Integrated Design and Process Technology, **IDPT-2002**
Printed in the United States of America, June, 2002

A Management System Supporting Interorganizational Cooperative Development Processes in Chemical Engineering

Markus Heller
René Wörzberger
RWTH Aachen University, Department of Computer Science 3
Ahornstrasse 55, 52074 Aachen, GERMANY
{***heller,rwoerz***}***@i3.informatik.rwth-aachen.de***

ABSTRACT:

Development processes in chemical engineering are hard to support. In particular, interorganizational development processes are concerned with the coordination of activities carried out by engineering teams distributed across several cooperating organizations.

The management system AHEAD is based on long-term experience gathered in multiple application domains (software, mechanical, or chemical engineering). AHEAD supports the management of dynamic interorganizational development processes by dynamic process views which allow to manage the partial visibility of processes for external organizations. AHEAD provides support for evolving process views which are able to reflect the changes in evolving development processes. Process views serve as the basis for inter-process connection. Processes can be split top-down into subprocesses for different organizations or composed bottom-up of pre-existing processes from different organizations. Mixing top-down and bottom-up process planning and execution is possible.

AHEAD allows process managers to monitor and control their intraorganizational development processes with all interorganizational connections to processes in cooperating organizations in a uniform modeling language.

I. INTRODUCTION

Development processes in different disciplines (software, mechanical, or chemical engineering) are hard to manage. Many changes in all development phases are possible due to changed product requirements, iterations or exceptional feedback to already finished process parts while exploring various design alternatives. Development methods like concurrent or simultaneous engineering require sophisticated coordination between inter-dependent design activities. Because of their highly creative and dynamic character, development processes can only be planned to a limited extent in advance.

These requirements challenge the capabilities of supporting *management systems*. For example, classical workflow management systems [26] support repetitive business processes, e.g. by automating routine work in banks, insurance companies, administrations, etc. According to a common process definition, a high number of workflows are executed in workflow management systems, ensuring that work is performed following a pre-defined procedure. This approach cannot be transferred to development processes, because it does not take process evolution into account: Developers would perceive themselves being tied in a straightjacket so that they cannot perform their creative work as desired.

In particular, today multiple organizations work together in distributed development projects. In general, *interorganizational processes* can be split top-down into subprocesses carried out in different organizations, or they can be composed bottom-up of pre-existing processes. The cooperating organizations carry out their development processes autonomously and are interested in hiding private details of their processes from other organizations. But in order to develop the overall product efficiently with respect to the spent development effort and given time or cost constraints, all participating organizations need to coordinate their work at least to a minimal extent. These two divergent interests of autonomy and coordination have to be *balanced* in distributed processes.

In this paper, we present the view-based support for interorganizational cooperation offered by *AHEAD* [10], an *A*daptable and *H*uman-Centered *E*nvironment for the M*A*nagement of *D*evelopment Processes. AHEAD is based on nearly 11 years of work on development processes in different engineering disciplines. So far, we have applied the concepts underlying the AHEAD system in software engineering, mechanical engineering, and chemical engineering.

Our research focus with respect to interorganizational development processes is based on two fundamental requirements in order to provide flexible and suitable tool support: (1) A powerful and flexible mechanism is needed to control the visibility of internal process information for other cooperating organizations. Confidential information has to be secured, but necessary information has to be exchanged between partners in order to achieve the common goal. (2) Appropriate support for interorganizational development processes is required, where processes of different organizations are integrated with each other not exclusively according to delegation-based relationships between them.

Instead, a set of accepted and customizable process relationships should be supported to fit the needs of various cooperative situations.

In preliminary work ([12], [9]), we have described our first results regarding tool support for interorganizational cooperation between partners in a delegation-based relationship: the client decomposes his overall process into smaller manageable parts and eventually he delegates a process part to a contractor organization for execution. The client can monitor and control the work of the contractor concerning the delegated process. In this first approach, only delegation-relationships for the cooperation between processes are supported which require both parties to have different roles during the collaboration (client and contractor). For example, delegation-based relationships can possibly introduce a hierarchical ordering between organizations insofar as the client is superior to the contractor with respect to the power to define and control the cooperation contents. Other definitions of a delegation between two parties are possible, too. Besides this kind of delegation-based cooperation relationship, there are other possibilities how organizations can work together. For example, organizations can work together as peers having the *same* rights and duties within the cooperation.

As follow-up work, we generalize this model from a more general perspective in order to support a broader spectrum of cooperation scenarios. With the extension described in this paper, AHEAD offers a flexible view-based process visibility model, process and view evolution support, uniform modeling of intra- and interorganizational processes, top-down process decomposition and bottom-up process composition, process model inconsistency detection and toleration, and contract-based support for multiple cooperation scenarios.

We proceed as follows: Section II describes the process modeling approach of the AHEAD system and introduces the concept of dynamic process views for process visibility management. Section III presents our approach to support interorganizational development processes based on dynamic process views. Section IV describes the support of the AHEAD system by using a case study from chemical engineering (due to lack of space we cannot cover other engineering disciplines in this paper as well). Section V describes the architecture and implementation of AHEAD. In section VI we shortly discuss related work. A conclusion is given in section VII.

II. Fundamental Concepts

Before we present our view-based cooperation model to support interorganizational development processes, we introduce the process modeling approach used in AHEAD and the concept of dynamic process views in the next two subsections.

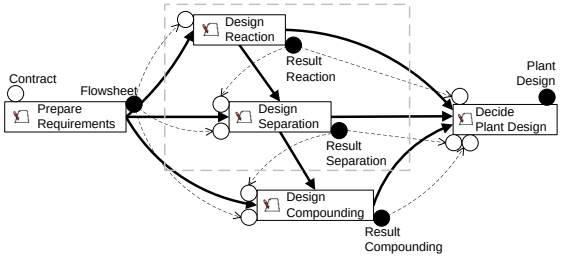

Fig. 1. Top-level task net for the Polyamide6 development process

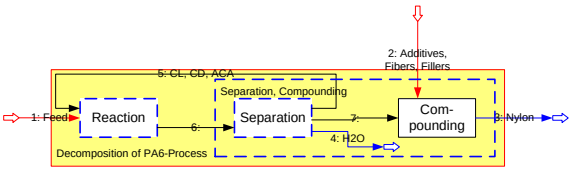

Fig. 2. Flow sheet for the chemical process

A. Dynamic Task Nets in AHEAD

We have developed AHEAD within the context of IMPROVE [15], a long-term running research project which is concerned with models and tools for development processes in *chemical engineering*. Previous papers on the AHEAD system have presented its core features, its support for process evolution [10] and a preliminary approach to support interorganizational development processes in chemical engineering based on a delegation-relationship between two organizations [9]. Within IMPROVE, we have studied a reference scenario referring to the early phases (conceptual design and basic engineering) of developing a plant for producing Polyamide6 [16].

