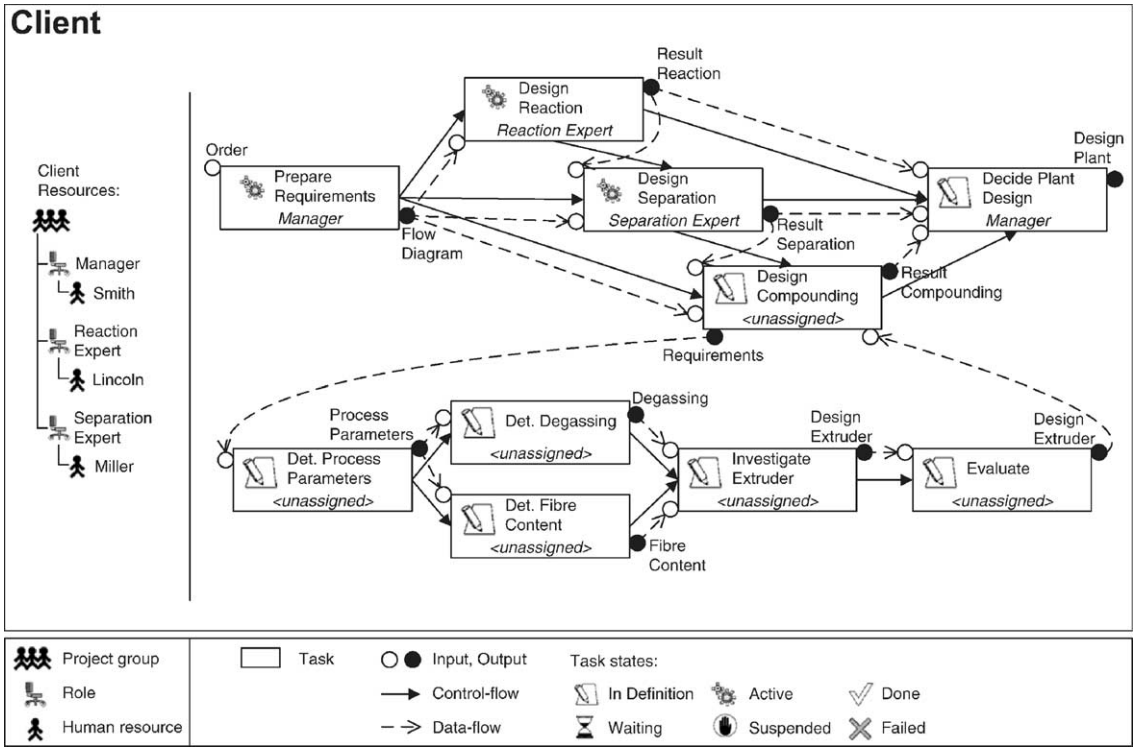

Fig. 9. Task net after refinement of task Design Compounding.

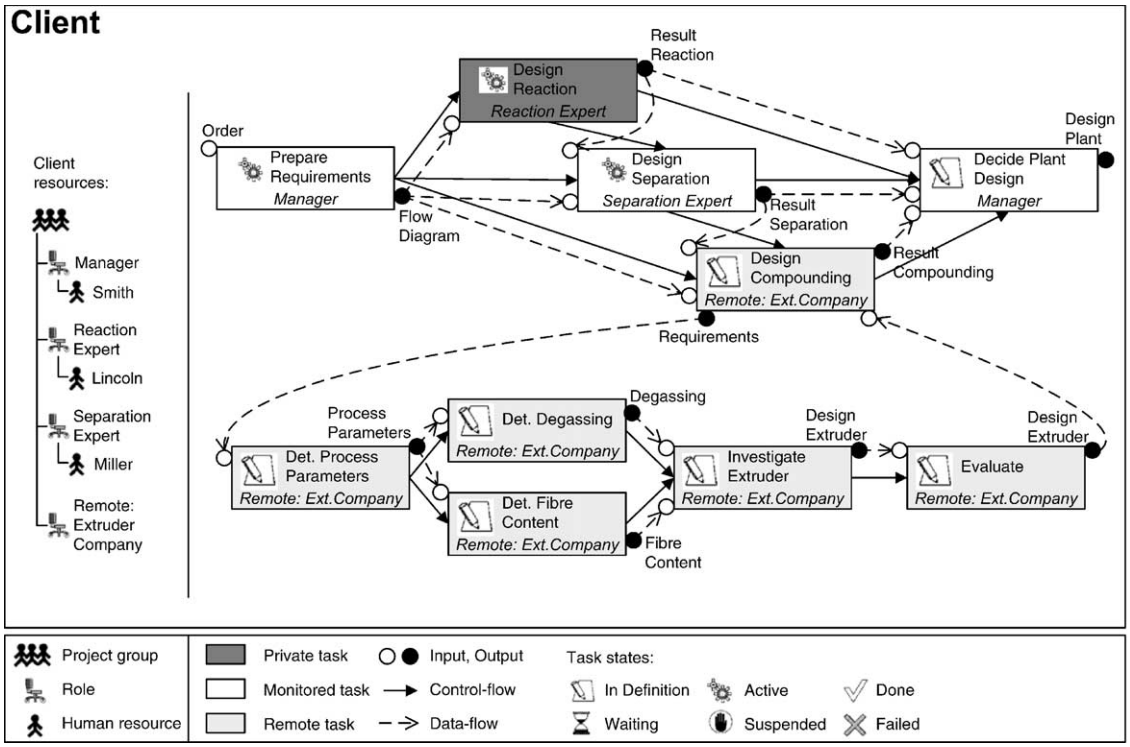

Fig. 10. Task net on client side after delegation.

motely executed tasks are filled with different patterns in the figure.

The process description file is transferred to the contractor, the extruder company, and there it is imported into the AHEAD system. Fig. 11 shows the corresponding task net and its context which are instantiated in the local database on the contractor side. All tasks are in state In Definition, so that they can be edited and executed in the management environment by the manager on the contractor side. The contract consists of the delegated tasks, their parameters, control-flows and data-flows. It defines exactly how the delegated process must be executed. Therefore, the contractor has to check carefully if he can accept the order to execute the transferred task net for the client. If the contractor wants to change some details, he has to contact the client and negotiate these changes. A new version of the export file including the changes is generated and transferred to the contractor. When the contractor agrees to execute the transferred process, he may begin with the execution of the corresponding task net in his management environment. A first step for the manager on the contractor side is to assign all delegated tasks to either the role Compounding Expert or Compounding Simulation Expert as shown in Fig. 11.

