Tool Support for Dynamic Development Processes

Thomas Heer[1], Markus Heller[2], Bernhard Westfechtel[3], and René Wörzberger[1]

[1] Department of Computer Science 3, RWTH Aachen University, D-52056 Aachen
{heer,woerzberger}@i3.informatik.rwth-aachen.de
[2] SAP Research, CEC Karlsruhe, Vincenz-Priessnitz-Straße 1, D-76131 Karlsruhe
markus.heller@sap.com
[3] Applied Computer Science I, University of Bayreuth, D-95440 Bayreuth
bernhard.westfechtel@uni-bayreuth.de

Abstract. Development processes in engineering disciplines are highly dynamic. Since development projects cannot be planned completely in advance, the process to be executed changes at run time. We present a process management system which seamlessly integrates planning and enactment. The system manages processes at the project management level, but goes beyond the functionality of project management systems inasmuch as it both monitors and controls development processes and supports enactment of tasks through a work environment. However, the process management system does not provide process automation as performed in workflow management systems. Therefore, we have developed tools for integrating process management and workflow management such that repetitive fragments of the overall development process may be enacted in workflow management systems and monitored in the process management system. Even in the case of repetitive process fragments, the need for deviations from the workflow definition may occur while a workflow is being enacted. Thus, we have also realized a tool which allows to perform dynamic changes of workflows during enactment. Altogether, dynamic development processes are supported through a synergistic combination of process and workflow management systems, integrating process planning and enactment.

Keywords: Project Management, Process Management, Dynamic Changes, Workflow Management.

1 Introduction

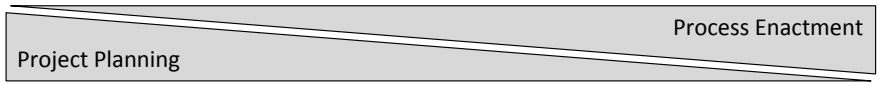
Development processes in engineering disciplines such as mechanical, chemical, and software engineering are highly dynamic. Typically, a development project may not be defined and planned completely in advance. Rather, at project runtime the process may have to be changed for a number of different reasons: The steps to be executed may depend on the structure of the product to be developed, problems may be detected which require feedback to earlier steps of the process, the knowledge of the process to be performed is incomplete or imprecise, etc. Thus, *dynamic changes* of the process have to be performed at project runtime, particularly in the case of long-term projects lasting for months or years.

This paper reports on *tool support* for dynamic development processes. It is based on long-term work which has been performed in the group of Manfred Nagl at RWTH

G. Engels et al. (Eds.): Nagl Festschrift, LNCS 5765, pp. 621–654, 2010.

Table 1. Comparison of solutions for development process management

	(1) Project Management Systems	(2) Integrated Project and Workflow Management Systems	(3) Process Management Systems	(4) Integrated Process and Workflow Management Systems	(5) Workflow Management Systems
Tools	• MS Project, Primavera, RPLan, et al.	• MILOS, IPPM, et al.	• AHEAD	• AHEAD + Shark	• Shark, Staffware, InConcert, WPS, et al.
Data	• Project plan	• Project plan • Workflow instances	• Dynamic task net	• Dynamic task net • Workflow instances	• Workflow instances
Characteristics	• Only planning • No enactment • No modeling of products	• Planning on project level • Enactment of individual subprocesses	• Dynamic task net represents plan and enactment state	• Workflow instances represent partially automated subprocesses	• Enactment of subprocesses • Modeling of overall development process infeasible
Consequences	• Monitoring and control difficult	• Mapping of status of workflows to project plan difficult	• Planning and controlling well supported • No automation of processes	• Automation of subprocesses, but • Less flexibility in workflow-managed subprocesses	• Subprocesses unconnected • No planning and scheduling support



Aachen University. Among these projects, the Collaborative Research Center *IMPROVE* [1], which was dedicated to models and tools for development processes in chemical engineering, played a dominant role. In addition, other engineering disciplines such as mechanical engineering and software engineering were studied, as well. In all of these research efforts, the dynamics of development processes have been a recurring theme which was addressed by a variety of approaches.

In the current paper, we present an overview of these approaches, which complement each other in order to provide comprehensive tool support for dynamic development processes. We structure the presentation with the help of Table 1. In the sequel, we first discuss related work and then describe the contributions of our own work.

1.1 Related Work

Project Management Systems. A development process is always enacted in the form of a development project which has a restricted set of human resources and given external deadlines. It is common practice to use *project management systems* for project planning and scheduling (column (1) of Table 1). Prominent examples for proprietary project management tools are Microsoft Project, Primavera and RPLan. Project planning and scheduling is essential for medium to large size projects and hence for all kinds of development projects. The project plan defines the tasks to be executed and their resource requirements.

Conventional project management systems do not allow to associate the tasks in a project plan with according parts of the product model. Furthermore, they support only the planning phase of the project but not the execution of the defined tasks. No client applications exist for the actual performers of the tasks. The enactment state of the

development process is not reflected in a project management system. Consequently, monitoring and control of a development project by using project management systems only is difficult, costly, and error-prone.

Workflow Management Systems. Workflow management has recently gained in importance in the different engineering domains. *Workflow management systems* [2] (see column (5) of Table 1) are used to support well-defined personal and collaborative processes. The workflow approach allows for a partial automation of processes, and the available technologies enable interoperability with other systems and applications in service oriented architectures. The usefulness of workflow support for development processes has been identified in both academia [3,4,5,6] and industry. Workflow support has been integrated into life cycle asset information systems for plant design [7,8] and integrated software development environments.

The home ground of workflow management systems, however, is in other domains such as business process management in administrations, banks, or insurance companies. In these domains, they constitute the industrial state of practice in process support. Therefore, it is not surprising that vendors and users of workflow management systems are looking for solutions which make workflow management systems applicable to dynamic development processes.

However, workflow management systems are not suitable for the management of whole development processes. Rather, they support only the enactment of structured subprocesses which may be defined in advance. Unlike project management systems, global project planning goes beyond the scope of workflow management systems. The tasks to be executed evolve during the course of the project. Many tasks cannot be planned until certain intermediate results of the development process are available, e.g., the flowsheet of a chemical plant or the design of a software architecture. It is not feasible to model a complete development process as one workflow due to the inherent uncertainties and dynamics. As a consequence, the workflow instances in the workflow management system are disconnected and not embedded into the context of the overall process.

Integrated Project and Workflow Management Systems. For these reasons, several research groups have investigated the opportunities of *integrating project management systems* with *workflow management systems* [3,4,5,6] (see column (2) of Table 1). In all of these approaches, project plans are used for global planning, and structured subprocesses are enacted with the help of workflow management systems. Furthermore, integration components couple these systems such that workflows may be represented in project plans.

However, the different integration approaches all face the problem that no information about the process enactment state is maintained in the respective project management system. Thus, the ability to monitor projects is severely limited. Furthermore, since products and data flows are not represented in project plans, product management is still beyond the scope of the integrated systems, and data flows between workflow instances have to be managed manually. Altogether, these restrictions call for an extended project management system (see next subsection).

1.2 Contributions

This paper makes several contributions regarding tool support for process management. The contributions can be divided into three parts which refer to the last three columns in Table 1.

Process Management System. The *A*daptable and *H*uman-Centered *E*nvironment for the M*A*nagement of *D*evelopment Processes (AHEAD) [9,10,11,12,13,14,15,16]) was developed within the long-term research project IMPROVE [1] and was applied to multiple domains (chemical, mechanical, and software engineering). AHEAD is a management system for development processes which provides integrated support for managing products, activities, and resources. A management system which offers this kind of integrated functionality is called a *process management system* throughout this paper (column (3) of Table 1).

AHEAD may be considered as an extended project management system. AHEAD differs from conventional project management systems inasmuch it manages the products of development processes in addition to activities and resources. Furthermore, it maintains state information of tasks and offers a work environment which developers use for performing the tasks assigned to them. In these ways, AHEAD integrates process planning and enactment, which offers great opportunities for the monitoring and control of development processes since the current execution state can be directly compared to the plan.

In contrast to AHEAD, workflow management systems do not support project planning. Their main contribution consists in the (partial) automation of routine processes. Furthermore, AHEAD differs from workflow management systems inasmuch it has been designed from the very beginning to allow for seamless interleaving of process planning and enactment. In contrast, most workflow management systems require a pre-defined workflow and support dynamic changes during enactment only to a limited extent.

Integrated Process and Workflow Management Systems. In the process management system AHEAD, activities are managed with the help of *dynamic task nets* [17]. The degree of automation of tasks is very limited on this level of process management. Work packages are assigned to developers who are responsible for delivering the specified results. For developers, a work environment is provided which maintains the documents required for each task and offers commands for activating tools operating on these documents. However, process automation as supported by workflow management systems is not provided.

Thus, process and workflow management systems offer complementary functionality which may be combined with the help of integration tools. As a result, we obtain an *integrated process and workflow management system* (column (4) of Table 1). This integration provides added value in particular when a company has already applied workflow management systems in a fragmentary way in its development projects. Then, the integrated system allows to reuse pre-defined workflows for process fragments, which are combined into a coherent development process.

Compared to the integration of project and workflow management systems, our solutions [15,18,19,20] provide for a tighter integration: Workflows are not only represented

in the task net managed by the process management system. In addition, the enactment state in the workflows may be monitored by transforming workflow operations onto state transitions in the task net. Furthermore, data flows may be propagated from the process management system to the workflow management system in order to provide workflows with inputs. Conversely, outputs created in the workflows are represented in the process management system such that they may be routed to successor tasks. Altogether, a much tighter integration is achieved than in systems integrating project and workflow management.

Adding Dynamics to Workflow Management Systems. State-of-the-art workflow management systems permit no or only limited deviations from the workflow definition at runtime of a workflow. However, even in the domains where workflow management systems are the state of practice for process support, such as business process management in administrations, banks, or insurance companies, the need for dynamic changes of workflows at runtime has been identified. For example, standard examples for routine processes such as travel reimbursement claims encompass so many alternative paths of execution that it is difficult to define a workflow which covers all of them. This argument is reinforced in the domain of development processes, which are less repetitive by their very nature. Therefore, using static workflows in dynamic processes alone does not provide a comprehensive solution to the problem of dynamic changes.

Many research projects focus on the development of *flexible workflow management systems* [21,22,23,24]. All prototypes have in common that they have been developed from the start to support dynamic workflows, i.e., flexibility is built *a priori* into the respective workflow engine. However, if a company has invested in a workflow management system which is widely used in its development projects, it may want to retain its investments and keep the system in use. This requires *a posteriori integration*: Tool support has to be developed in order to add dynamics to a legacy workflow management system.