In AHEAD, development processes are modeled as *dynamic task nets* [8]. Tasks are organized in a hierarchy. Complex tasks are decomposed into nets of subtasks while atomic tasks are the leaves of the hierarchy. Tasks have an execution state and a resource can be assigned to each task as its performer. Control flows can be used to connect tasks horizontally to determine the order of task execution. Tasks can have inputs and outputs which are connected by data flows. Task nets are created and modified at run time by inserting new tasks. Planning and execution may be interleaved seamlessly. In contrast to project plans known from project management systems, task nets may represent feedback in the development process. Handling of feedback may imply reactivation of already terminated tasks in order to propagate changes through the task net. Finally, AHEAD provides simultaneous engineering by control flows which allow for overlapping execution of predecessor and successor tasks. Preliminary versions of outputs may be released to and consumed by successor tasks, resulting in a dynamic workspace which is updated according to changes in the context of a task.

For instance, the top-level dynamic task net for the Polyamide6 process of the IMPROVE case study is shown in the upper part of Figure 1. Tasks are denoted as boxes with a task name, a performer name, and an icon for the execution state. Sequential control flows between tasks are shown as solid arcs. Output and input parameters of tasks (denoted as circles) are connected by data flows (dotted arcs). The process starts with a pre-study where the requirements for the plant design are collected and an initial flow sheet for the chemical process is drawn (Figure 2). After that, several tasks are planned to study the chemical reaction in more detail: reaction of monomers, separation of polymers from monomers, which are fed back into the reaction phase, and compounding, which is concerned with fine-tuning the properties of the produced material. Finally, in an evaluation step it is determined if the desired requirements to the chemical process are met. If not, the design developed so far is changed and the design process is iterated. The process has not started yet and all tasks are in their initial state. The Polyamide6 process is used in the example in section IV again.

B. Dynamic Process Views

Based on dynamic task nets, a *dynamic process view* is defined for a process instance with its products and resources. The main purpose of a process view is to define a certain cut-out of its underlying process, its products, and its resources. A process view contains mainly three elements: a subgraph (or partial abstraction) of a dynamic task net, an own document workspace to store documents or document versions which are visible in the view, and a view resource set (with all plan resources and actual resources which are visible within the view). A process view has some additional attributes: a unique view identification and a name, a view definition rule set which defines the contents of the view, and some other attributes not relevant in this overview paper. Figure 3 illustrates this by an example showing a cut-out of the process of the Chemical Engineering Company shown in the top part of the Figure where the task Design Reaction is refined by a subnet with four child tasks (drawn below their parent task).

The process-related elements of a process view constitute a structurally and behaviorally consistent dynamic task net according to the process model of dynamic task nets. For example, input or output parameters contained in a view always require their tasks to be in the view as well, source and target tasks of control flows in a view have to be in the view, etc. Zero or more tasks of a task net can be in the view, not all of a task's parameters have to be in the view and not necessarily all flow relationships between tasks in a view have to be in the view.

A process view is described by a set of rules called *view definition*. These rules define which parts of the overall process configuration should be visible in the view. For example, a view definition can contain rules to include tasks

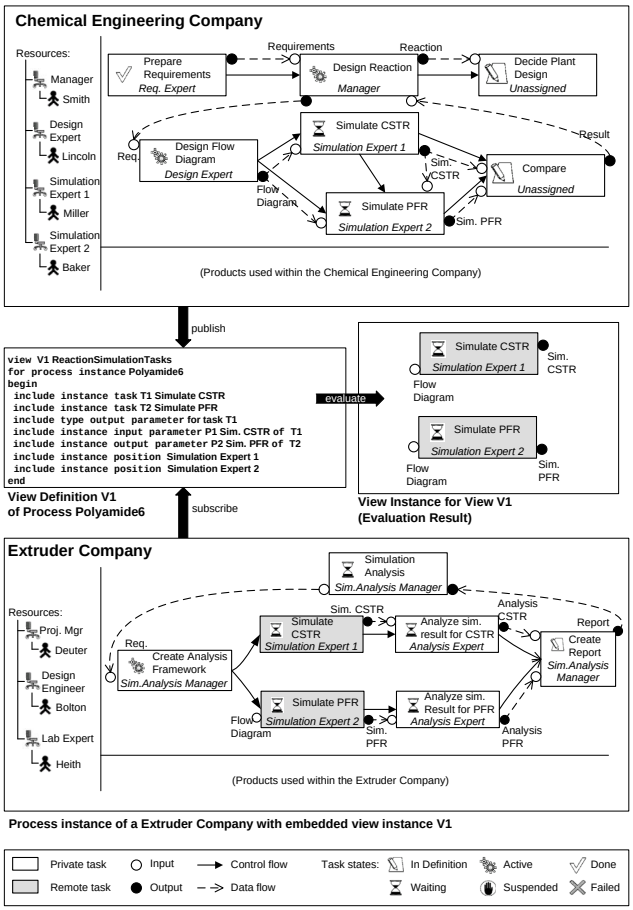

Fig. 3. Dynamic process view for the polyamide6 process

(resp. documents, resources, or flow relationships) or all tasks (resp. documents, resources, or flow relationships) of a specific type in a view. Figure 3 shows a process view definition named ReactionSimulationTasks containing only two simulations tasks of the overall process, while the control flow between bot tasks is not included in the process view. The application of a view definition to a process results in a *view instance*. A certain view instance can change due to two reasons: (1) changes of the underlying process, and (2) changes in the view definition. Therefore, view instances have to be updated continuously to be consistent with the underlying process and the corresponding view definition.

Different process views provide access to the process from different perspectives, for example: views for managers neglecting some irrelevant process details, views for other parties to observe specific parts of the overall process, etc. A process instance can have an arbitrary number of process views attached to it. A view-based approach constitutes a natural and promising approach for managing the visibility of private process elements and the access to them by external parties.

Dynamic process views provide the basis for interorganizational cooperation, because elements of process views of different organizations can be inter- connected with each other as follows. Process view definitions are published by one organization and they can subsequently be subscribed by other organizations. The view instances as the result of the view definition's evaluation are embedded in the private process instances (lower part of Figure 3).

As a result, private processes can contain local tasks as well as remote tasks embedded from other organizations within process views. We re-use the control flow, feedback flow, and data flow concepts of dynamic task nets in our approach to model interorganizational process integration as well. We believe, that modeling intra-organizational as well as interorganizational cooperation in a uniform way is feasible and should be supported by an approach that is simple to understand and use by process managers.