The management systems of client and contractor are loosely coupled together by exchanging events to synchronize their local copies of the shared task net. The client is informed about changes of the delegated tasks. The execution state of these *milestone* activities keeps the client informed about the progress of the delegated process fragment. Vice versa, the contractor is informed about changes of the context tasks which are executed on the client side. Of course, no information is exchanged between both systems about elements of the task net which are private either on the client side or on the contractor side. To illustrate this concept, it is assumed that the state of the task Prepare Requirements, which is executed on the client side and monitored on the contractor side, is changed from Active to Done. A corresponding change event is sent from the client to the contractor and on the contractor's side the execution state of the copy of the task Prepare Requirements is updated (see Fig. 11).

The delegated task Design Compounding is activated by the human resource playing the role Compounding Expert role on the contractor side. Subsequently, the tasks Determine Process Parameters, Determine Fibre Content, Determine Degassing and Investigate Extruder are executed by the assigned resources. According to the predefined data-flows the produced results are passed between these tasks. After the results of the investigation have been evaluated in task Evaluate, the extruder design is sent as a preliminary result to the parent task Design Compounding. All these updates of the delegated tasks by the contractor can be monitored by the client as well as the produced result, the first version of the extruder design.

4.3. Changing the delegated task net dynamically

In our example, the client and the contractor agree that the preliminary design for the extruder could be optimized if the

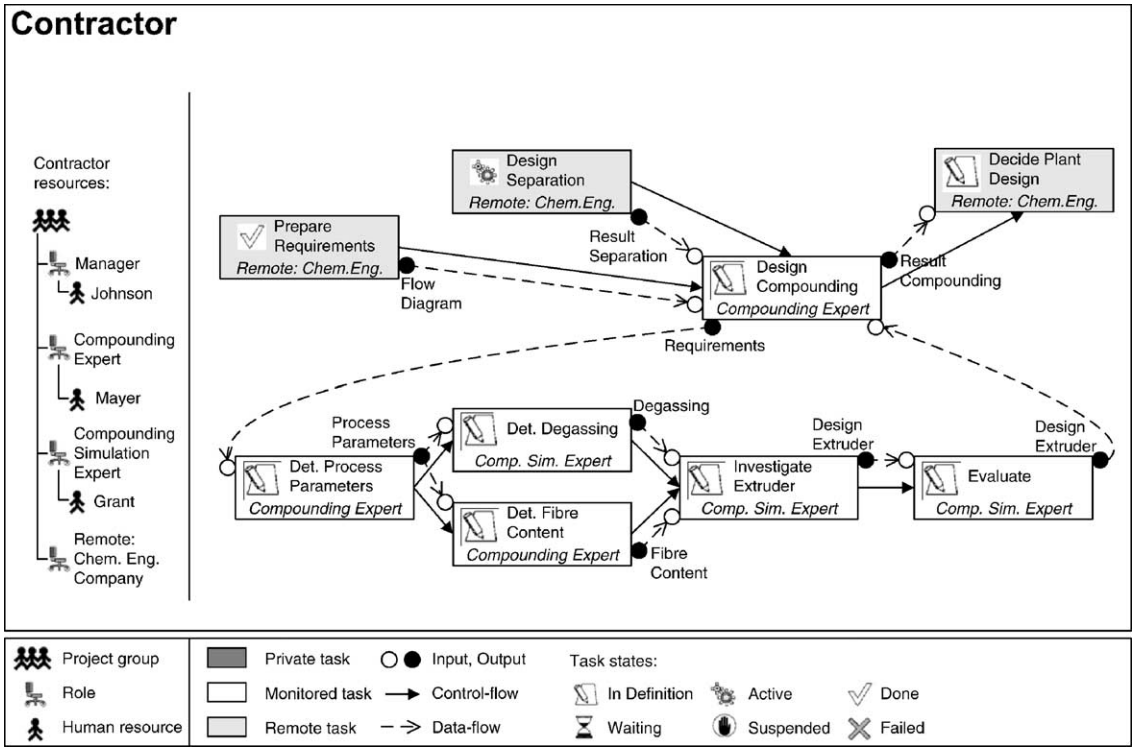

Fig. 11. Task net on contractor side after delegation and resource assignments.

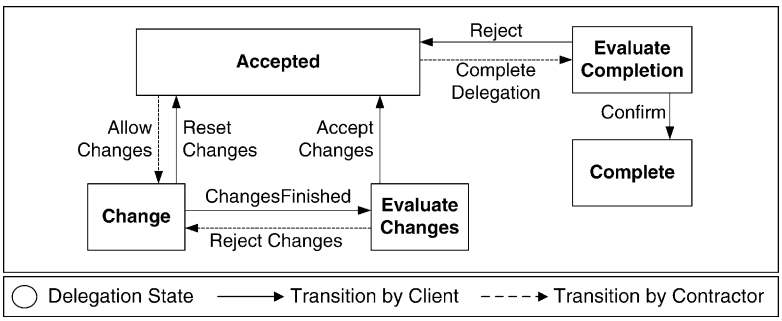

Fig. 12. Change protocol for changes of the contract (after Jäger, 2003).

mixing quality of the materials in the extruder is investigated further. This is done by performing a three-dimensional simulation of the polymer flow in the extruder. The contractor agrees to carry out the additional simulation, and the contract between client and contractor can be extended. Changing a delegated process after its enaction is not unusual in the design process of a chemical plant.

AHEAD supports dynamic changes of the contract between client and contractor. Changes are allowed only when both parties agree on them, because the changes may have substantial consequences for either of them. The delegated task net is changed according to a formal *change protocol* which is shown by means of a state transition diagram in Fig. 12. The delegated task net is at every time in exactly one of the three *delegation states*, Accepted, Change and EvaluateChanges. As described below, the transitions between these states define the commands which can be executed either on client and contractor side during the change process.