Driven by this motivation, we have *added dynamics* to an existing *workflow management system* [25,26]. In this way, we have improved the capabilities of the workflow management system such that it may be applied to dynamic processes (contributing an improvement to column (5) of Table 1). In contrast to the flexible workflow management systems mentioned above, we had to solve the challenging problem of adding dynamics without modifying the given workflow management system. This problem was solved by a tool for dynamic changes which allows to add and delete workflow activities while the workflow is being enacted.

1.3 Structure of the Paper

The rest of this paper elaborates on the different and complementary support tools for dynamic development processes which have been described in Subsection 1.2. Section 2 presents the AHEAD process management system (column (3) of Table 1). Our solutions for using workflows in dynamic processes are described in Section 3 (column (4) of Table 1). Section 4 explains how we have added dynamics to a commercial workflow

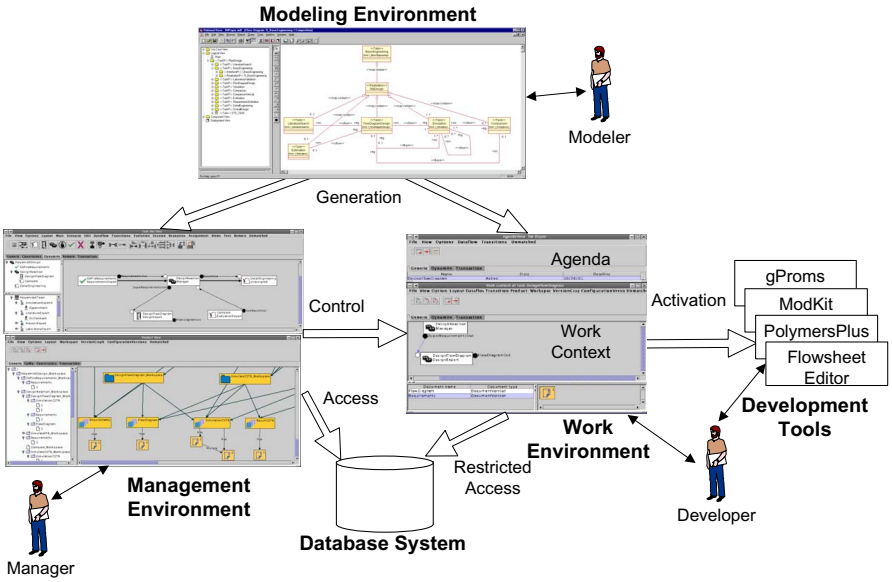


Fig. 1. Overview of the AHEAD system

management system (column (5) of Table 1). Related work is compared at the ends of all of these sections. The technical sections are followed by a discussion (Section 5). Finally, Section 6 concludes the paper.

2 The Process Management System AHEAD

The process management system AHEAD supports the integrated management of products, activities and resources for dynamic development processes. Subsection 2.1 provides an overview of the AHEAD system. Subsection 2.2 describes dynamic task nets for activity management, focusing on dynamic changes at run time. Subsection 2.3 explains how AHEAD may be adapted to a specific domain by defining a process model. Subsection 2.4 deals with deviations from and evolution of process models. A discussion of related work (Subsection 2.5) concludes this section.

2.1 System Overview

Figure 1 gives an overview of the AHEAD system. AHEAD offers environments for different kinds of users, which are called *modeler*, *manager*, and *developer*, respectively.

The *management environment* supports project managers in planning, analyzing, monitoring, and controlling development processes. It provides graphical tools which address the management of activities, products, and resources, respectively:

- For *activity management*, AHEAD offers dynamic task nets which allow for seamless interleaving of planning, analyzing, monitoring, and controlling. Dynamic task nets may be considered as extended project plans which in particular provide state information and include inputs and outputs of tasks as well as data flows in addition to control flows.
- *Product management* is concerned with the products of development processes, their versions and relationships (i.e., with the management of documents and models as supported by product/engineering data management systems, document management systems, or software configuration management systems).
- *Resource management* deals with the management of human resources (i.e., the members of the project team) which are assigned to development tasks, taking their roles and capabilities into account.

In the rest of this paper, we will focus on the management of activities, and we will not elaborate further on the management of products and resources.

AHEAD does not only support managers. In addition, it offers a *work environment* which consists of two major components:

- The *agenda tool* displays the tasks assigned to a developer in a table containing information about state, deadline, expected duration, etc. The developer may perform operations such as starting, suspending, finishing, or aborting a task.
- The *work context tool* manages the documents and tools required for executing a certain task. The developer is supplied with a workspace of versioned documents. He may work on a document by starting a tool such as e.g. a flowsheet editor, a simulation tool, etc.

The management environment and the work environment are both used at project run time. Both environments are highly *interactive*: The manager is provided with interactive tools for managing the development process, and the developer utilizes a work environment which provides commands for state transitions and tool invocations.

AHEAD consists of a generic kernel which is domain-independent. This means that the management environment and the work environment may be applied "out of the box". Optionally, models of development processes may be defined with the help of the *modeling environment*. For example, in order to apply AHEAD to design processes in chemical engineering, the modeler may define task types for flowsheet design, steady-state and dynamic simulation, etc. From a process model, code is generated for adapting the management and the work environment. The adapted system still provides seamless interleaving of planning and enactment, but offers pre-defined types for instantiating tasks, control flows, etc.

2.2 Dynamic Task Nets

A *dynamic task net* consists of tasks that are connected by hierarchical and non-hierarchical relationships:

- Tasks may be decomposed into subtasks, resulting in *task hierarchies*. Complex and atomic tasks are assigned to managers and developers, respectively.

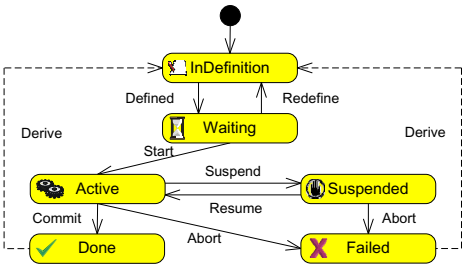


Fig. 2. State diagram

- *Control flows* resemble precedence relationships in Gantt diagrams. A sequential control flow corresponds to an end-start dependency. A simultaneous control flow enforces both start-start and end-end dependencies. Finally, a standard control flow represents an end-end dependency.
- *Feedback flows* are oriented oppositely to control flows. They are used to represent feedback in the development process, which is not possible in a conventional project plan.
- Each task has *inputs* and *outputs* which may be considered as ports for consuming and producing the products of development processes. *Data flows* describe the routing of documents along horizontal and vertical task relationships.

Seamless interleaving of planning and enactment constitutes the core mechanism for supporting dynamic changes in the AHEAD system at project run time. The manager may modify the task net at any time in the course of the project. Only minimal constraints are imposed on these modifications based on the state of execution. For example, it is not allowed to delete an active task because otherwise work performed by the developer would be lost. Since tasks are performed by humans (with the help of development tools), changes may be accommodated more easily than in the case of automated processes. For example, a new input may be attached to a task which has already started; it is up to the assigned developer how to process this input. In the case of an automated process, a change of this type usually has to be prohibited.

Thus, at project run time both edit operations and enactment operations may be performed on a task net. *Edit operations* change the structure of the task net. *Enactment operations* change the states of tasks (see below) or produce or consume data. All of these operations have to take the current enactment state into account.

The enactment states of tasks and their allowed changes are defined by a *state diagram* (Figure 2). In the initial state InDefinition, only edit operations are permitted on the current task (creation of input and output ports, creation of a refining task net in the case of a complex task). In Waiting, the task waits for its activation. In the state Active, both edit and enactment operations are allowed. When a task is Suspended, it may no longer produce or consume data, and it may have no active subtasks; editing operations are still allowed. Successful and failing terminations are represented by the states Done and Failed, respectively. In these states, no changes are allowed at all to ensure traceability of the development process.

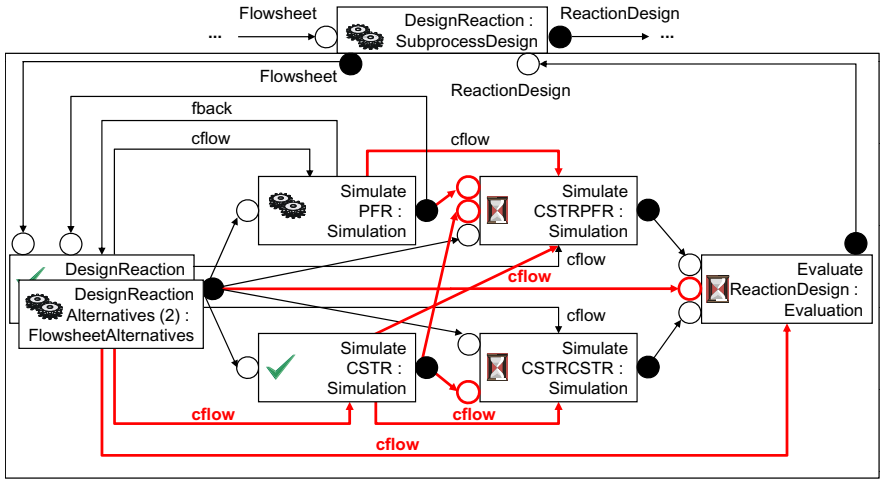


Fig. 3. Dynamic task net

Traceability is further supported by different ways of *version control* for elements of dynamic task nets:

- If a terminated task has to be reactivated, a new task version is created via the Derive operation[1]. The data of the old version remain unaffected and are copied to the new version.
- While a task is active, it may produce and receive multiple versions of outputs and inputs, respectively. Therefore, the tokens referencing these product versions are versioned, as well.

Figure 3 shows a snapshot of a (cutout of) a task net for a design process from the chemical engineering domain[2]. In the diagram, each task is represented by a rectangle containing its name, its type, and its state (displayed as an icon). Control, feedback and data flows are shown as arrows, composition relationships are represented by nesting component tasks into a box for the composite task.

In the sample process, which deals with the design of the reaction part of a chemical process, the designer has inserted four alternatives into the flowsheet: using single reactors of different types (CSTR and PFR, respectively) and using reactor cascades (CSTR-PFR and CSTR-CSTR, respectively). The designer already knows that eventually one of the cascades will be selected as the best alternative. However, simulating these cascades in a single step is too complex. Therefore, the individual reactors are simulated first, and the simulation results are propagated to the tasks for simulating the cascades.

[1] Strictly speaking, Derive is not a transition since it involves the creation of a new object (a new task version) rather than the state change of an existing object.

[2] The elements displayed with red thick lines denote inconsistencies with respect to the process model definition; see next subsection.