III. View-based Interorganizational Cooperation Model

A. Conceptual Layers of the Cooperation Model

We now present our approach to support interorganizational development processes based on dynamic process views for processes. We describe this model according to three layers which are located on top of each other starting at the bottom of the layer stack (Figure 4):

A.1 Private process layer

This layer contains the private processes of organizations containing confidential information which is usually not shared completely with other organizations. Multiple process views from different organizations can be subscribed by a private process in order to import process elements from other remote organizations. After that, local and remote process elements can be connected with each other by control flows, feedback flows or data flows to establish inter-process cooperation.

A.2 Process view layer

This layer contains all process views which are published by different organizations. In our approach, process view instances from other organizations are embedded into private processes. This enables process managers to oversee the whole process instance with all relevant process aspects including connections to external processes within a single dynamic task net. The task net always provides an accurate and comprehensive process overview for process managers. This approach is feasible within our application domain of development processes in chemical engineering, because the private process instance can be modified during the course of the development process. Therefore, we need not introduce additional cooperative processes where interorganizational cooperation aspects are represented.

A.3 Cooperation layer

In the cooperation layer, we put our focus on different relationships that can exist between organizations – or precisely, between the processes carried out in organizations – concerning their collaboration upon a special matter of discourse (for example, a task net fragment). On the cooperation layer, we introduce three different kinds of *basic* cooperation relationships between processes as follows:

- *Monitoring relationships* between processes of different organizations indicate, that one process (termed as observer) has embedded a certain view of another process (termed as observed) in another organization right within its own private process. This allows for the observation of process information transferred from other organizations. Monitoring relationships do not allow interaction between the observing and observed processes.
- *Interaction relationships* model the exchange of task state information and data information between both processes by control or feedback flows or data flows between tasks in processes of different organizations. All state transitions of tasks in a private process – either local tasks or tasks embedded from remote different organizations – can be restricted by ingoing control or data flows of local or remote predecessor tasks and they can restrict states of local or remote tasks by outgoing control or data flows. This allows to interweave different processes in the sense that they are executed in parallel while they are coupled in a loosely coupled way at the same time.
- *Outsourcing relationships* model cooperation in a customer-producer manner, where an organization (termed as customer) assigns single process tasks or a task net fragment to another organization (termed as producer) for execution. The task is then transferred to the other organization and regarded there as a local task in the future. The task is executed within the other organization and only observed from the outsourcing organization which is not responsible for the task any more. So in an extreme scenario, the receiving organization could once decide to change the task's visibility from public to private again so that even the outsourcing organization could not observe the task any more. Outsourcing can be interpreted as the transfer of certain authorization rights between organizations, for example, the right to define and execute a task is transferred from the customer to the producer. Cooperation happening without outsourcing (or delegation) at all represents a broadly seen and useful scenario in interorganizational cooperation, for example, if process parts are only observed by partner organizations in order to keep processes synchronized with respect to a common time-schedule.

B. Contracts

In an interorganizational process, different cooperation relationships may be appropriate depending on the level of trust between the partners. For example, organizations

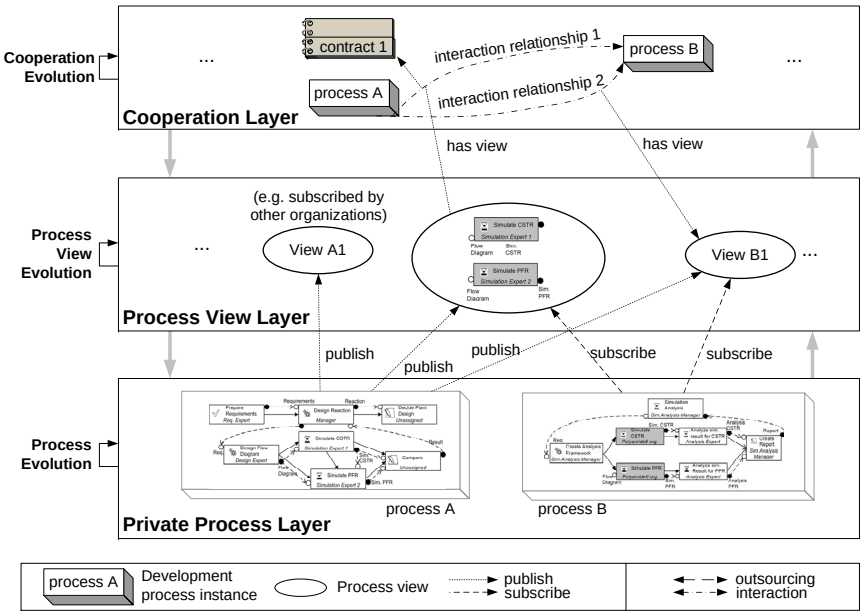

Fig. 4. Layers of the view-based interorganizational cooperation model

working together in a long-term cooperation often trust each other and thus cooperate in less formal relationships. For example, they communicate frequently to discuss problems or exchange preliminary results to speed up the overall process. In contrast, a formal setting with strict contracts may be chosen for the outsourcing of process parts, or when the partners have not built up sufficient trust between each other yet. In general, *contracts* are used for detailing cooperation between partners according to individual needs. Contracts allow for the definition of negotiable aspects of a cooperation engagement between partners. For example, the object of discourse (e.g. a task net fragment), the different roles played by the contract partners together with their rights and duties, and other contract data are defined.

Currently, in the delegation-based approach only one fixed contract protocol between client and contractor organizations is implemented. It is our goal to support the individual configuration of contracts to support a broader spectrum of possible cooperation scenarios between the partners. On the cooperation layer, all three basic cooperation relationships (monitoring, interaction, and outsourcing) between processes defined in lower layers, can optionally be refined by contracts. Our basic idea is to implement a very light-weight default contract protocol for the cooperation between partners and to provide contracts as a means to further define the fine-grained structure of a cooperation between partners if needed.

Contracts contain zero or more process views published by the contract partners and additional contract data. Currently, we concentrate on the contract details regarding the structure of the contracted process fragments and therefore we only require that outsourcing relationships are always accompanied by a contract setting up an *contract protocol* for the communication between the contract partners. For example, a contract protocol for the communication between partners regarding contract changes can be established as follows: Partner A signals Changes requested which is answered by a partner B signaling either Changes allowed and Changes forbidden. If allowed, partner A can process with the changes and can signal Changes Finished afterwards and so on. The exact structure of these protocols can be explicitly defined in our approach. Currently, we give the state transition diagram for a protocol within an XML file. We are currently developing a set of feasible contract protocols concerning the initiation, execution and modification, and the final evaluation of the outsourcing (or delegation). These protocols can be used to define cooperation relationships in a fine-grained manner.