Initially, the contractor has issued the command Allow changes to signal that he agrees to the change proposal of the client. The delegation state of the task net is changed from Accepted to Change. After that, the client is able to modify the delegated process. As shown in Fig. 13, the client adds a new task Determine Mixing Quality in the subnet of the Design Compounding and adds the appropriate control- and data-flow relationships from Determine Process Parameters and Investigate Extruder. While the client changes the task net, all changes to the task net are propagated to the contractor. Eventually, the client may either discard his changes by using the command Reset Changes or he may signal that the structural changes are finished by using the command Changes Finished. In this case, the task net changes its delegation state to EvaluateChanges.

In our example, the contractor evaluates and accepts the proposed changes of the delegated process as valid. Triggering the command Accept Changes changes the delegation state of the task net to Accepted and yields an update of both process views on the client side and the contractor side according to these changes. As an alternative, the contractor may reject the change of the contract by use of the command Reject Changes. In this case, the delegated process changes its delegation state back to Changes, the changes are discarded and the client would be informed about the rejection. Both partners then would have to talk about the problem again before eventually the contractor would accept a proposition made by the client (or the client invokes the command ResetChanges).

The complex new task Determine Mixing Quality in the delegated process is refined by the manager on the contractor side by a private subnet to break it down into smaller working units and assign separate resources to each of the tasks. This refining task net comprises the tasks Prepare Simulation, Generate Mesh, Perform 3D-Simulation and Evaluate as shown in Fig. 14. The subnet is not part of the contract between client and contractor and therefore invisible to the client. Of course, client and contractor can decide on their own to make some private task net elements visible to the other party at every time and can turn these elements into private ones again.

4.4. Finishing the delegation

After the process has been resumed, on the contractor side a second version of the extruder design has been finally produced and released to the task Design Compounding. This result should be taken as the final outcome of the delegated task. The contractor can signal this to the client with the command Complete Delegation stating that he wishes to complete the contracted delegation relationship. The client can confirm this with the command Confirm or reject it with the command Reject. If the result is accepted, the coupling of the two AHEAD systems of client and contractor is finished. In the other case, the rejection is signaled to the contractor and the coupling is maintained.

5. Realization

In the sequel, we describe first how the locally operating AHEAD system is realized. Next, we turn to the realization of the distribution mechanisms (delegation and monitoring of subprocesses, which were illustrated at a conceptual level in Fig. 6).

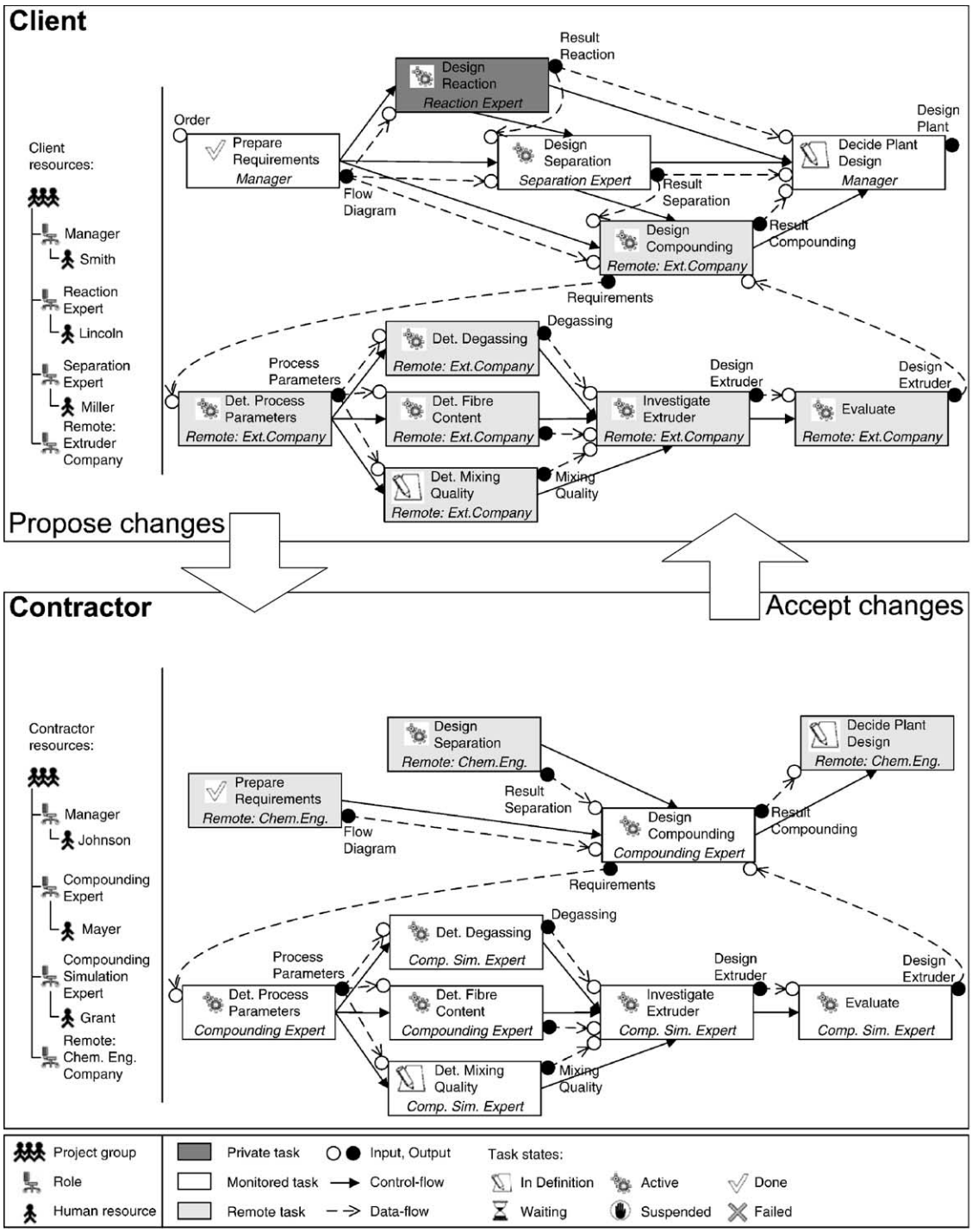

Fig. 13. Delegated process after addition of a new task Determine Mixing Quality.