The task net of Figure 3 illustrates a state of enactment which is reached at some time during the course of the respective development project. In particular, the task net depends on the reaction alternatives, which are elaborated only in the initial design task. Even when these alternatives are known, it is not possible to generate the task net from the product structure. As explained above, the reaction alternatives are investigated in a certain order which has to be determined by the manager. Furthermore, unanticipated feedback may occur at any time, requiring changes to the task net for the purpose of feedback processing.

In the current state of enactment as shown in Figure 3, the tasks for simulating the reactor cascades as well as the final evaluation task still reside in state Waiting. The task for simulating the CSTR was already completed successfully. In contrast, the designer who is responsible for simulating the PFR detected a problem which caused the creation of a feedback flow to the initial design task. Since this task had already been committed, a new task version (rectangle in front) had to be created which now resides in state Active.

2.3 Process Model Definitions

The AHEAD system may be applied "out of the box" using so-called standard types for tasks, control flows, data flows, etc. Optionally, a domain-specific *process model definition* may be created which introduces domain-specific types and constraints. Processes are defined with the help of UML diagrams which have been tailored to process modeling through a UML profile defining adapted versions of class, communication, and state diagrams using stereotypes and tagged values [27,28,14].

With the help of *class diagrams*, process models may be defined at the type level. Figure 4 shows a class diagram taken from the reference scenario developed and used throughout the IMPROVE project [29]. This scenario deals with the basic engineering of chemical plants. For the task class interface SubprocessDesign, a realization class is defined which involves simulations of the respective chemical subprocess (as an alternative to laboratory experiments[3]) . For the simulation-based realization, a refining task net is defined on the type level. This task net consists of exactly one task for defining alternative chemical processes (class FlowsheetAlternatives), at least one Simulation task, and exactly one Evaluation task which selects the best alternative and delivers the overall subprocess design to the parent task. Tasks of these classes are ordered by control flow associations. Furthermore, a feedback flow association is defined from Simulation back to FlowsheetAlternatives. As in instance-level task nets, outputs and inputs are represented by black and white circles, respectively. Finally, data flow associations are used to connect outputs to inputs along vertical or horizontal task associations.

Class diagrams primarily serve to define *structural models*. Structural elements may be augmented with behavioral properties using tagged values (as e.g. EnactmentOrder = simultaneous in Figure 4). Furthermore, state diagrams and communication diagrams are offered for *behavioral modeling* (which are not described further here).

[3] In the process model definition, multiple realizations may be defined for a single interface. However, on the instance level the (single) realization of a task has to be selected when the task is instantiated. Furthermore, interface and realization are merged into a single object.

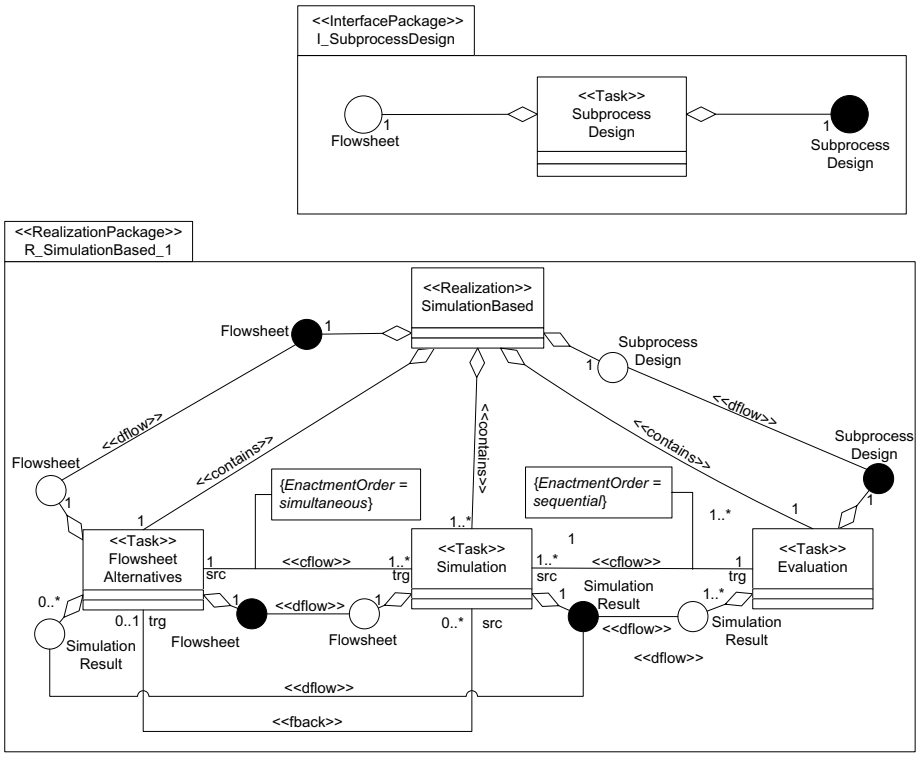


Fig. 4. Class diagram as process definition at the type level

Altogether, AHEAD provides a *wide spectrum approach* to process modeling. The spectrum covers ad hoc processes, which are based on standard types rather than on a domain-specific process model definition, process models defined on the type level with the help of class diagrams, and instance-level process patterns defined by communication diagrams.

2.4 Process Model Evolution

Despite of the inherent flexibility of our object-oriented approach to process modeling, it may happen that a process has to deviate from its definition. For example, the process model definition may not include the class of some task which has to be performed in the project at hand, or a control flow which is defined as sequential prohibits the overlapping enactment of the source and the target task. Therefore, AHEAD allows for *controlled deviations* of a task net instance from its definition: The project manager is supplied with commands for allowing/disallowing deviations at the level of subnets of the overall hierarchy.

Deviations result in *inconsistencies* which are signaled to the manager [14]. In Figure 3, inconsistent elements are emphasized with bold face fonts and thick lines in red

color. For example, the control flows between simulation tasks are marked as *structurally inconsistent* since they were not anticipated by the process modeler. Furthermore, the control flow ending at the task SimulateCSTR is *behaviorally inconsistent*: The task has already been terminated while its (reactivated) predecessor is still active.

Deviations may trigger process model evolution. In our example, the structural inconsistencies may be removed by adding further elements to the process model which were missing so far. This can be achieved by creating a new *process model version* and *migrating* the process model instance to the new version. In this way, AHEAD supports *round-trip process model evolution*.

2.5 Related Work

By using class diagrams, we follow an *object-oriented approach* to process modeling. This approach differs from the *procedural approaches* realized in workflow management systems [2,30,31,32]. The object-oriented approach provides for more flexibility since a task net may be composed specifically for the project at hand at run time by dynamic instantiation of task classes and associations. In contrast, instantiation of a workflow means that a pre-defined workflow is started.

AHEAD may rather be considered as an *extended project management system* than as an extended workflow management system. In particular, the *control flows* available in AHEAD strongly resemble precedence relationships in project plans. Unlike workflow management systems, AHEAD does not support control structures involving conditions. It was a deliberate decision not to support these control structures because we considered it too demanding for the project manager to develop a plan with multiple variants.

A variety of mechanisms has been designed and implemented for supporting dynamic processes (not only in workflow management systems, but also in *process-centered software engineering environments* [33,34]). In the following, we will list some of these mechanisms (without striving for completeness):

Ad Hoc Processes. Processes are defined on the fly for each specific case, i.e., there is no predefined process definition [21].

Exception Handling. Processes defined for the "standard case" of processing are extended with exception handlers, which describe how to react on deviations from the standard case [35].

Flexible Control Flows. Many process definitions suffer from imposing too rigid constraints on the order in which process steps are executed. Thus, definition languages have been developed for describing more flexible control flows [22,36].

Late Binding. Process definitions are usually decomposed into subprocesses to manage complexity. In the case of late binding [37], the definition of a subprocess may be deferred until it is going to be executed.

Interleaving of Definition and Enactment. A more general approach to dynamic changes supports seamless interleaving of definition and enactment in a similar way as in integrated programming environments, where a program may be modified during execution [38,39,23].

Toleration of Inconsistencies. Flexibility is increased by tolerating inconsistencies of instances with respect to their definitions [40]. For example, in [41] a process step may be executed even if its preconditions are not satisfied.

Version Control for Process Definitions and Migration. In these systems, process definitions are put under version control [24,42,43]. Furthermore, running instances may be migrated to revised definitions.

AHEAD supports *dynamic changes* through ad hoc processes ("untyped" task nets), interleaving of planning and enactment, and toleration of inconsistencies. Furthermore, *process evolution* is supported through version control of process definitions and migration of process instances. AHEAD does not address exception handling and flexible control flows, which are partially obsolete since process instances may be changed flexibly at run time. Moreover, AHEAD does not support late binding.

3 Using Workflows in Dynamic Processes

In this section, the integration of a workflow management system with the management environment for development processes in AHEAD is described. This integration is beneficial when existing workflow management systems within an organization and all existing workflows can be utilized to represent relevant workflow details within the overall development process. Not all details modeled in workflows have a real value for the process manager on the abstract process planning level that addresses the coordination and distribution of work in the managed overall development process. Instead, only relevant aspects of workflow processes need to be monitored by the manager. For this purpose, we have developed a generic integration approach which is applicable to the majority of integration scenarios and deals with the desired partial monitoring of workflows in development processes. With respect to the details of the integration, several alternatives can be distinguished which are more or less appropriate in certain scenarios.

Subsection 3.1 briefly explains the motivation for our work. In Subsection 3.2 the coupling of a workflow management system with the AHEAD system is described so that workflows can be projected into dynamic task nets and monitored by the process manager in the management environment of AHEAD. Subsection 3.3 describes the weaving of workflow processes with other (dynamic) process parts within the overall dynamic process and Subsection 3.4 explains how workflow modeling capabilities can be additionally used. Finally, Subsection 3.5 gives an account of related work.