C. Phases of the View-based Cooperation Model

Our approach to interorganizational view-based cooperation management defines the following cooperation phases:
1. The process manager of an organization defines its private process as a dynamic task net. He can decompose each task locally into sub-tasks and can assign each task to a performing resource within the organization.
2. Parts of intra-organizational processes can be made visible for external organizations by the process manager. He creates and publishes process view definitions which contain a set of (type-level and instance- level) rules defining the process cut-out to be published. Selected tasks of a process view can be *outsourced* to other organizations.
3. Published process view definitions can be subscribed by process managers of an organization and embedded into

their private processes by managers of other organizations. Subsequently, inter-connections between local and remote process elements can be established to form an interorganizational process.
4. On the cooperation layer, the process manager can model the intented cooperation relationships and attach any locally existing process view definitions to them. Optionally, he can refine each relationship with a contract.
5. The processes are executed within their respective organizations. The corresponding AHEAD systems are coupled at run-time to inform each other about process changes with respect to interorganizational process relationships. If the related process instances evolve in all organizations, the corresponding process view instances embedded elsewhere are updated accordingly by a view maintenance mechanism. The evolution of view definitions is possible by adding, deleting or modifying view definition rules, and this leads to the re-evaluation of the rule set and necessary change messages are sent to all AHEAD systems where the respective view instances are re-embedded into the private process.
6. The previous phases can be repeated during the course of the development process with process planning and execution being interwoven with each other seamlessly.
7. At last, process interconnections can be stopped by the process managers of all related organizations by withdrawing the published process views definitions. The processes evolve independently of each other after the interconnection has been terminated.

D. Features of the View-based Cooperation Model

In summary, the new view-based cooperation model can be characterized by the following features:
• *Dynamic process view model*. As a prerequisite to interorganizational cooperation, we introduce dynamic process views to allow access to processes by external organizations. Process views allow for balancing between the conflicting requirements of the organizations of protecting the privacy and enactment autonomy of their processes and sharing processes with other organizations for cooperation. The process view concept offers a high degree of flexibility for publishing different process views tailored to different information needs, e.g. requested by different cooperation partners.
• *Process view evolution on definition-level and instance-level*. Because planning and enactment on dynamic task nets may be interleaved seamlessly (process evolution), evolution capability is a core requirement for process view instances. First, process views evolve on the instance-level in order to always keep them consistent with the underlying process by re-evaluating the view definitions upon process changes. Second, view definitions evolve by adding, deleting, or modifying its definition rules. After modifications, the view definitions are re-evaluated to update the view instances according to the current set of view definition rules.
• *Uniform modeling of processes and process interconnection*. By using process views, we are able to offer a single unform basic concept on which interorganizational cooperation can be based. Organizations can use views to connect their process with processes carried out in other organizations. As stated above, process views can be subscribed and embedded into processes, and subsequently process elements of different organizations can be connected with modeling elements of dynamic task nets. In this way, private processes and the interorganizational connection of processes is modeled in a uniform way. This eases the modeling task for the process managers and avoids the use of a second modeling language for process interconnection.
• *Top-down process decomposition and bottom-up process composition*. Processes can be hierarchically decomposed *top-down* into sub-processes and subsequently distributed across organizations using process views. Alternatively, they can be composed *bottom-up* by importing dynamic process views of pre-existing processes in other organizations into a private process and connecting process elements of different organizations subsequently.
• *Conformance monitoring and toleration*. Process views can always be subscribed and embedded into any process and process view elements and private process elements can be connected with modeling elements of dynamic task nets. Because the connected processes can evolve autonomously within their organizations, the inter-connection elements can eventually violate behavioral or structural constraints of dynamic task nets. These deviations are reported to the process managers who may decide to reinforce behavioral or structural conformance or to tolerate the violation.
• *Contract-based support for multiple cooperation scenarios*. Different scenarios are possible for the cooperation between processes carried out in different organizations, for example, monitoring of processes, interaction between organizations or outsourcing scenarios where working tasks are delegated by one organization (customer) to another organization (producer) for execution. Contracts can be defined and enforced to fix agreements between organizations about important cooperation aspects of the cooperation, like the different organizational roles with rights and duties, the involved process views, as well as different kinds of cooperation policies for changes of the contract or related process view definitions. Additional contract parameters like cost or time schedules can be attached to existing contracts (not detailed in this paper).

IV. Example

After having presented the view-based approach to interorganizational cooperation support of the AHEAD system, we now give an example which is derived from the IMPROVE case study reference scenario referring to the early phases (conceptual design and basic engineering) of developing a plant for producing Polyamide6. The require-

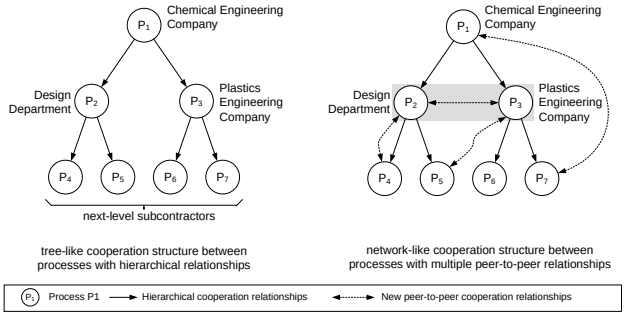

Fig. 5. Cooperation relationships in the example scenario

ments for the support of interorganizational processes have been drawn from this case study. We are showing an example which is capable of demonstrating the cooperation support features of AHEAD. The overall development process is much more complex (as described in [16]).

A. Initial Situation

The example focuses on interorganizational cooperation between two processes which are carried out in different organizations. Both processes are integrated with each other to show how top-down process decomposition and bottom-up process composition are performed. We deal with the part of the overall development process which is related to the design of the reaction and separation as well as the design of the extruder. The Chemical Engineering Company acts as a customer and outsources the task of designing the reaction and separation to another organization, the Design Department having an own project manager who manages his own process, products and resources independently. The task to design the extruder component is outsourced to an external Plastics Engineering Company. Of course, it is also possible that reaction and separation design are contracted to an external organization, when the Chemical Engineering Company only acts as a project manager for the overall development project.

In the following, we neglect the establishing of the outsourcing relationships between the Chemical Engineering Company and its contractors, the Design Department and Plastics Engineering Company. For didactic reasons, we will explain later how such outsourcing relationships between customer and producer are established by using process views. Within the example scenario, we will take the resulting situation as our starting point in order to highlight the direct cooperation between both contractors (right part of Figure 5).

This kind of direct cooperation between organizations in a peer-to-peer mode is not supported in the delegation-based concept where both contractors cannot cooperate with each other directly but only through their common client, the chemical company (shown in the left part of Figure 5). Although the delegation-based process decomposition approach is sufficient in many situations, often direct cooperation between all partners of a cooperative *network* of companies is needed as well.