5.1. Local AHEAD system

The architecture of a locally operating AHEAD system is illustrated in Fig. 15. The AHEAD system provides *environments* (collection of tools) for different kinds of users. Please note that the figure refers to roles rather than persons. In particular, a single person may acquire multiple roles. For example, it is quite common that the same person acts both as manager and as chief designer.

In this paper, we have focused mainly on the *management environment*, which offers graphical tools for managing products, activities, and resources. In particular, the manager may build up a task net for planning the overall design process, monitoring process execution, and re-planning the task net whenever considered appropriate or necessary.

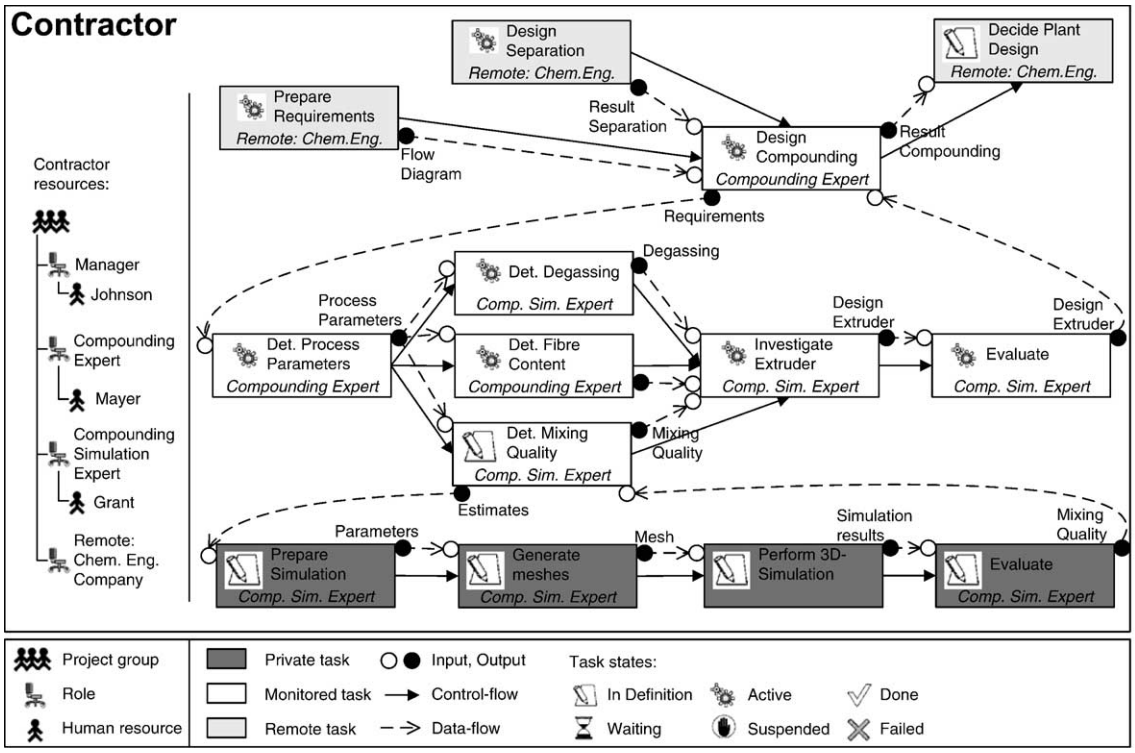

Fig. 14. Private refinements of the delegated process on the contractor side.

The *work environment* provides designers with an agenda tool, which shows all tasks assigned to a specific designer. The designer may pick up a task from the agenda, start its execution, and open a work context displaying all documents required for this task. From the work context, the designer may launch external tools such as flowsheet tools, simulation tools, etc.

The implementation of the AHEAD system is based on *graph technology* and makes heavy use of tools for rapid prototyping as described below. These tools have been developed

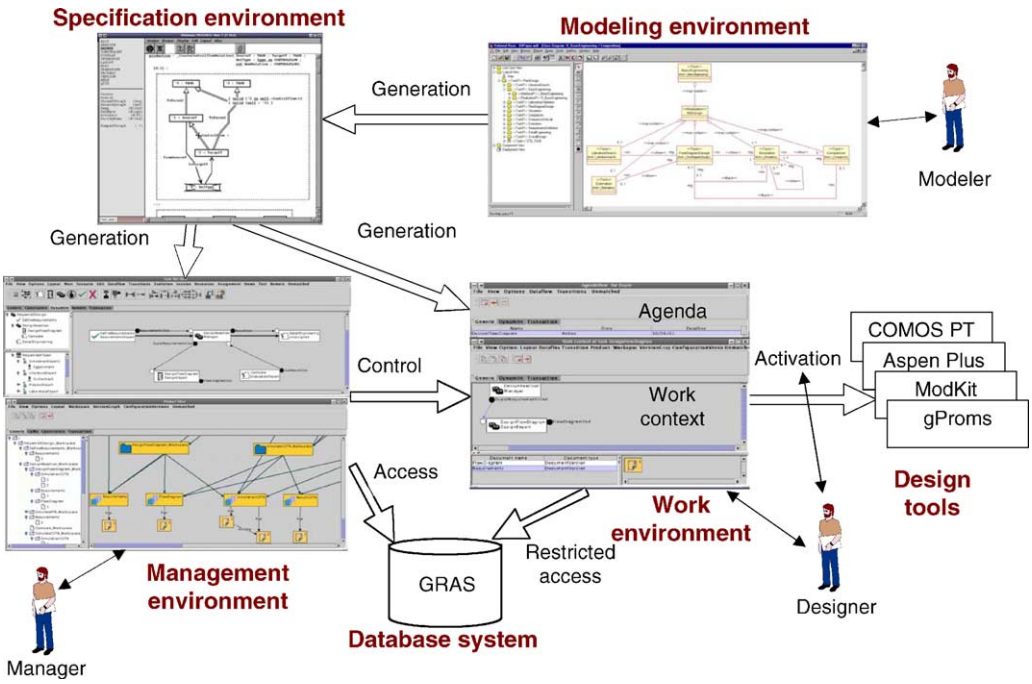

Fig. 15. Architecture of a local AHEAD system.

at our department in various research projects. By using these tools a demonstrator was implemented with comprehensive functionality, which, however, cannot be immediately employed in industry.