3.1 Motivation

We have developed an approach to integrating workflow processes into the overall development process and have realized a coupling of workflow management systems with AHEAD for use within an organization. In this setting, the AHEAD system is used within the organization as the central instance for the project planning of the development process and the coordination of all (human) work that is carried out during the process. The overall dynamic development process is represented as a dynamic task

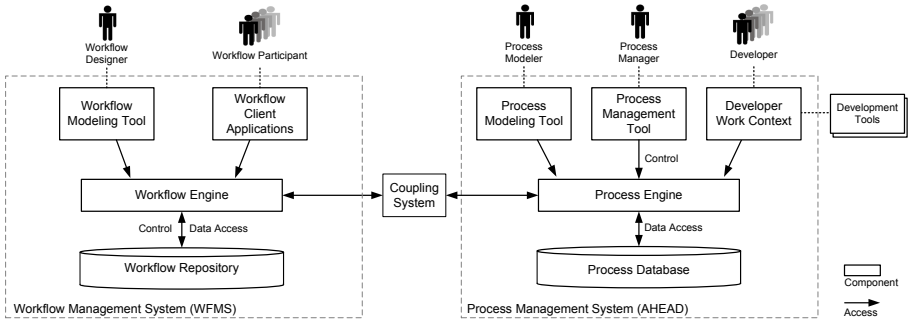


Fig. 5. Coupling of a workflow management system with the process management system AHEAD

net and can be *monitored* and controlled by the process manager within the management environment of AHEAD. Repetitive routine processes are executed in workflow management systems. The workflow processes are integrated as *process views* [15,19] into the overall dynamic processes and projected onto dynamic task nets. The process manager can monitor all parts of the development process within the management environment in the same process notation of dynamic task nets. In this way, an overall development process comprising dynamic and repetitive partial process fragments is executed across heterogeneous process support systems.

The outlined integration approach is based on the observation that although development processes are generally human-driven, some fragments of the development processes are indeed of a repetitive nature, i.e., repetitive process definitions can be defined for these fragments and modeled as workflow definitions. A similar approach to use workflow management systems within human-driven development processes is described e.g. in [44].

Within a prototypical realization, the AHEAD system has been integrated with the workflow management system SHARK from ENHYDRA [45]. A full account of the coupling approach and the realized prototype is given in [15,18,19]. The resulting solution approach was adopted in the technology transfer project T6 [19,20] for the integration of the *Proc*ess Management *E*nvironment for *E*ngineering *D*esign Processes (PROCEED) with a workflow management system. PROCEED is a new implementation of AHEAD based on an industrial platform.

3.2 Monitoring Workflows in Dynamic Processes

Figure 5 shows the resulting architecture for a coupling of a workflow management system (left) with AHEAD (right) via an event exchange infrastructure. The AHEAD system triggers the instantiation of a workflow. Upon the triggering event, the workflow management system spawns a new workflow instance and creates new run time instances for the first startable workflow activities of that workflow definition. All necessary input data is transferred from AHEAD to the workflow management system at this moment for further consumption within the new workflow instance. The created

workflow-activity instance is immediately started in the workflow management system to begin the workflow execution. In particular, for each human workflow activity, a developer can start working on this activity. During task execution, all relevant workflow changes (activity or task state changes, resource assignment changes, data changes) are signaled as events to the AHEAD system where the corresponding task net that represents the workflow process is updated accordingly. Vice versa, all necessary changes in the workflow (and state changes) are signaled from the AHEAD system to the workflow management system as events where they are processed in a similar way.

Mapping of Workflow Processes into Dynamic Task Nets. Workflow processes can be represented within the overall dynamic task net via a *projection* of the workflow process onto a dynamic task net fragment (called *workflow task net fragment*). Each task in a workflow task net fragment that represents a workflow activity is called a *workflow task*.

The transformation of workflow processes to dynamic task nets does not need to preserve the full semantics of both formalisms, because this would lead to very rigid requirements for the systems to be integrated. Since the workflow fragments are used in AHEAD for monitoring purposes only, it is tolerable if some process information is lost during the generation of the workflow fragments. Thus, only selected details of the workflow process, which are necessary to represent the coordination aspects of the activities in the workflow, are mapped into a dynamic task net. The projection is realized with a transformation algorithm to convert a workflow process definition into a corresponding dynamic task net fragment definition. While a brief summary is given in the sequel, more details of the chosen transformation projection and the coupling infrastructure are described in [46].

Within AHEAD, a workflow task net fragment can then be inserted into the task net at a location that is selected by the process manager (i.e. as children of the parent task in the current process fragment). Initially, it is isolated at this moment from the rest of the dynamic task net. Hence, the process manager has to connect tasks from the new imported workflow task net fragment with tasks from the remaining dynamic task net via control flows and data flows (a *weaving* step).

In order to reduce the effort for the integration of multiple different workflow management systems, we make use of a neutral exchange format to represent a wide-spread workflow modeling language. The language XPDL has been chosen since it is a standard notation brought forward by the Workflow Management Coalition [31]. Therefore, multiple workflow management system products that adopt this standard can be supported with the coupling solution in general.

The projection defines both the structural mapping of workflow structures onto task net structures as well as the mapping of the state diagram of workflow activities (in XPDL) onto the state diagram of AHEAD tasks. Additionally, the data flow between workflow activities is projected onto the data flow between AHEAD tasks. Workflow definitions can include subprocess calls and iterations. These mapping details are explained in the sequel.

To illustrate our projection approach, Figure 6 introduces a simple example of a workflow from the chemical engineering domain and shows its projection into a dynamic task net fragment. The underlying scenario in the chemical engineering domain

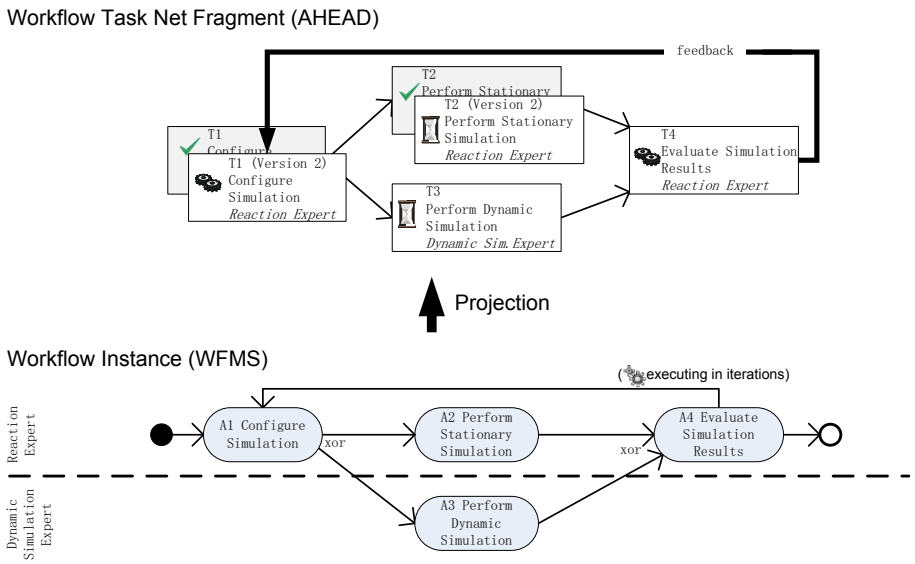


Fig. 6. Workflow instance and according task net fragment

has already been introduced in Section 2. The design of a certain part of the chemical plant during the development process, such as the design of the reaction part, can be supported with workflow management system technology. Specific routine processes can be executed within workflow management systems but need to be monitored within dynamic task nets as the contained process steps are important on the coordination and management level for the process manager.

The example represents a routine subprocess for performing simulation calculations for a given reactor part in the flowsheet. This process refines the simulation task defined in Figure 4 and instantiated multiple times in the task net of Figure 3. Simulations are carried out iteratively until a satisfactory result is achieved. Two different simulation methods can be used here to perform the simulation: A stationary or steady-state simulation allows to simulate the reactor with respect to the equilibrium state while a more complex (and more expensive) dynamic simulation investigates the behavior of the chemical process in non-steady states (e.g., start of the reaction or technical faults). For this scenario, the lower part of the figure shows the workflow process (workflow management system) and the upper part shows the projected dynamic task net (AHEAD). The workflow is comprised of four activities: In a first activity A1 Configure Simulation, a reaction expert estimates the requirements for the desired simulation and decides which type of simulation is carried out. According to this decision, in each iteration either activity A2 Perform Stationary Simulation is executed by the reaction expert or A3 Perform Dynamic Simulation is delegated to a simulation expert with specific expert knowledge (only one of both activities is executed). In activity A4 Evaluate Simulation Results, the reaction expert interprets the simulation results and decides if the simulation data are sufficient or if the simulation has to be repeated with modified

Table 2. Mapping of workflows to dynamic task nets [15]

Workflow Model Concept(XPDL)		*Process Model Concept(AHEAD)*
Workflow Process	↔	Task Net
Workflow Activity	↔	Task
Workflow Participant	↔	Performer
Data Field	→	Document
Formal Parameter	→	Document
Transition Information	↔	Control Flow (seq.)

configuration parameters or a different simulation method (iteration from activity A4 to A1). This example is used throughout the remainder of this subsection.

Structural Mapping. The mapping of the most important elements of the workflow meta model in XPDL to elements of the AHEAD meta model is described by a set of mapping rules as follows (summarized in Table 2). A workflow process definition from XPDL is mapped to a task net in AHEAD. All activities in a workflow process definition are mapped to AHEAD tasks. In the example of Figure 6, activities A1-A4 are mapped to tasks T1-T4 in the dynamic task net. All tasks are inserted at the same time into the development process. However, for an activity there can be multiple task versions in the task net (due to iterations, see below). This is shown in the figure for activities A1 and A2 which have two different task versions in the task net. For subprocess definition activities and route activities in XPDL, a new task is contained in the task net. All workflow activities within an XPDL subprocess are mapped to AHEAD tasks likewise.

Workflow participants (XPDL) are assigned to tasks in the task net and assignments of participants to workflow activities are carried over to AHEAD by assigning participants to tasks. In Figure 6, the workflow participants Reaction Expert and Dynamic Simulation Expert are mapped to corresponding resources in the dynamic task net. For example, the assignment of activity A1 to participant Reaction Expert from the workflow process is also visible in the task net where task T1 is assigned to the corresponding resource (shown in the bottom line of the task's label).

For workflow-relevant data from XPDL, corresponding document elements are created in AHEAD (not shown in the figure). Updates to workflow-relevant data (in workflow variables) are mapped to the creation of new versions of the corresponding documents. Not all workflow-relevant data (variables) need to be mapped to AHEAD task nets. For pragmatic reasons, a naming scheme was defined to automatically separate mapped from unmapped workflow-relevant data.

Transitions, Iterations, and Feedback Flows. Transitions between workflow activities in XPDL are transformed into sequential control flows in AHEAD (as shown in Figure 6). Please note that XOR-transitions and AND-transitions are both mapped in the same way[4]. The corresponding state machine mapping is described below. A workflow activity inside a loop control structure may be associated with multiple task versions

[4] The mapping is not injective, i.e., the transformation implies a loss of information. As mentioned earlier, branches cannot be represented in AHEAD. AND- and XOR-transitions may be distinguished in AHEAD only by monitoring the enactment of the respective workflow.