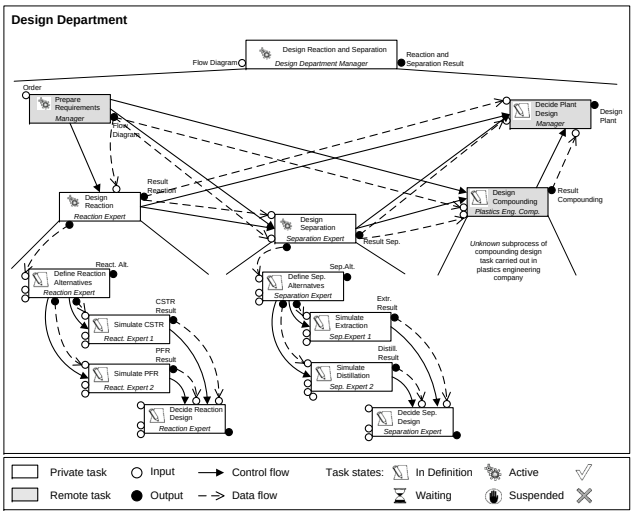

Fig. 6. Initial task net of the process in the Design Department

Figure 6 shows the initial situation after the manager of the Design Department has received the delegated task net from the Chemical Engineering Company. As stated above, the reaction and separation aspects are investigated on the side of the Design Department and its manager has already refined the flow sheet with different alternatives for the chemical reaction and separation process and has created tasks to further investigate all alternatives (tasks Simulate CSTR, Simulate PFR, Simulate Extraction, Simulate Degassing).

On the side of the Plastics Engineering Company, the compounding task is already refined by a subnet (for clarity, this is the same subnet as in the example of the delegation-based concept): 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. 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 three 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.

B. View Definition, Publication and Subscription

Two dynamic process view definitions V1 and V2 are created which contain selected portions of the private process on the side of the Design Department (part *a* in the upper part of Figure 7). The view definition V1 PFR-Information (shown left in part *b* of the Figure) contains tasks of the Design Reaction subnet which are especially

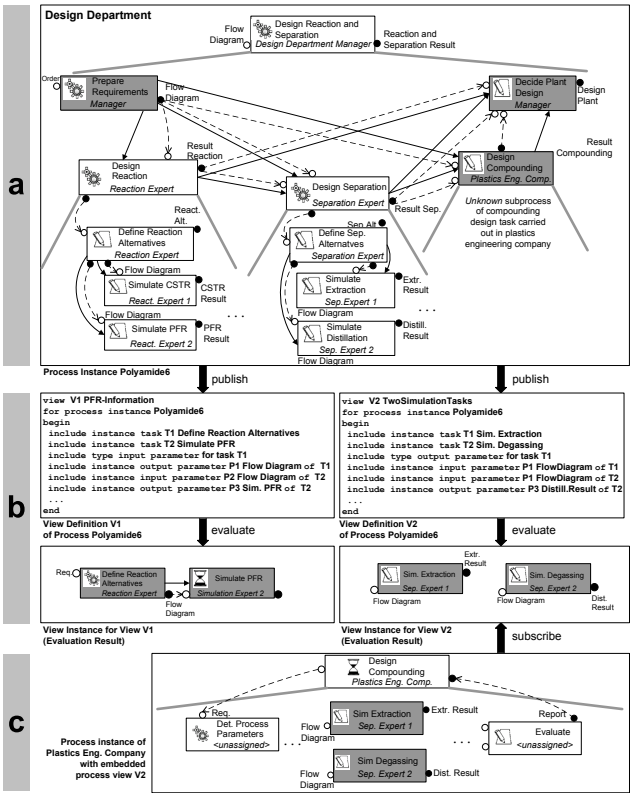

Fig. 7. Definition, Publication and Subscription of Process Views

related to the simulation of the plug flow reactor (Simulate PFR). This view definition can be published and then subscribed by interested organizations with information needs regarding this cut-out of the reaction.

We will only proceed in the demonstration with the process view definition V2 TwoSimulationTasks (shown right in part *b* of the Figure) containing the simulation tasks Simulate Extraction and Simulate Degassing with parameters. Process view V2 is published by the Design Department manager. After that, the view definition V2 is subscribed by the manager of the Plastics Engineering Company and embedded into his private process (part *c* of Figure 7). Similarly, the first process view V1 could be subscribed by the manager of the Plastics Engineering Company and embedded into his private process. This would result in the situation, that both processes are connected at two different locations which can be planned and evolved independently.

The management systems of both organizations are coupled together with a communication server in between to exchange change events with each other. The partners are informed about changes of externally visible elements in the processes of the cooperation partners (process evolution). Please note, that the view definition itself is not changed, only the view instance is re-evaluated to reflect the changes. For instance, if Define Reaction Alternatives is changed from `In Definition` to `Waiting` on the side of the Design Department, this state change is reflected in all view instances (embedded in processes of other organizations) of the view definition V1.

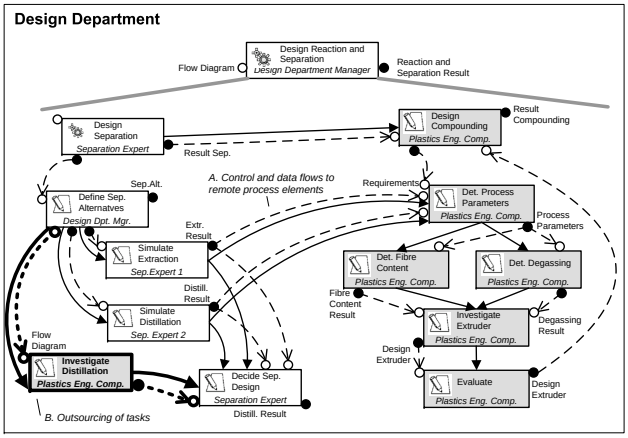

Fig. 8. Interconnections between Design Department process and Plastics Engineering process

C. Bottom-Up Process Composition

Although both process instances in the Design Department and the Plastics Engineering Company are already running in parallel, they can be integrated with each other (process composition). The manager of the Design Department now wants to connect two simulation tasks Simulate Extraction and Simulate Distillation with the task Determine Process Parameters of the other organization in order to synchronize the tasks in both organizations with respect to their execution states and to exchange documents between them (see mark *A.* in Figure 8). For this reason, the manager inserts new control flow and data flow dependencies between these tasks in his private task net. These changes can cause two different consequences. First, locally relevant inter-process dependencies between tasks (going out of a remote task into a local task) do not cause problems since they do not impose new behavioral restrictions on the remote tasks. Second, remotely relevant inter-process dependencies (going from a local task into a remote task) are problematic. Because the new control flows are going into the remote task Determine Process Parameters, the second alternative applies here and the intended changes are only allowed if the manager of the Plastics Engineering Company agrees to them.