All management data (task nets, resources, and products) are represented as graphs. The data are stored in the graph-based database GRAS (Kiesel, Schürr, & Westfechtel, 1995). Management environment and work environment have access to these data. The internal graph structures and all commands operating on the graph structures are formally specified as a graph rewriting system using the language PROGRES (Schürr, Winter, & Zündorf, 1999). The specification of the AHEAD system comprises about 200 pages of PROGRES code. From this specification the application logic of AHEAD is generated as C-Code which operates on the GRAS database.

The application logic is integrated with a graphical user interface which is based on the UPGRADE framework (Böhlen, Jäger, Schleicher, & Westfechtel, 2002). With UPGRADE the internally used management graphs can be presented to the user by graphical representations as well as tree or table representations. UPGRADE is fully implemented in JAVA, based on public-domain and commercial components (ILOG JViews), and comprises about 70 packages, 250 classes, and 70 000 lines of code.

Finally, the *modeling environment* is used to define domain-specific design processes on the type level using the unified modeling language (UML). In this way, modelers, i.e., domain experts such as chemical engineers are shielded from the underlying graph technology. In particular, we employ class diagrams to define classes of tasks and control flow and data flow associations between them. The UML models are automatically transformed into PROGRES specifications.

The architecture of the AHEAD system differs considerably from the architecture of traditional workflow management systems (Lawrence, 1997). The core component of a workflow management system is a workflow engine which executes a workflow instance, which is instantiated from a workflow definition. To a limited extent, the workflow instance may be changed during execution; the mechanisms provided to this end vary from system to system. In AHEAD, the instance-level task net stored in the graph-based database system GRAS roughly corresponds to the workflow instance. However, AHEAD provides full-fledged support for dynamic changes of the instance-level task net. The workflow engine is realized by the C code generated from the PROGRES specification. In the case of AHEAD, the workflow engine offers not only execution support (starting of tasks, reading of inputs, etc.), but also fully integrated planning and re-planning support (i.e., the task net may be changed on the fly, taking the current state of execution into account). All operations on the task net are constrained by process model definitions represented in the Unified Modeling Language. But again, a process model definition in AHEAD is much more general and flexible than a workflow definition. In particular, we follow an object-oriented approach by dynamically instantiating tasks and relationships from corresponding classes and associations. In contrast, a workflow definition represents a (more or less) static program to be executed at runtime.

5.2. Distributed AHEAD system

The realization of the distribution mechanisms (delegation and monitoring of subprocesses) is shown in Fig. 16. Two instances of AHEAD are coupled together and exchange events to keep each other informed about changes of the delegated process. After client and contractor have exchanged the pro-

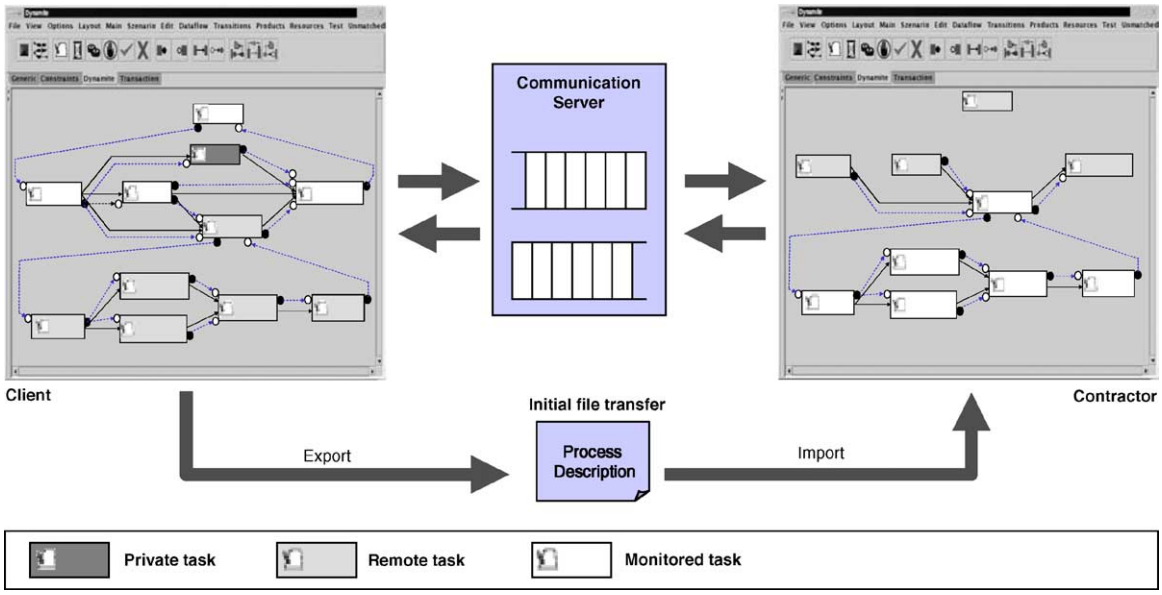

Fig. 16. Concept for the coupling of two AHEAD systems.