Table 3. State mapping between workflow activities and tasks

Workflow Activity State(XPDL)		*Process Task State(AHEAD)*
—	↔	InDefinition
open.not_running.not_started	↔	Waiting
open.not_running.suspended	↔	Suspended
open.running	↔	Active
closed.completed	↔	Done
closed.terminated	→	Failed
closed.aborted	→	Failed

(in the task net) according to the iterations of the loop. For example, the activity A1 in the workflow is associated with two versions of task T1 in Figure 6. While the loop in the workflow process expresses that the activities A1–A3 have to be repeated until a success criterion matches, in the task net the concept of *feedback flow* is used to express the same meaning. In our example, initial versions of tasks T1 and T2 are executed. Subsequently, T4 detects a failure. To repair the failure, a second version of task T1 is instantiated and a feedback flow from T4 to the new version of T1 is created. Subsequently, the new version of T1 is reactivated. Since the steady-state simulation has to be performed once more, a new version of T2 is created and prepared for enactment.

Dynamic Semantics: State Machine Mapping. Table 3 shows the mapping of the state machine of tasks in a dynamic task net, workflow instances and workflow activities. The coupling is described here informally. When a workflow activity is in a pre-activation state (open.not_running.not_started) so is the corresponding AHEAD task (here: Waiting). The task state InDefinition in AHEAD is only used within AHEAD and has no direct mapping from a XPDL state. When a workflow activity enters the running state (open.running), the corresponding AHEAD task is activated (Active). The XPDL meta model defines three terminal states, namely successful termination (closed.completed), user-initiated cancellation (closed.terminated), and system-initiated cancellation (closed.aborted). The AHEAD meta model defines only two terminal states for successful termination (Done) and failed task cancellation (Failed). The workflow state closed.completed is mapped to task state Done, while the additional information provided in XPDL is lost during the mapping when both cancellation states closed.terminated and closed.aborted from XPDL are mapped to the cancellation state Failed in AHEAD.

3.3 Weaving of a Workflow Fragment into a Dynamic Task Net

The individual workflow processes which are enacted by means of the workflow management system are all part of one overall development process which is represented by a hierarchically structured task net in the process management system. Therefore, the workflow fragments representing the workflow instances have to be embedded into the task net.

The workflow activities are represented by workflow tasks in the workflow fragment. The workflow instance itself can also be represented by a task in the task net. In this

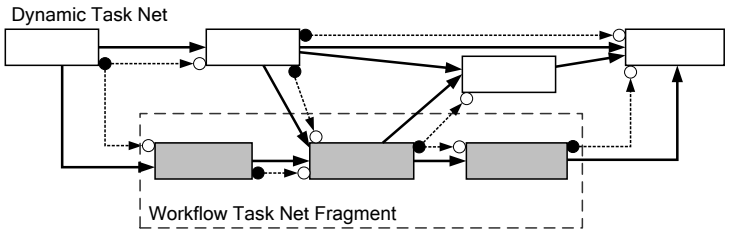


Fig. 7. Weaving of a workflow fragment into a task net

case, the subtasks of this task represent the workflow activities and no other tasks are contained in its realization.

However, the workflow instance does not necessarily have to be represented by a task in the task net. Alternatively, the tasks representing the workflow activities can be directly embedded into an existing task net. In this case, there may be sibling tasks which do not represent workflow activities.

When a subprocess of a development process is enacted in a workflow management system, the enactment of the workflow may depend on the state of tasks which are not part of the workflow. This is mostly the case because data or documents which are created outside of the workflow are required in certain activities of the workflow, e.g. when a workflow for the permit procedure of a chemical plant requires at a certain step the result of the cost calculation. On the other hand, the enactment of the workflow and its (intermediate) results may influence other tasks in the development process, e.g. when a workflow is enacted for the specification of a device, and this specification is required for the procurement of the device later on. Altogether, two different cases for the embedding of a workflow into the surrounding development process may be distinguished.

In the first case, the workflow as a whole requires input data from preceding tasks and produces results which are required in succeeding tasks. All required inputs have to be available at the start of the workflow and cannot be provided later at run time. Intermediate results of the workflow are not available in the surrounding process. This case requires that the workflow instance is represented by a task in the dynamic task net.

In the second case, the individual activities of the workflow may consume external data from outside the workflow and may provide intermediate results which can be used by external tasks. This is illustrated in Figure 7, where the gray boxes surrounded by the dashed line represent workflow tasks in the workflow task net fragment while the white boxes represent external tasks in the surrounding process. Required inputs for workflow tasks can be provided by external tasks as required, even after the start of the workflow, and intermediate results of the workflow are available for external tasks before the completion of the workflow. This case in which individual workflow tasks are connected with external tasks is called *fine grained weaving* of processes. In this case, it is not required that the workflow instance is represented by a task in the dynamic task net. Data inputs and outputs do not have to be defined for the workflow instance but can be directly defined for workflow activities.

The weaving of a workflow with the surrounding process imposes several technical requirements for the integration of the respective management systems. The handover of data between the two systems requires *well-defined data formats* which can be written and read by both systems. In case of documents which serve as input for tasks or workflow activities it may be required to transfer uniform resource locators (URLs) which specify the location of the respective documents in a common repository.

In the case of fine grained weaving, control flow connections between the tasks in a workflow fragment and the surrounding process context influence the execution of the workflow and the surrounding process. In the case of outgoing control flows, the workflow management system should provide an interface for the notification of external systems about state changes of activities and workflows. If such an interface is not available, a pull mechanism could be realized in which the process management system queries the workflow management system for changes in regular intervals. Vice versa, the execution of a workflow instance should respect the constraints imposed by incoming control flows. Different technical solutions are possible. The original workflow definition could be augmented by additional activities and control flows which represent the context of the workflow. Alternatively, special workflow activities and according workflow variables could be used in the original workflow definition to allow for later definitions of control flows from external tasks.

3.4 Workflow-Managed Dynamic Task Nets

The main motivation for the integration of AHEAD with a workflow management system is the continued utilization of existing workflow management solutions. In this case, the process participants who used the workflow management system before the integration, will go on using the client applications of the workflow management system after the integration. In particular they will use the worklist handler of the workflow management system to get informed about their assigned tasks.

The workflow fragment which is embedded into the dynamic task net of the overall development process serves the project manager mainly to view the status of the workflow tasks. The workflow tasks in the fragment may be atomic, i.e. they are not refined by any subtasks, or the tasks may themselves represent workflow instances and can therefore be refined by workflow fragments.

The second case accounts for the common scenario in workflow management system where one workflow invokes another workflow in a synchronous fashion, i.e., a workflow activity initiates the execution of another workflow and waits for its termination. This can be modeled in the dynamic task net by a task which represents at the same time the workflow activity and the invoked workflow instance. The subtasks of this task represent the activities of the invoked workflow.

Another motivation for the integration of an AHEAD-like process management system with a workflow management system may be to use the modeling capabilities for workflows, in particular alternative branching and loop structures, to enable a (partial) automation of the defined processes. This was the main motivation for the integration of PROCEED with a workflow management system in the technology transfer project T6 [19,20]. Workflows are not only monitored in PROCEED but the workflow fragments are also managed automatically by the workflow engine according to the enactment of

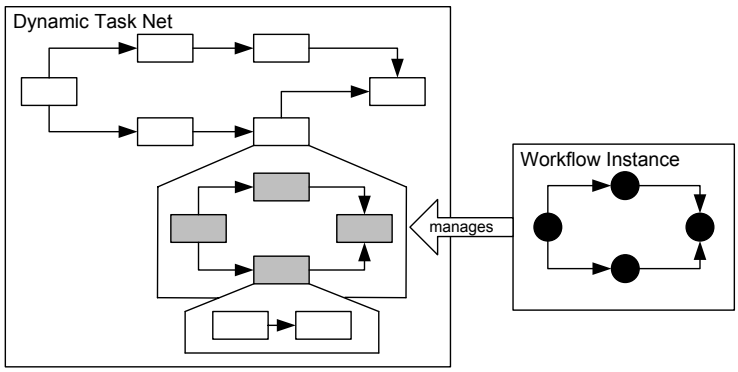


Fig. 8. Nesting of workflow-managed and manually managed dynamic task nets

the according workflow instances (tasks are prepared for execution, new task versions are created, etc.). In this case, the workflow task net fragments are called *workflow-managed dynamic task nets*. The advantage of this kind of integration is the usage of the workflow modeling capabilities and the workflow engine of the workflow management system. The process participants can manage their assigned tasks by means of the PROCEED work environment and do not need to use the worklist handler of the workflow management system.

The usage of the PROCEED work environment by the process participants has an influence on the required state machine mapping of PROCEED and the workflow management system. The start of a workflow activity has to be mapped to the transition of the according task in the workflow fragment from *InDefinition* to *Waiting* because tasks have to be started by the assigned resources and should not be activated by the workflow management system automatically.

To allow for the use of a workflow definition for a high-level subprocess of the overall development process, it is possible to refine workflow tasks in a workflow-managed task net by manually managed dynamic task nets. Otherwise, all subprocesses of the workflow process would have to be defined as workflows as well, which is infeasible for dynamic development processes. Figure 8 shows a hierarchically structured dynamic task net in which a subprocess is managed by the workflow engine but a workflow task is refined by a manually managed task net.

The management of task net fragments in PROCEED by a workflow engine imposes additional technical requirements for the integration of the two systems. A workflow activity whose corresponding task in the workflow fragment is realized by a manually managed task net has to wait for the successful termination of this task net in PROCEED before it can be terminated. This can be realized by the implementation of special workflow activities which wait for an external event before they terminate. Some workflow management systems provide this functionality.

3.5 Related Work

Several research groups have investigated the possibilities to integrate project management systems with workflow management systems.

Bauer distinguishes two approaches in [3]: loose coupling, where several workflow instances can be mapped to a single project task, and close coupling, where there is a one-to-one mapping of workflow activities and tasks in the project plan. The close coupling approach is not applicable, when the project plan and the workflows are on different abstraction levels. Hence Bauer presents a generic integration architecture for loose coupling which uses event-condition-action(ECA)-rules for data propagation and aggregation.

The MILOS tool [4] is an integrated solution for the management of software development processes. MS Project is used as a planning interface and is extended by means to define information flow between different tasks. The run time coupling between the workflow management component and MS Project is realized by means of ECA-rules.

In [5], the IPPM system is described which integrates features for both project and workflow management. An approach for the unfolding of workflow control structures to tasks in a project plan is presented.

In [6], Bussler discusses issues regarding the integration of workflow management systems and project management systems in general. Two approaches are distinguished for the mapping of control structures. Continuous mapping describes the case where tasks are inserted into the project plan at workflow run time according to the decisions made. With static mapping, all alternative paths are inserted before the start of the workflow.