By re-using an already existing process view definition or by creating a new process view definition, the intended changes are published to the manager of the Design Department as follows. The view definition of the used process view V1 is modified accordingly (view evolution) and published afterwards (view definition evolution) by the manager of the Design Department. On the other side,

the manager of the Plastics Engineering Company accepts the changes, they become persistent. Otherwise, the manager of the Plastics Engineering Company can either make modifications to the changed task net fragment under discussion, or can choose to remove them if no consensus is possible. After the changes have been carried out in both management systems, the managed process instances are coupled with each other.

After these changes, both process instances can be executed and evolved by the process managers of both organizations, while they are coupled at least through newly inserted control and data flows between elements of both processes at the same time.

D. Top-Down Process Decomposition with Outsourcing

The manager of the Design Department recognizes, that the separation alternatives in the separation step are better checked by the Plastics Engineering Company very early in this project. He inserts a new task Investigate Distillation together with its parameters (see mark *B.* in Figure 8). He creates a new process view definition V3a and inserts the new task to it together with its connections to the tasks Define Separation Alternatives and Decide Separation Design. Then he marks the task as outsourced in the view and calls a command to add a minimal context of the outsourced task in order to maintain consistency with the surrounding task net. In the example, the context comprises the predecessor task Defines Separation Alternatives and the parameter Flow Diagram as well as the control and data flows to task Investigate Distillation.

The process view V3a is published by the manager of the Design Department. At this moment, the marking outsourced is detected and the system uses a special *outsourcing procedure* on both sides: First, the manager of the Plastics Engineering Company subscribes the process view V3a in the usual way. He then accepts the outsourced task. Second, a new process view definition V3b is created and filled with the outsourced task and its context, where V3a and V3b are structurally the same, but the role of local tasks and remote tasks is reversed in the view. Third, the manager of the Design Department has to accept this process view V3b in order to signal, that he agrees to the partner who has offered to execute the work regarding the outsourced task Investigate Distillation. Now the outsourcing relationship between both processes is fully established. The outsourced task Investigate Distillation is executed by the Plastics Engineering Company (it is a local task there) while the other context tasks are executed by the Design Department as before.

Currently, there is a set of process views in use between the cooperating organizations. Figure 9 shows the involved processes of the organizations and their cooperation relationships from the perspective of the Design Department. This *collaboration graph* is also presented to the manager of the Design Department upon request. The

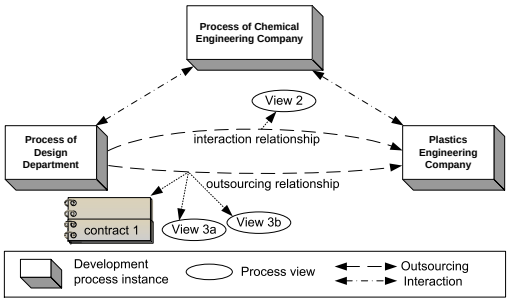

Fig. 9. Cooperation relationships between participating organizations

process Polyamide6 has an outsourcing relationship with the process Compounding carried out by the Plastics Engineering Company through the process views V3a and V3b. Additionally, there exists an interaction relationship between the Design Department and the Plastics Engineering Company through the view V2.

Finally, the manager of the Plastics Engineering Company states that the outsourced work is done. He sets the task Investigate Distillation to Done and initiates an outsourcing evaluation protocol. The protocol regulates the exchange of messages between the customer and producer organization in the outsourcing relationship. Currently, we use a light-weight protocol with just three possible messages: the producer sends Outsourcing Completed and the customer can react upon this by either sending Accepted or Rejected. Rejection means, that the outsourcing relationship between both partners is not finished yet. If the customer accepts the intended end of the outsourcing relationship, the views V3a and V3b are closed by the management system and thus cannot be modified any more.

V. REALIZATION

Figure 10 displays the *architecture* of AHEAD. The tools provided for different kinds of users are shown: “Process Manager”, and “Developer” denote roles rather than persons: A single person may play multiple logical roles, and a single role may be played by multiple persons.

The lower part of the Figure shows internal components of the AHEAD system which are not visible at the user interface. Internally, AHEAD is based on a formal specification as a programmed graph rewriting system. To this end, we use the specification language *PROGRES* as well as its development environment, which offers a graphical editor, an analyzer, an interpreter and a code generator [18]. The application logic of AHEAD is fully specified in PROGRES and it was created once by the tool builders of AHEAD. That is, the users of AHEAD are not aware of the PROGRES specification which is employed internally.

The overall AHEAD specification is automatically translated into C code. The generated code constitutes the application logic of the tools and it operates on the management configuration data which are stored in a graph-based Data-

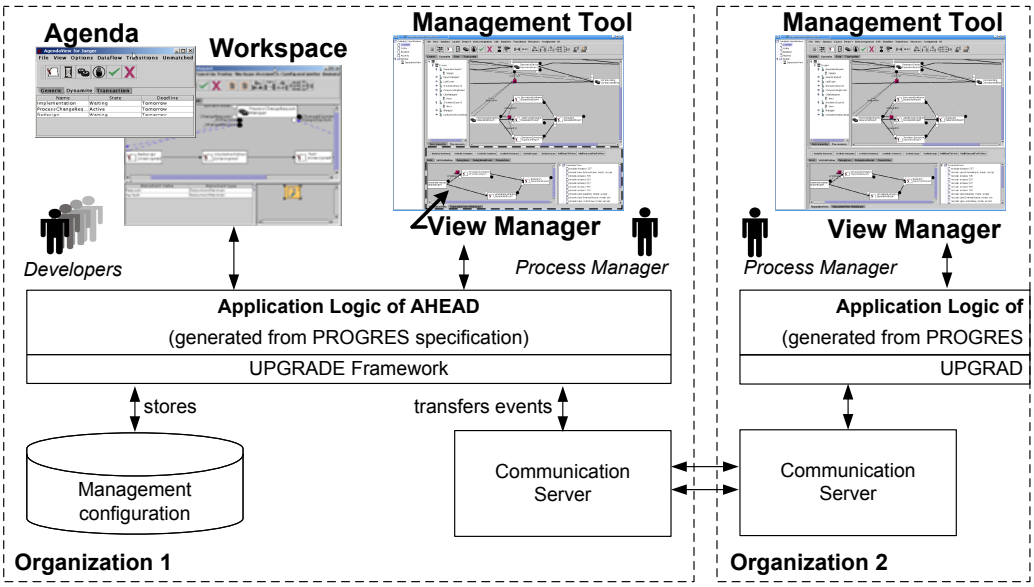

Fig. 10. Architecture of the AHEAD System for interorganizational process support

base Management System. The user interface of AHEAD is implemented with *UPGRADE*, a framework for building graph-based interactive tools [2].