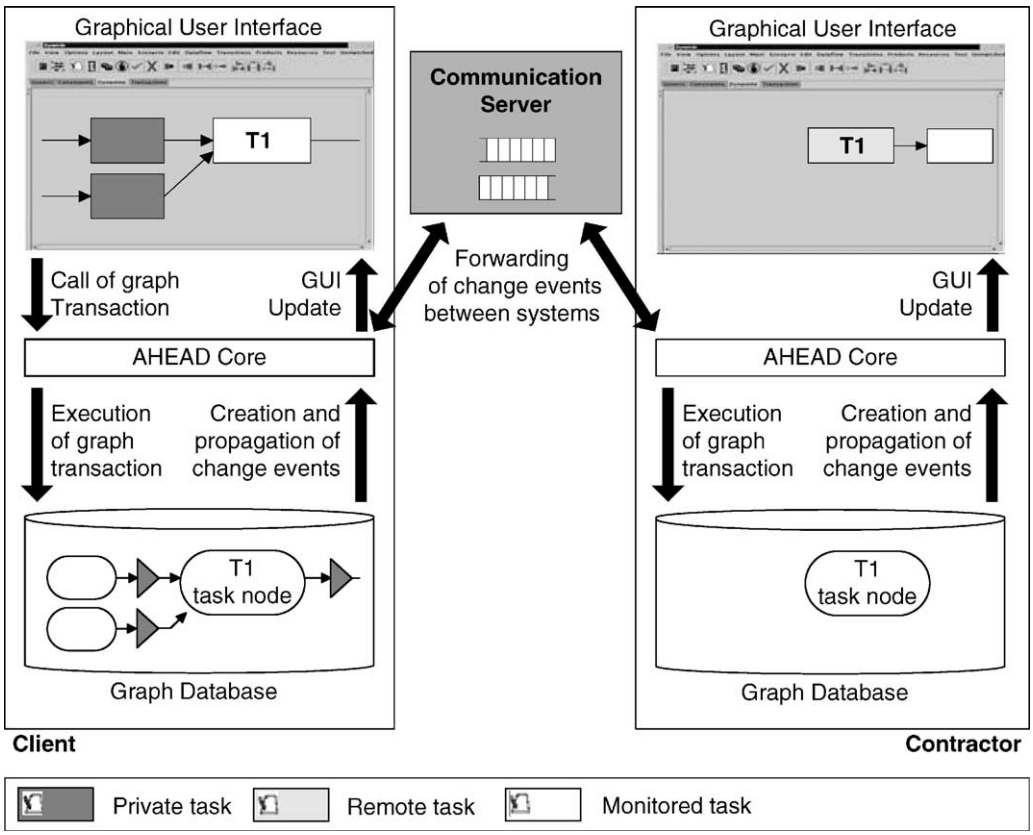

Fig. 17. Realization of the coupling of two AHEAD systems.

cess description file and set up their copies of the delegated process in their local graph databases initially, both management systems connect to a *communication server*. Now the delegation relationship is established and maintained until the delegation has been completed. The communication server between both systems is used to store the exchanged change events if one of the systems is temporarily disconnected. All queued messages are sent to the management system once it reconnects to the communication server. Via the communication server both systems are loosely coupled at run-time, so they do not need to run at the same time in both companies. But if at least one system is running, the communication server has to be available to receive, store and forward change events.

A more detailed view on the graph-based realization of the delegation concept is given in Fig. 17. Two AHEAD systems are coupled together using a communication server. Let us first concentrate on the AHEAD system on the left hand side. Each AHEAD system consists of a graphical user interface, the AHEAD core and the underlying graph database. The task net shown in the graphical user interface is created step by step by invoking special user interface commands, for example, to insert a new task or a new control-flow relationship between two tasks. Each user interface command calls a graph transaction of the application logic in the AHEAD core. The execution of a graph transaction leads to the manipulation of the graph data stored in the graph database. In the example, at the graphical user interface a task T1 is displayed. This task is represented as a special (task) node in the graph database. Invoking a user interface command to activate task T1 leads to a change of one of the attributes of the corresponding graph node in the database. The database propagates all changes on the graph data back to the AHEAD core. According to these change events the current state of the graphical user interface is updated.

If some task net elements are also monitored within another coupled AHEAD system, all change events regarding to these elements are forwarded (via the communication server) to the coupled system, too. There, corresponding graph transactions in the AHEAD core are called for each of these change events. Accordingly, the graph data stored in the graph database is manipulated and the graphical user interface is updated. In the figure, the task T1 is also monitored within the AHEAD system on the right hand side. There, a copy of T1 is displayed in the graphical user interface (marked as *remote*) and a corresponding graph node is stored in the graph database. By the exchange of change events, all changes of T1 in the client system are also performed on the copy of T1 in the contractor system. Every AHEAD system can at the same time act as a producer of change events regarding all elements which are monitored in coupled systems and as a consumer of change events regarding all elements which are executed elsewhere and only monitored locally.

Dynamic changes of the structure of the delegated task net (for example, inserting a new control-flow relationship between two tasks) are possible according to a formal change protocol (as presented in Section 4.3). This change protocol is fully implemented within the application logic in the AHEAD core. Therefore, all graph transactions of the application logic which are responsible for performing structural changes are equipped with an *analysis mechanism* to check if only a locally executed part of the task net is involved or not. If the structural change also affects task net elements which are executed in another coupled system, additional checks implementing the formal change protocol are performed in the local AHEAD system to ensure that client system and contractor system both are in appropriate delegation states. If these tests are not passed successfully, the further execution of the graph transaction is aborted. Otherwise, the graph transaction is executed and the local graph database is manipulated. This change may lead to the execution of a graph transaction in a coupled system (as described before).

The communication server and the required coupling infrastructure in the AHEAD system are implemented in JAVA and integrated with the UPGRADE framework (see above).

6. Related work

6.1. Related work in chemical engineering

n-dim (Subrahmanian et al., 1997) is a design process support system which is based on an object-oriented data model. The model is defined such that data may be organized along multiple dimensions. Objects are instantiated by cloning prototypical objects. An n-dim database may contain references to external data of any kind, e.g., web pages, files, tuples in databases, etc. n-dim has been applied in different contexts to model versioned design data as well as the design process.

KBDS (Bañares-Alcántara, 1995; Bañares-Alcántara & Lababidi, 1995) supports the design process by recording design alternatives which are related to design objectives. Process support is integrated into a flowsheet tool which represents design alternatives and their rationales. This way, the designer may document and evaluate alternative design decisions. This approach has been developed further in *PRIME* (Pohl et al., 1999) and *MODKIT* (Bogusch, Lohmann, & Marquardt 2001), each of which relies on a process engine for providing both process guidance and process automation.

All design process support systems mentioned so far address design processes at a rather fine-grained level. Their main purpose is to record the design process history and to support the designer with process fragments for frequently recurring sequences of process steps. These systems do not address the management of interorganizational design processes as provided by the AHEAD system.

Document management systems such as, e.g., *Documentum* or the *Intergraph Engineering Framework* (Jenkins, 2002) are becoming more and more popular in chemical engineering. Document management systems provide distributed access to a wide range of documents in a local area network (LAN) or even in a wide area network (WAN). However, they focus on the management of documents rather than on the management of activities.