The integration of AHEAD with a workflow management system is based on a well-defined mapping between the according process meta-models. Therefore, the definition of ECA-rules is not necessary, neither by the system developer nor by the user.

The different related integration approaches all face the problem that no information about the process enactment state and the data flow is available in the respective project management systems. As a consequence, fine grained weaving of a workflow instance with other tasks in the project plan is not possible. The context of a workflow instance in a project is merely defined by the complex task to which it belongs.

4 Adding Dynamics to Workflow Management Systems

4.1 Motivation

In the previous section, we were concerned with using workflows in dynamic development processes. By integrating process and workflow management systems, workflows are glued together in order to form a coherent development process. However, this approach does not release the restrictions of the used workflow management systems concerning support of dynamic changes.

In this section, we will present tools for *adding dynamics* to a *workflow management system* a posteriori. These tools were developed in a project in the business process domain. In contrast to the integration approach of the previous section, the extended workflow management system may be used as a stand-alone component. The primary goal was to make classical workflow applications more flexible. However, at the same

time this flexibility makes the workflow management system better suited for the application in development processes, which generally require more flexible support for dynamic processes than business processes. Therefore, the extended workflow management system may be integrated with a process management system along the lines of Section 3. In this way, the approach presented below complements the work described in the preceding sections.

In the AHEAD system, dynamic changes are supported by seamless interleaving of planning and enactment. In contrast, dynamic changes to repetitive processes are better handled by *deviation* from a workflow definition rather than by continuous evolution of some dynamic task net. Yet, classical workflow management systems do not allow for dynamic changes. In the following, we describe how and to which extent a classical workflow management system can be *upgraded a posteriori* in order to support even dynamic business processes. In particular, we extended the workflow management system *IBM WebSphere Process Server (WPS)*, which provides execution support for workflow definitions. In contrast to the concepts of Section 3, we just consider a single modeling language for executable models, namely the standardized language WS-BPEL [32] used in WPS.

In Subsection 4.2, we draw the big picture of the a posteriori extension of the workflow management system WPS before we elaborate on the kinds of dynamic changes which are actually supported by the approach (cf. Subsection 4.3). Finally, related work is discussed in Subsection 4.4.

4.2 Simulating Dynamics

Adding functionality for dynamic run time changes a posteriori to an existing workflow management system like WPS does not work in a straight forward fashion. WPS is a closed source system. Thus, adding dynamics by modifying the existing source was not an option. Instead, an additional architectural layer – called *dynamics layer* – was introduced (cf. Figure 9). This dynamics layer simulates dynamic changes inasmuch as workflow participants experience dynamic structural changes in their running workflow instances while the actual workflow definitions within WPS (forcedly) remain structurally unchanged.

The dynamics layer reflects the intrinsic distinction between build time and run time of classic workflow management systems as it consists of two parts. Existing workflow definitions are automatically augmented by a WS-BPEL transformer at build time by additional control flow structures. At run time, these additional control flow structures are used to deviate from the original control flow definition. Their run time behavior is controlled by the dynamics component which is the run time counterpart of the WS-BPEL-transformer. The additional complexities that come along with the dynamics layer are completely hidden from process participants. In the so called process model editor, which serves as a front end for process participants, dynamic changes are indeed displayed as structural changes in process instance models, i.e., models of running processes.

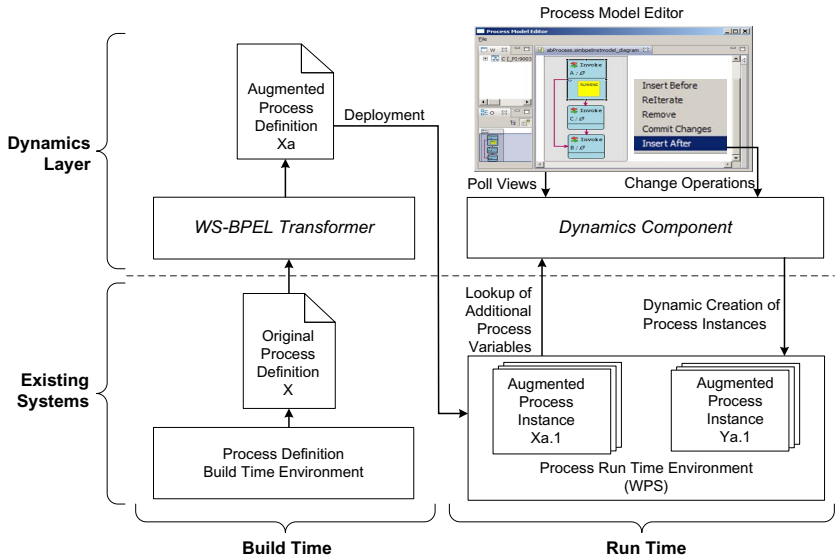


Fig. 9. Deviation of workflow enactment from workflow definitions

4.3 Dynamics Patterns

The realization of the dynamics layer is aligned with typical kinds of dynamic changes. These are called *dynamic patterns* which are described in detail in [25,26] and briefly in the following:

Dynamic Adding. Frequently, there is a need to dynamically add activities or even workflow fragments consisting of several activities to a running workflow instance. In a classical, i.e. static, workflow management system one could model placeholder activities — serving as *extension points* — among the actual activities. These added activities can then be used at run time for intermediate rerouting the control flow to another newly created process instance. However, manually inserting such activities is a tedious task and renders the workflow definition unmaintainable as every possible position in the workflow definition has to be supplied with an additional activity. Thus, the WS-BPEL transformer automates the augmentation and the process modeler can stick to his or her original workflow definition.

Figure 10 provides a small example from the software engineering domain (an implement-test-release process). The original workflow definition X is augmented with additional activities DAI1 to DAI4 yielding a workflow definition Xa. This workflow definition can be executed at run time within a workflow instance Xa.1. The process model editor hides the additional activities from the process participant. Assume that during the execution of Implement the need for an additional quality assurance step is identified. The process participant then just embeds an according activity Review within the existing control flow structure. This addition is reflected by the dynamics component inasmuch the DAI2 activity is bound to a newly created instance Ya.1. Each DAI

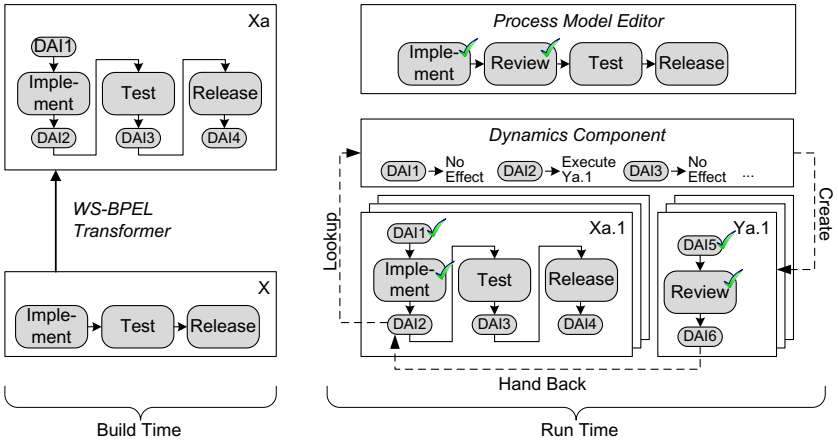


Fig. 10. Realization of Dynamic Adding

activity retrieves binding information from the dynamics component when executed. In the case of DAI2, the new instance Ya.1 is executed such that the Review activity is performed. After completion of Ya.1, the control flow returns to the caller DAI2, and Xa.1 is executed further.

Effectively, the DAI activities together with the binding information of the dynamics component, which is indirectly manipulated by the process participant, simulate an actual dynamic change, which is possible due to late binding.

Dynamic Removal. Complementary to dynamic addition of activities, process participants frequently need to remove mandatory activities. *Dynamic removal* constitutes another dynamics pattern which is supported by the dynamics layer. Yet the overall approach remains the same: At build time, the WS-BPEL transformer augments the workflow definition by additional control flow structures which are used at run time in order to simulate the dynamic removal of activities from a running workflow instance.

As Figure 11 depicts, each original activity is surrounded by a DRD decision activity (diamond). Depending on a dedicated value, which is stored in the dynamics component and evaluated when the respective DRD decision is reached by the control flow at run time, the original activity is either executed (default case) or bypassed. The latter takes place if the process participant has dynamically removed the activity. Again, the generated DRD activities are invisible in the process model editor. Here, dynamic removals are rendered by actually removing the respective activity from the workflow definition.

Dynamic Iteration. Besides the decision elements labeled with DRD, Figure 11 also depicts a decision labeled with DID. This element is generated by the WS-BPEL transformer in order to allow for *dynamic iterations*. These are the equivalents to the feedback flows in AHEAD, i.e. they allow to step back in the control flow and to re-execute activities, which might have been erroneously executed before. At run time, the DID decision works in combination with DRD decisions: Consider the case, where the

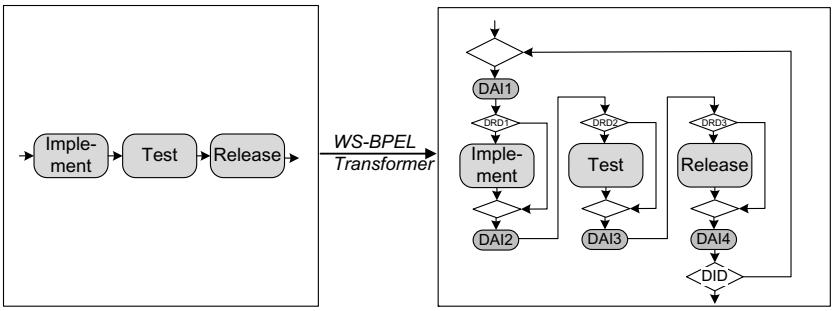


Fig. 11. Transformation for Dynamic Removal and Iteration

workflow instance already has proceeded to Release and the process participant wants to step back to Implement (e.g., because an automated regression test which was not part of the unit Test activity fails in the Release step). Then the value backing the DID in the dynamics component is reset such that the control flow returns to the very beginning of the workflow. In this way, another iteration is started until eventually the final Release step is passed successfully.

Expressive Power of Pattern-Oriented Changes. Interleaved planning and execution calls for maximal freedom concerning the editing of process models (as provided by dynamic task nets in the AHEAD system). In contrast, the dynamics patterns – each constituting a certain kind of deviation – are more restrictive. Regarding the example of Figure 10, it is, e.g., not possible to add the activity Review in parallel to Test (allowing to delegate review and test to different developers working in parallel). Yet, this lack of flexibility does not pose a real problem: The decline of process performance due to the serial instead of parallel execution is neglectable as the dynamic change just affects a single workflow instance.