At the user interface, AHEAD provides a *manager tool* for the process manager. The tool assists a process manager in planning, analyzing, monitoring, and controlling a development process. Additionally, all created products and all managed process resources are presented to the manager. Process views can be defined and controlled in a new View Manager tool. A screenshot from the management tool including the View Manager is shown in Figure 11. For developers, the developer environment offers an *agenda* with all currently assigned tasks as well as a *work context* with detailed process information for a selected task. From within the work context, domain-specific tools may be activated in order to work on design documents, e.g. simulation tools.

Instances of AHEAD in different organizations coupled via (distributed) communication server which exchange messages to keep each other informed about process changes. Events are sent from one system to coupled systems after each operation of AHEAD, if the effect of the operation is relevant to other systems. The communication servers also handle the management of views (publication of views, subscription, embedding). The CORBA-based communication server uses the existing event-handling mechanism in AHEAD which was modified and extended for the new view-based concept. The detailed graph-based realization of the event mechanism is given in [9]. The View Manager tool allow process managers to create process views and and publish them to the communication server, Via the communication server, process managers can subscribe to process views and embed them into their private process.

VI. Related Work

In this section, we discuss relevant related work in the field of workflow support for interorganizational processes, view-based workflow approaches, and communication-oriented approaches to cooperation modeling.

In general, *Workflow management systems* [11], [13], e.g., Staffware, FlowMark, or COSA, have been applied in banks, insurance companies, administrations, etc. Their most important restriction is limited support for the dynamics of design processes. Many workflow management systems assume a statically defined workflow that cannot be changed during execution. In this way, dynamic design processes can be supported only to a limited extent (i.e., the statically known fractions can be handled by the workflow management system). Recently, this problem has been addressed in a few university prototypes (see e.g. [5], [7]).

Many approaches to process management support are *domain-independent*. For example, workflow management systems may be applied to arbitrary business processes. We are aware of only a few *domain-specific* approaches which have been developed for chemical engineering. For example, n-dim [19] is a distributed and collaborative computer-aided environment for process engineering design; KBDS [1] deals with the management of design alternatives and design histories. These approaches are better tailored towards design processes in chemical engineering. However, they do not provide comprehensive support for the management of products, activities, and resources. Moreover, they lack the generality of domain-independent systems which can be used in and adapted to different domains.

Management of *distributed processes* is addressed by a number of workflow management systems. But a distrib-

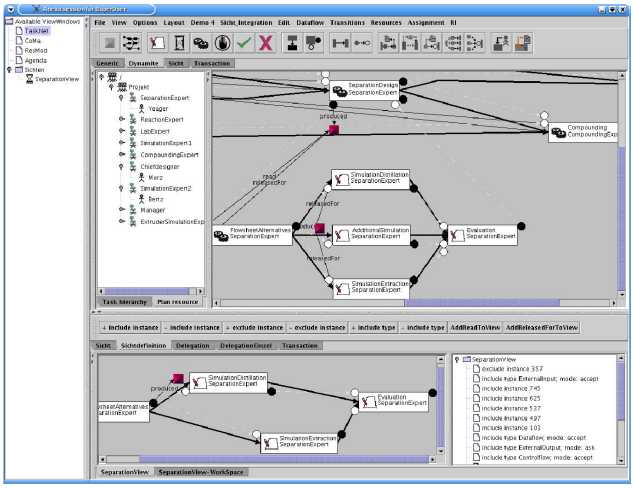

Fig. 11. Screenshot from the management tool with view manager

uted 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* [25] is a workflow management system which provides multiple workflow servers. Work is distributed among the workflow servers according to a sophisticated load balancing algorithm.

[22] provides an overview of paradigms for *interorganizational processes*. Among others, the following two paradigms are identified: (1) With *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. (2) *Loosely coupled processes* are executed in parallel in different organizations. Occasionally, they interact at pre-defined communication and synchronization points.

The work of [22] primarily focuses on case transfer and an extended variant thereof. In [23], the same author discusses loosely coupled processes. The interaction paradigm subcontracting is supported by the standards defined by the Workflow Management Coalition (WfMC [13]). In addition, subcontracting was introduced as early as 1987 by the Istar system [6] in the software engineering domain.

The previously developed *delegation-based cooperation model* [12] of the AHEAD system is focusing on the subcontracting paradigm only. Delegation differs from loosely coupled processes because there is a hierarchical relationship between client and contractor. With the extension of AHEAD by the view-based cooperation model described in this paper, we can support a broader spectrum of cooperation scenarios covering also loosely coupled processes where the processes are peer to peer in general.

Van der Aalst [21] focuses on independently running but loosely coupled interorganizational workflow processes modeled in a language based on Petri-Nets. A predefined and static communication structure between the private processes of the partners is assumed in this approach. Van der Aalst and Weske [24] define a public workflow containing process elements for interest to all partners and split it up into private workflows distributed to each partner. In these two contributions, a top-down approach is followed which is feasible if the overall process structure is known in advance. In our application domain of dynamic development processes in chemical engineering, this is not feasible in all situations, since development processes cannot be planned fully in advance and evolve with the time. Therefore, we regard it more feasible to allow for both a top-down and a bottom-up strategy where the interorganizational process integration points can (but need not) be defined beforehand and where it is possible for every organization to introduce new integration points to processes from other organizations at any time.

Regarding the modeling of the process aspects where different processes are inter-connected, several approaches are possible. For example, Perrin et al. [17] present a model of *synchronization points* in interorganizational development processes. Synchronization points are not integrated into a modeling language. Instead, a separate independent modeling language is proposed for flexibility reasons and thus this second language remains independent of the used underlying process definition language. In our approach, all process aspects are modeled in a uniform way as dynamic task nets to allow process managers a complete overview over all process details including interorganizational process connections in one single representation. So we do not use a separate model for cooperation modeling and we re-use the control flow, feedback flow, and data flow modeling elements of the dynamic task nets modeling language for this purpose.

Regarding process visibility management, the works of Liu and Shen [14], Chiu et al. [4], and Tata, Chebbi et al. ([3], [20]) contain similar view-based approaches for workflow support with a different focus on support for interorganizational routine business processes which are not structurally changed at run-time. First, they re-use workflow definitions to act as views for other (private) workflow definitions in order to protect private details of the workflows. The process definitions (termed views) provided by each partner are not changed after the start of the overall workflow, so that views and their underlying instances cannot become inconsistent during run-time. Second, the mentioned approaches aim at minimizing necessary modifications of the private workflow definitions. This eases rapid composition of business processes from pre-existing processes as another goal of these approaches. Additionally, the mentioned approaches do not focus on the interleaved definition and execution of process and views.

The view-based cooperation model in AHEAD also has related work in the research field of communication-oriented interorganizational cooperation. Inspired by the authors, our cooperation layer introduces three different cooperation relationships (monitoring, interaction, and outsourcing) as well as contracts for defining the formal guidelines structuring the cooperation.