6.2. Related work in other domains

Management of *distributed processes* is addressed by a number of workflow management systems. However, a distributed process need not be interorganizational as addressed in this paper. In our terminology, "interorganizational" refers to the cooperation across multiple enterprises. Many workflow management systems, however, address only the distribution of tasks and data within one enterprise. For example, *Mentor* (Wodtke, Weissenfels, Weikum, Kotz-Dittrich, & Muth, 1997) is a workflow management system which provides multiple workflow servers. Work is distributed among the workflow servers according to a sophisticated load balancing algorithm.

van der Aalst (1999) provides an overview of paradigms for *interorganizational processes*. Among others, the following paradigms are identified:

Process chaining. From some process p, a process q is launched to continue the overall process. The only interaction between p and q occurs when q is started. Subsequently, p and q perform independently of each other.

Subcontracting. A task t of the overall workflow is passed to a contractor, which executes t and passes the results back to the client. From the perspective of the client, t appears to be atomic. The client and the contractor interact both at the start and at the end of the execution of the subcontracted process.

Loosely coupled processes. Processes are executed in parallel in different organizations. Occasionally, they interact at pre-defined communication and synchronization points.

Case transfer. The workflow is seen as a case which has to be transferred among different organizations. Transferring the case includes transfer of documents and transfer of the current state of execution. Only one organization at a time may execute the case.

van der Aalst (1999) primarily focuses on case transfer and an extended variant thereof. In van der Aalst (2000), the same author discusses loosely coupled processes. The interaction paradigms process chaining and subcontracting are supported by the standards defined by the workflow management coalition (WfMC: Lawrence, 1997). In addition, subcontracting was introduced as early as 1987 by the Istar system (Dowson, 1987) in the software engineering domain.

The *delegation model* of the AHEAD system adds a new paradigm to the classification scheme presented above. It differs from process chaining inasmuch as client and contractor do interact while the delegated subprocess is being

executed. The delegation-based approach also differs from the case transfer model because both parties perform their parts of the overall process in parallel: The client is not suspended when a subprocess is delegated to a contractor. Delegation constitutes a significant extension of subcontracting because subprocesses rather than single tasks may be delegated in general. Like loosely coupled processes, client and contractor may interact during the execution of the delegated subprocess rather than merely at the start and the end, respectively. Delegation differs from loosely coupled processes because there is a hierarchical relationship between client and contractor (while loosely coupled processes are peer to peer in general). Finally, the delegation-based approach supports dynamic changes, while loosely coupled processes have been introduced for statically defined workflows.

7. Conclusion

We have presented the AHEAD system, which supports the management of dynamic and interorganizational design processes. AHEAD provides a delegation-based approach for coordinating distributed design processes in which multiple enterprises are involved. A subprocess may be delegated to a contractor. Delegation is performed in three steps: export of the subprocess from the client's system, import into the contractor's system, and subsequent run time coupling. Delegation conforms to the need-to-know principle; only those data required for the delegation are shared by both parties. The autonomy of client and contractor is retained as far as possible. Client and contractor are coupled by exchanging events such that both parties are mutually informed about relevant state changes. This way, the client may monitor process execution by the contractor (e.g., completion of milestones), and the contractor is made aware of relevant changes in the context of the delegated subprocess. Once the run time coupling has been established, the contract may be changed at any time according to a pre-defined change protocol allowing for change proposals initiated by the client which are either accepted or declined by the contractor.

We have applied the AHEAD system to an industrially relevant case study, namely the basic engineering and conceptual design of a chemical plant for producing polyamide 6. The experiences gained from this case study are promising. Design of a plant for polyamide 6 requires the cooperation of multiple enterprises. It is crucial that cooperation is based on clearly defined contracts and sensitive information is kept private. The latter is achieved by adhering to the need-to-know principle. Delegation of subprocesses rather than single tasks makes it possible to monitor the achievement of milestones. Finally, the required flexibility is provided by enabling changes to established contracts as well as by refining delegated subprocesses at the contractor's site.

The experiences gained from this application have shown some potential areas of future work:

Integration of heterogeneous management systems. In this paper, we have assumed *homogeneous management systems*, i.e., all involved parties need to use the AHEAD system. However, it is likely that different enterprises employ different management systems. So far, we have implemented a prototypical coupling with a commercial workflow management system (COSA) (Jäger, 2003), but this work still needs to be generalized.

Improved flexibility. In the delegation-based approach developed so far, the contractor has to execute the delegated subprocess as it stands. If the steps are too coarse-grained, the contractor may refine delegated subtasks. Other changes are not allowed, e.g., the contractor may not add internal milestones in between the milestones monitored by the client. A more flexible approach would involve the separation of an *interface process*, which defines the contract with the client, and a *realization process*, which is defined by the contractor as desired (and which is mapped onto the interface process); see van der Aalst and Weske (2001).

Data security. While the delegation-based approach obeys the need-to-know-principle, the current implementation does not check whether documents involved in the delegation contain sensitive data. To this end, a classification scheme has to be developed such that documents may be assigned to classes of confidentiality. Based on such a classification, the AHEAD system would prevent the transfer of documents which are classified as confidential.

Still, there is a clear need for further evaluation in an industrial context. Not only does this apply to the delegation-based approach presented in this paper; rather, this need extends to the AHEAD system as a whole. After an initial ergonomic evaluation (Foltz, Westfechtel, Schmidt, & Luczak, 2003), we are currently redesigning the user interface of the AHEAD system to make it more amenable to industrial use. More generally, we are addressing *technology transfer* along the lines described in Nagl et al. (2003).

Acknowledgments

This work has been carried out in the IMPROVE project, which is funded by the Deutsche Forschungsgemeinschaft (DFG). Funding support is gratefully acknowledged.

References

Bañares-Alcántara, R. (1995). Design support systems for process engineering. I. Requirements and proposed solutions for a design process representation. *Computers & Chemical Engineering, 19* (3), 267–277.

Bañares-Alcántara, R., & Lababidi, H. (1995). Design support systems for process engineering. II. KBDS: An experimental prototype. *Computers & Chemical Engineering, 19* (3), 279–301.

Bayer, B., Eggersmann, M., Gani, R., & Schneider, R. (2002). Case studies in design and analysis. In B. Braunschweig, & R. Gani (Eds.), *Software architecture and tools for computer aided process engineering* (pp. 591–634). Amsterdam: Elsevier.