4.4 Related Work

Many research groups have investigated the development of flexible workflow management systems. Weber et al. [47] provide an overview over the different aspects of flexibility in workflow management systems and define according change patterns. Within the ADEPT project, Reichert et al. [23] investigate dynamic changes to running workflow instances. For this purpose, they use a graph based calculus with some optimizations to ensure the correctness of control flow graphs. The WASA approach [48] is similar to the ADEPT approach. Dynamic changes in workflows are supported by special planning activities which define possible extension points for the workflow. Casati et al. [49] advocate the automated handling of unforeseen process exceptions using a so-called exception-specification language. This language provides additional declarative set-oriented conditions which complement common imperative workflow definitions. All prototypes have in common that they have been developed from the start to support dynamic workflows, i.e. flexibility is integrated a priori. Furthermore, the prototypes use their own modeling languages for workflow definitions instead of standardized languages.

5 Discussion

In this section, we summarize several lessons learned, i.e., conclusions which we have drawn from our work on process support for dynamic development processes. These conclusions are condensed into a set of theses, each of which is discussed briefly.

5.1 Domain-Specific vs. Domain-Independent Management System

Thesis 1 (Domain-Independent Management Systems). *The required functionality of process support tools depends on the actual application domain only to a limited extent. Other process properties such as the level of granularity and the degree of structuring are much more relevant.*

In this respect, we completely agree with [50], Thesis 3. Process support systems cannot be reasonably classified with respect to their application domains. For example, the AHEAD system was applied to multiple engineering disciplines such as mechanical, chemical, and software engineering. Rather, each process support system builds upon certain assumptions concerning the target processes to be addressed. A process support system may be applied to processes satisfying these assumptions, regardless of the specific domain. For example, a workflow management system which has been designed for supporting static business processes may be applied to software processes as long as these processes match the properties assumed by the workflow management system (e.g., structured change request processes, where a well-defined workflow exists which has to be enforced). Furthermore, a process management system like AHEAD may be applied not only to development processes, but also to other processes provided that there is a need for project planning and integrated management of products, activities and resources (e.g., construction of buildings).

Thesis 2 (Ubiquitous Need for Dynamic Changes). *The need for changing processes at run time arises in virtually any application domain. In particular, dynamic changes need to be supported even for structured routine processes.*

In our work, we have primarily studied development processes in different engineering disciplines. However, a small fraction of our work has been dedicated to business processes, as well. In all of these application domains, a strong need for dynamic changes has been identified. The specific requirements for supporting dynamic changes may vary from application to application. However, the mechanisms listed in Subsection 2.5 — ad hoc processes, exception handling, flexible control flows, late binding, interleaving of planning and enactment, or deviations from process model definitions — may be employed in any domain. In particular, we argue that workflow management systems need to support dynamic changes instead of merely enacting static workflows as defined.

5.2 Project vs. Workflow Management

Thesis 3 (Different Focus: Project Planning vs. Process Automation). *Conventional project and workflow management systems focus on different process support functions.*

While project management systems address project planning on a fairly coarse-grained level, workflow management systems automate routine processes typically at a more fine-grained level.

Here, we disagree with Thesis 1 of [50], which basically claims that all kinds of process support tools virtually address the same kinds of problems. In fact, conventional project management systems do not support process enactment, while conventional workflow management systems do not address project planning. Furthermore, the ways processes are modeled in these classes of systems differ considerably. In the case of project management, a plan is built manually at run time. The plan contains only the tasks which actually are to be enacted, and it may be modified at any time during the course of the project. In contrast, a workflow management system instantiates a pre-defined workflow which has to cover all potential execution paths. Furthermore, conventional workflow management systems still fall short of supporting dynamic changes. As a consequence, a project management system cannot replace a workflow management system and vice versa.

Thesis 4 (Integration of Workflow Process Fragments). *Only fragments of the overall development process may be defined as workflows. Thus, these fragments may have to be glued together at the project management level. For supporting integrated process management, project and workflow management have to be integrated or even unified.*

This observation motivated the research presented in Section 3. Development processes may be defined as workflows at best in a fragmentary way. However, if such fragments exist, a workflow management system may provide (partial) process automation. In this case, using a workflow management system improves process support. Those fragments which are relevant at the managerial level have to be embedded into the overall development process (consider, e.g., well-defined and rigorously controlled change request processes in software maintenance). This requires at least an integration of the respective process support systems (as shown in this paper) — or even a unification into a novel process management system, which, however, is not available to date.

5.3 A Priori vs. A Posteriori Integration

Thesis 5 (Economic Value and Necessity). *A posteriori integration allows to retain investments into existing systems. It is an economic necessity in the case of big investments whose replacement would imply a huge development effort. In particular, this applies to management systems for development processes.*

A priori integration means that components of an overall system are designed from the very beginning with a common concept of integration in mind. This approach was realized successfully e.g. in the IPSEN project [51], and it was also applied in the AHEAD system for integrating the management of products, activities, and resources. A priori integration is attractive for several reasons (tight integration, homogeneous development environment, etc.). However, *a posteriori integration* was required by our industrial partners in various projects, and it acted as a driving theme in the IMPROVE project. Companies which have invested into systems for workflow, project, document,

or engineering/product data management need to reuse these systems. Re-development from scratch is not economically feasible.

Thesis 6 (Technical Achievements and Limitations). *A posteriori integration is constrained by limitations imposed by the reuse of existing systems which have been developed independently. Even in the face of these limitations, tight integration may be achieved.*

A posteriori integration is well known to be a challenging task. The functionality of the coupling which may be achieved with the help of a posteriori integration strongly depends on the interfaces provided by the systems to be integrated. However, in the work presented in this paper we succeeded in providing fairly tight integration: Workflows may be monitored in development processes, and dynamic changes may be realized with a conventional workflow management system. This work demonstrates what can be achieved under the constraints of a posteriori integration.

5.4 Barriers of Technology Adoption

As a matter of fact, support technology for development processes has made its way into industrial practice only to a limited extent. In this subsection, we discuss barriers of technology adoption since they are relevant for future work in this area. However, we should stress that these barriers are by no means specific for our own work and have been observed also e.g. in [50,52].

Thesis 7 (Ease of Use). *Ease of use is an essential requirement to process support systems. Users of process support systems expect light-weight, yet powerful support for dynamic development processes. Unfortunately, the tools which have been developed so far do not meet this requirement.*

Potential users of management systems for dynamic development processes expect that they are supported in their processes by tools which are easy to understand, require only little effort in their use, and yet provide a significant added value. To achieve this, systems are needed which hide the underlying and inherent complexity in managing development processes under a simple and carefully designed user interface. This may be explained by an example from software configuration management: The success of version control systems such as CVS [53] or Subversion [54] is due to the fact that they may be operated through a small set of commands which are simple to use (e.g., checkout, update, and commit). Most process management systems are not as easy to use (this also applies to our own work concerning the AHEAD system).

Thesis 8 (Modeling Effort and Return of Investment). *Modeling of development processes is inherently difficult, in particular if it comes to modeling processes to be enacted. The effort of modeling needs to be balanced against the improved process support achieved with the help of the model.*

In the scenarios which we studied in engineering disciplines, it was always hard to come up with a process definition. Usually, process knowledge was only implicit (i.e., domain

experts had acquired process knowledge which was not documented in any way), and we were faced with the task of defining process definitions which are reusable for a sufficiently large set of development processes. It was inherently difficult to define types of tasks, inputs, outputs, control, data, and feedback flows, etc. For a process definition, both the structure and the behavior need to be specified precisely so that the definition may be used to drive enactment. This is far more challenging than creating a descriptive and informal process definition for documentation or guidance. For gaining acceptance of the prospective users of a process management system, it is crucial that the modeling effort may be balanced against the improved process support such that an appropriate return of investment is achieved. An initial step towards this goal is provided by the AHEAD system, which may be used "out of the box" without any prior modeling at all. However, more research is still required with respect to ease of use and return of investment.

6 Conclusion

We presented a management system for dynamic development processes which integrates the management of products, activities, and resources and provides for seamless interleaving of planning and enactment. Furthermore, we investigated the application of workflow management systems to dynamic development processes and described two complementary solutions. The first solution combines process management and workflow management by integrating workflows into a coherent overall development process. The second solution improves the capabilities of a classical workflow management system by adding components for dynamic changes.

Resuming the topics of the discussion in Section 5, we envision the need of future research activities in the following areas:

Dynamic Processes in Different Application Domains. As we have argued above, process support systems cannot be reasonably classified with respect to their application domains. Rather, each process support system is built for some class of processes which are characterized by structural properties (e.g., degree of structuring, degree of automation, granularity of process support, organizational scale; see [16], Chapter 4). Further work is required to elaborate on these properties and the implications on tool support.

Integrating Project and Workflow Management. From our work, we conclude that project and workflow management systems typically operate at different levels of granularity and satisfy different requirements, e.g., with respect to the dynamics of work processes. While we have developed several integration approaches, we still believe that further work has to be performed to clarify the borderlines between and the cooperation of project and workflow management systems and to investigate the potentials for unification.

A Posteriori Integration. As a matter of fact, there is no way around a posteriori integration, which aims at integrating existing tools into an overall environment providing an added value to its users. We are convinced that integration or extension of existing systems is far superior from an economic point of view and more feasible than building a process management system from scratch, which would have

to cover the whole process scope and would have to compete with existing systems in terms of functionality, maturity, and compliance to standards. While we have successfully realized different models of integration, our work still has to be generalized beyond the integration or extension of specific systems.

Ease of Use and Return of Investment. While a rich variety of mechanisms for supporting dynamic development processes has been proposed and realized, the impact of research on industrial practice is still limited. We have identified ease of use as a critical factor determining technology adoption. More research has to be carried out to provide light-weight, yet powerful support for dynamic development processes which provides a significant return of investment on behalf of the users of process support systems.

Acknowledgments. All work reported in this paper has been performed at Department of Computer Science 3, RWTH Aachen University, under the supervision of Manfred Nagl, who served as a speaker of three consecutive research projects funded by the Deutsche Forschungsgemeinschaft (DFG), namely SUKITS, IMPROVE, and the technology transfer project T6. Support from our industrial partners Generali Deutschland Informatik Services GmbH and Comos Industry Solutions is also gratefully acknowledged.