VII. Conclusion

We have presented a management system which supports seamless interleaving of planning and execution of evolving development processes. Our approach is applied to development processes in different engineering disciplines (software, mechanical, or chemical engineering). In this paper, we have focused on chemical engineering as studied in the IMPROVE project. Based on the study of development processes in chemical engineering, we have developed a cooperation model to support interorganizational development processes grounded on a view-based approach to process visibility management covering a broad spectrum of cooperation scenarios. The AHEAD system has been applied to the reference scenario studied in the IMPROVE project. Future work will address the detailed study of mixed cooperation scenarios between organizations, the extension of the process view concept for abstraction concepts, e.g. aggregation of multiple tasks within process views, and the extension of the contract-based cooperation support.

VIII. Acknowledgments

Financial support of Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center 476 IMPROVE is gratefully acknowledged.

References

[1] R. Bañares-Alcántara and H. Lababidi. Design Support Systems for Process Engineering – II. KBDS: An Experimental Prototype. *Computers & Chemical Engineering*, 19:3:279–301, 1995.

[2] B. Böhlen, D. Jäger, A. Schleicher, and B. Westfechtel. UPGRADE: Building Interactive Tools for Visual Languages. In *Proceedings of the 6th World Multi-Conference On Systemics, Cybernetics and Informatics (SCI 2002), Orlando, Florida, USA*, pages 17–22, 2002.

[3] I. Chebbi, S. Tata, and S. Dustdar. The view-based approach to dynamic inter-organizational workflow cooperation. Technical Report TUV-1841-2004-23, Technical University of Vienna, Information Systems Institute, Distributed Systems Group, 2004.

[4] D. K. W. Chiu, K. Karlapalem, and Q. Li. Views for inter-organizational workflow in an e-commerce environment. In *Proceedings of the IFIP TC2/WG2.6 9th Working Conference on Database Semantics*, pages 137–151, Deventer, The Netherlands, 2003. Kluwer, B.V.

[5] J.-C. Derniame, A. K. Baba, and D. Wastell, editors. *Software Process: Principles, Methodology, and Technology*, volume 1500 of *LNCS*. Springer, 1998.

[6] M. Dowson. Integrated Project Support with Istar. *IEEE Software*, 4:6–15, 1987.

[7] D. Georgakopoulos, W. Prinz, and A. L. Wolf, editors. *Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration (WACC-99), San Francisco, CA, USA*, volume 24:2 of *ACM SIGSOFT Software Engineering Notes*. ACM Press, 1999.

[8] P. Heimann, G. Joeris, C.-A. Krapp, and B. Westfechtel. DYNAMITE: Dynamic Task Nets for Software Process Management. In *Proceedings of the 18th International Conference on Software Engineering, Berlin, Germany*, pages 331–341. IEEE Computer Society Press, 1996.

[9] M. Heller, D. Jäger, M. Schlüter, R. Schneider, and B. Westfechtel. A Management System for Dynamic and Interorganizational Design Processes in Chemical Engineering. *Computers & Chemical Engineering*, 29:1:93–111, 2004.

[10] M. Heller, A. Schleicher, and B. Westfechtel. A Management System for Evolving Development Processes. In *Proceedings of the 7th International Conference on Integrated Design and Process Technology (IDPT 2003), Austin, Texas, USA, 10 pp.* SDPS, 2003.

[11] S. Jablonski and C. Bußler. *Workflow Management — Modeling Concepts and Architecture*. International Thomson Publishing, 1996.

[12] D. Jäger. *Unterstützung übergreifender Kooperation in komplexen Entwicklungsprozessen*. PhD thesis, RWTH Aachen University, Wissenschaftsverlag Mainz, 2003.

[13] P. Lawrence, editor. *Workflow Handbook*. John Wiley & Sons, 1997.

[14] D.-R. Liu and M. Shen. Business-to-business workflow interoperation based on process-views. *Decision Support Systems*, 38(3):399–419, 2004.

[15] M. Nagl and W. Marquardt. SFB 476 IMPROVE: Informatische Unterstützung übergreifender Entwicklungsprozesse in der Verfahrenstechnik. In M. Jarke, K. Pohl, and K. Pasedach, editors, *Informatik als Innovationsmotor (GI Jahrestagung 97)*, Informatik Aktuell, pages 143–154. Springer, 1997.

[16] M. Nagl, B. Westfechtel, and R. Schneider. Tool Support for the Management of Design Processes in Chemical Engineering. *Computers & Chemical Engineering*, 27:2:175–197, 2003.

[17] O. Perrin, F. Wynen, J. Bitcheva, and C. Godart. A model to support collaborative work in virtual enterprises. In W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske, editors, *Business Process Management (BPM 2003)*, volume 2678 of *LNCS*, pages 104–119. Springer, 2003.

[18] A. Schürr, A. J. Winter, and A. Zündorf˙ Graph grammar engineering with PROGRES. In W. Schäfer and P. Botella, editors, *Proceedings of the 5th European Software Engineering Conference (ESEC '95)*, LNCS 989, pages 219–234, Sitges, Spain, 1995. Springer.

[19] E. Subrahmanian, S. Konda, Y. Reich, A. Westerberg, and the N-dim group. Designing the Process Design Process. *Computers & Chemical Engineering*, 21, Suppl.:S1–S9, 1997.

[20] S. Tata and I. Chebbi. A bottom-up approach to inter-enterprise business processes. In *13th IEEE International Workshops on Enabling Technologies (WETICE 2004), Infrastructure for Collaborative Enterprises, 14-16 June 2004, Modena, Italy*, pages 129–134. IEEE Computer Society, 2004.

[21] W. van der Aalst. Interorganizational workflows: An approach based on message sequence charts and petri nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.

[22] W. van der Aalst. Process-Oriented Architectures for Electronic Commerce and Inter-Organizational Workflow. *Information Systems*, 24:8:639–671, 1999.

[23] W. van der Aalst. Loosely Coupled Interorganizational Workflows: Modeling and Analyzing Workflows Crossing Organizational Boundaries. *Information and Management*, 37:2:67–75, 2000.

[24] W. van der Aalst and M. Weske. The P2P approach to interorganizational workflows. In K. Dittrich, A. Geppert, and M. Norrie, editors, *Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE 2001), Interlaken, Switzerland*, volume 2068 of *LNCS*, pages 140–156. Springer, 2001.

[25] D. Wodtke, J. Weissenfels, G. Weikum, A. Kotz-Dittrich, and P. Muth. The MENTOR Workbench for Enterprise-Wide Workflow Management. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data, Tucson, Arizona, USA*, pages 576–579. ACM Press, 1997.

[26] Workflow Management Coalition. Terminology & Glossary. Technical Report Document Number WFMC-TC-1011, Workflow Management Coalition, 1999.