Becker, S., Jäger, D., Schleicher, A., & Westfechtel, B. (2001). A delegation-based model for distributed software process management. In V. Ambriola (Ed.), *Proceedings of the 8th European Workshop on Software Process Technology*. (EWSPT 2001) LNCS 2077 (pp. 130–144). Witten, Germany: Springer-Verlag.

Bogusch, R., Lohmann, B., & Marquardt, W. (2001). Computer-aided process modeling with ModKit. *Computers & Chemical Engineering*, *25* (7–8), 963–995.

Böhlen, B., Jäger, D., Schleicher, A., & Westfechtel, B. (2002). UPGRADE: Building interactive tools for visual languages. In N. Callaos, L. Hernandez-Encinas, & F. Yetim (Eds.), *Proceedings of the 6th World Multiconference on Systemics, Cybernetics, and Informatics*. (SCI 2002) (Information Systems Development I, Vol. I, pp. 17–22). Orlando, FL.

Braunschweig, B., & Gani, R. (Eds.). (2002). *Software architecture and tools for computer aided process engineering*. Amsterdam: Elsevier.

Canning, K. (2000). Fostering agility through outsourcing. *Chemical Processing*, *63* (10), 28–31.

Dowson, M. (1987). Integrated project support with Istar. *IEEE Software*, *4*, 6–15.

Eggersmann, M., Hackenberg, J., Marquardt, W., & Cameron, I. (2002). Applications of modelling: A case study from process design. In B. Braunschweig, & R. Gani (Eds.), *Software architecture and tools for computer aided process engineering* (pp. 335–372). Amsterdam: Elsevier.

Foltz, C., Westfechtel, B., Schmidt, L., & Luczak, H. (2003). Use-centered interface design for an adaptable administration system for chemical process design. In C. Stephanidis, J. Jacko (Eds.), *Proceedings of the International Conference on Human–Computer Interaction* (Vol. 2, pp. 365–369). Crete, Greece: HCI International.

Georgakopoulos, D., Prinz, W., & Wolf, A. L. (Eds.). (1999). *Proceedings of the International Joint Conference on Work Activities Coordination and Colloration (WACC-99). Vol. 24-2 of ACM SIGSOFT Software Engineering Notes*. San Francisco, CA: ACM Press.

Haberstroh, E., & Schlüter, M. (2002). Design of twin screw extruders with the MOREX simulation software. Polymer Processing Society (PPS) 18.

Heller, M., & Westfechtel, B. (2004). Dynamic project and workflow management for design processes in chemical engineering. In *Proceedings of the 8th International Symposium on Process Systems Engineering*. (PSE 2003) (pp. 208–213). Kunming, China.

Jäger, D. (2003). Unterstützung übergreifender Kooperation in komplexen Entwicklungsprozessen. Ph.D. thesis, RWTH Aachen, Aachen, Germany, Vol. 34 of Aachener Beiträge zur Informatik.

Jäger, D., Schleicher, A., & Westfechtel, B. (1999). AHEAD: A graph-based system for modeling and managing development processes. In M. Nagl, A. Schürr, & M. Münch (Eds.), *AGTIVE—Applications of graph transformations with industrial relevance. LNCS 1779* (pp. 325–339). Castle Rolduc, The Netherlands: Springer-Verlag.

Jenkins, A. (2002). The engineering framework. In *Engineering in der Prozessindustrie—Anforderungen und Lösungen. VDI-Berichte 1684* (pp. 17–23). Frankfurt, Germany: VDI-Verlag.

Kerzner, H. (1998). *Project management: A systems approach to planning, scheduling and controlling*. New York: Wiley.

Kiesel, N., Schürr, A., & Westfechtel, B. (1995). GRAS, a graph-oriented software engineering database system. *Information Systems*, *20* (1), 21–51.

Lawrence, P. (Ed.). (1997). *Workflow handbook.* Chichester, UK: Wiley.

Nagl, M., & Marquardt, W. (1997). SFB-476 IMPROVE: Informatische Unterstützung übergreifender Entwicklungsprozesse in der Verfahrenstechnik. In M. Jarke, K. Pasedach, & K. Pohl (Eds.), *Informatik '97: Informatik als Innovationsmotor. Informatik aktuell* (pp. 143–154). Aachen, Germany: Springer-Verlag.

Nagl, M., Westfechtel, B., & Schneider, R. (2003). Tool support for the management of design processes in chemical engineering. Computers & Chemical Engineering, *27* (2), 175–197.

Pohl, K., Klamma, R., Weidenhaupt, K., Dömges, R., Haumer, P., & Jarke, M. (1999). Process-integrated (modelling) environments (PRIME): Foundations and implementation framework. *ACM Transactions on Software Engineering and Methodology*, *8* (4), 343–410.

Probst, G., Raub, S., & Romhardt, K. (2000). *Managing knowledge*. New York: Wiley.

Schürr, A., Winter, A., & Zündorf, A. (1999). The PROGRES approach: Language and environment. In H. Ehrig, G. Engels, H. -J. Kreowski, & G. Rozenberg (Eds.), *Handbook on graph grammars and computing by graph transformation: Applications, languages, and tools* (Vol. 2, pp. 487–550). Singapore: World Scientific.

Subrahmanian, E., Konda, S., Reich, Y., Westerberg, A., & The N-dim group (1997). Designing the process design process. *Computers & Chemical Engineering*, *21* Suppl., S1–S9.

van der Aalst, W. (1999). Process-oriented architectures for electronic commerce and inter-organizational workflow. *Information Systems*, *24* (8), 639–671.

van der Aalst, W. (2000). Loosely coupled interorganizational workflows: Modeling and analyzing workflows crossing organizational boundaries. *Information and Management*, *37* (2), 67–75.

van der Aalst, W., & Weske, M. (2001). The P2P approach to interorganizational workflows. In K. Dittrich, A. Geppert, & M. Norrie (Eds.), *Proceedings of the 13th International Conference on Advanced Information Systems Engineering*. (CAiSE 2001). LNCS 2068 (pp. 140–156). Interlaken, Switzerland: Springer-Verlag.

Wodtke, D., Weissenfels, J., Weikum, G., Kotz-Dittrich, A., & Muth, P. (1997). The MENTOR workbench for enterprise-wide workflow management. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data* (pp. 576–579). Arizona: ACM Press Tucson.