References

1. Nagl, M., Marquardt, W. (eds.): Collaborative and Distributed Chemical Engineering - From Understanding to Substantial Design Process Support. LNCS, vol. 4970. Springer, Heidelberg (2008)
2. Jablonski, S., Bußler, C.: Workflow Management — Modeling Concepts and Architecture. International Thomson Publishing, Bonn (1996)
3. Bauer, T.: Kooperation von Projekt- und Workflow-Management-Systemen. Informatik - Forschung und Entwicklung 19, 74–86 (2004)
4. Maurer, F., Dellen, B., Bendeck, F., Goldmann, S., Holz, H., Kötting, B., Schaaf, M.: Merging Project Planning and Web-Enabled Dynamic Workflow Technologies. IEEE Internet Computing 4(3), 65–74 (2000)
5. Chan, K., Chung, L.: Integrating Process and Project Management for Multi-Site Software Development. Annals of Software Engineering 14, 115–143 (2002)
6. Bussler, C.: Workflow instance scheduling with project management tools. In: Proceedings of the 9th International Workshop on Database and Expert Systems Applications (DEXA 1998), Washington, DC, USA, pp. 753–758. IEEE Computer Society, Los Alamitos (1998)
7. Comos Industry Solutions (April 2009), `http://www.comos.com`
8. Intergraph: Smartplant enterprise (March 2010), `http://www.intergraph.com/global/de/ppm/spe.aspx`
9. Heller, M., Jäger, D., Krapp, C.A., Nagl, M., Schleicher, A., Westfechtel, B., Wörzberger, R.: An adaptive and reactive management system for project coordination. In: Nagl, M., Marquardt, W. (eds.) Collaborative and Distributed Chemical Engineering. LNCS, vol. 4970, pp. 300–366. Springer, Heidelberg (2008)
10. Nagl, M., Westfechtel, B., Schneider, R.: Tool support for the management of design processes in chemical engineering. Computers and Chemical Engineering 27, 175–197 (2003)

11. Heller, M., Jäger, D., Schlüter, M., Schneider, R., Westfechtel, B.: A management system for dynamic and interorganizational design processes in chemical engineering. Computers and Chemical Engineering 29, 93–111 (2004)
12. Krapp, C.A.: An Adaptable Environment for the Management of Development Processes. Aachener Beiträge zur Informatik, vol. 22. Augustinus Buchhandlung, Aachen (1998)
13. Jäger, D.: Unterstützung übergreifender Kooperation in komplexen Entwicklungsprozessen. Aachener Beiträge zur Informatik, vol. 34. Augustinus Buchhandlung, Aachen (2003)
14. Schleicher, A.: Management of Development Processes: An Evolutionary Approach. Informatik. Deutscher Universitäts-Verlag, Wiesbaden (2002)
15. Heller, M.: Dezentralisiertes sichtenbasiertes Management übergreifender Entwicklungsprozesse. Informatik. Shaker Verlag, Aachen (2008)
16. Westfechtel, B.: Models and Tools for Managing Development Processes. LNCS, vol. 1646. Springer, Heidelberg (1999)
17. Heimann, P., Joeris, G., Krapp, C.A., Westfechtel, B.: DYNAMITE: Dynamic task nets for software process management. In: Proceedings of the 18th International Conference on Software Engineering (SE 1996), Berlin, Germany, pp. 331–341. IEEE Computer Society Press, Los Alamitos (March 1996)
18. Hai, R., Heller, M., Marquardt, W., Nagl, M., Wörzberger, R.: Workflow support for interorganizational design processes. In: Marquardt, W., Pantelides, C. (eds.) 16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering, Garmisch-Partenkirchen, Germany. Computer-Aided Chemical Engineering, vol. 21, pp. 2027–2032. Elsevier, Amsterdam (2006)
19. Heller, M., Nagl, M., Wörzberger, R., Heer, T.: Dynamic Process Management Based Upon Existing Systems. In: Nagl, M., Marquardt, W. (eds.) Collaborative and Distributed Chemical Engineering. LNCS, vol. 4970, pp. 733–748. Springer, Heidelberg (2008)
20. Heer, T., Briem, C., Wörzberger, R.: Workflows in dynamic development processes. In: Ardagna, D., Mecella, M., Yang, J. (eds.) Business Process Management Workshops, Milano, Italy. Lecture Notes in Business Information Processing, vol. 17, pp. 266–277. Springer, Heidelberg (2008)
21. Voorhoeve, M., van der Aalst, W.M.P.: Ad-hoc workflows: Problems and solutions. In: Wagner, R. (ed.) Proceedings 8th International Workshop on Database and Expert Systems Applications (DEXA 1997), Toulouse, France, pp. 36–40. IEEE Computer Society Press, Los Alamitos (September 1997)
22. Heinl, P., Horn, S., Jablonski, S., Neeb, J., Stein, K., Teschke, M.: A comprehensive approach to flexibility in workflow management systems. In: Georgakopoulos, D., Prinz, W., Wolf, A.L. (eds.) Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration (WACC 1999), San Francisco, CA. ACM SIGSOFT Software Engineering Notes, vol. 24-2, pp. 79–88. ACM Press, New York (March 1999)
23. Reichert, M., Dadam, P.: ADEPTflex — Supporting Dynamic Changes of Workflows Without Loosing Control. Journal of Intelligent Information Systems 10(2), 93–129 (1998)
24. Joeris, G., Herzog, O.: Managing evolving workflow specifications. In: Proceedings of the International Conference on Cooperative Information Systems (CoopIS 1998), New York, NY, pp. 310–321. IEEE Computer Society Press, Los Alamitos (1998)
25. Wörzberger, R., Ehses, N., Heer, T.: Adding support for dynamics patterns to static business process management systems. In: Pautasso, C., Tanter, É. (eds.) SC 2008. LNCS, vol. 4954, pp. 84–91. Springer, Heidelberg (2008)
26. Wörzberger, R.: Management dynamischer Geschäftsprozesse auf Basis statischer Prozessmanagementsysteme. Aachener Informatik-Berichte, Software Engineering, vol. 2. Shaker Verlag, Aachen (2010)

27. Jäger, D., Schleicher, A., Westfechtel, B.: Using UML for software process modeling. In: Nierstrasz, O., Lemoine, M. (eds.) ESEC/FSE 1999. LNCS, vol. 1687, pp. 91–108. Springer, Heidelberg (1999)
28. Schleicher, A., Westfechtel, B.: Beyond stereotyping: Metamodeling approaches for the UML. In: IEEE Hawaii International Conference on System Sciences (HICSS-34), Minitrack Unified Modeling Language: A Critical Review and Suggested Future, Maui, HI, pp. 1–10 (January 2001)
29. Schneider, R., Westfechtel, B.: A scenario demonstrating design support in chemical engineering. In: Nagl, M., Marquardt, W. (eds.) Collaborative and Distributed Chemical Engineering. LNCS, vol. 4970, pp. 39–60. Springer, Heidelberg (2008)
30. Lawrence, P. (ed.): Workflow Handbook. John Wiley & Sons, Chichester (1997)
31. Workflow Management Coalition: Workflow process definition interface – XML process definition language (XPDL), version 1.0 (April 2002), `http://www.wfmc.org/standards/XPDL.htm`
32. OASIS: OASIS Web Services Business Process Execution Language Version 2.0 (April 2007), `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf`
33. Finkelstein, A., Kramer, J., Nuseibeh, B. (eds.): Software Process Modelling and Technology. Advanced Software Development Series. Research Studies Press (John Wiley & Sons), Chichester, UK (1994)
34. Derniame, J.C., Baba, A.K., Wastell, D. (eds.): Software Process: Principles, Methodology, and Technology. LNCS, vol. 1500. Springer, Heidelberg (1999)
35. Hagen, C., Alonso, G.: Exception handling in workflow management systems. IEEE Transactions on Software Engineering 26(10), 943–958 (2000)
36. Jablonski, S.: Do we really know how to support processes? Considerations and reconstruction. In: Engels, G., Lewerentz, C., Schäfer, W., Schürr, A., Westfechtel, B. (eds.) Graph Transformations and Model-Driven Engineering: Essays Dedicated to Manfred Nagl on the Occasion of his 65^{th} Birthday. LNCS, vol. 5765, pp. 393–410. Springer, Heidelberg (2010)
37. Bolcer, G.A., Taylor, R.N.: Endeavors: A process system integration infrastructure. In: Proceedings of the 4th International Conference on the Software Process, Brighton, England, pp. 76–89. IEEE Computer Society Press, Los Alamitos (December 1996)
38. Bandinelli, S., Fuggetta, A., Ghezzi, C.: Software process model evolution in the SPADE environment. IEEE Transactions on Software Engineering 19(12), 1128–1144 (1993)
39. Jaccheri, M.L., Conradi, R.: Techniques for process model evolution in EPOS. IEEE Transactions on Software Engineering 19(12), 1145–1156 (1993)
40. Cugola, G., Nitto, E.D., Fuggetta, A., Ghezzi, C.: A framework for formalizing inconsistencies and deviations in human-centered systems. ACM Transactions on Software Engineering and Methodology 5(3), 191–230 (1996)
41. Cugola, G.: Tolerating deviations in process support systems via flexible enactment of process models. IEEE Transactions on Software Engineering 24(11), 982–1001 (1998)
42. Casati, F., Ceri, S., Pernici, B., Pozzi, G.: Workflow evolution. In: Thalheim, B. (ed.) ER 1996. LNCS, vol. 1157, pp. 438–455. Springer, Heidelberg (1996)
43. Kradolfer, M., Geppert, A.: Dynamic workflow schema evolution based on workflow type versioning and workflow migration. In: Proceedings of the International Conference on Cooperative Information Systems (CoopIS 1999), Edinburgh, pp. 104–114. IEEE Computer Society Press, Los Alamitos (September 1999)
44. Maurer, F., Dellen, B., Bendeck, F., Goldmann, S., Holz, H., Kötting, B., Schaaf, M.: Merging project planning and web-enabled dynamic workflow technologies. IEEE Internet Computing 4(3), 65–74 (2000)
45. Enhydra.org Community: Enhydra Shark – Java Open Source XPDL workflow, version 1.1-2 (2005), `http://www.enhydra.org/workflow/shark/index.html`

46. Weisemöller, I.: Verteilte Ausführung dynamischer Entwicklungsprozesse in heterogenen Prozessmanagementsystemen. Master's thesis, RWTH Aachen University (2006)
47. Weber, B., Rinderle, S.B., Reichert, M.: Change patterns and change support features in process-aware information systems. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 574–588. Springer, Heidelberg (2007)
48. Vossen, G., Weske, M.: The WASA approach to workflow management for scientific applications. In: Dogac, A., Kalinichenko, L., Ozsu, M.T., Sheth, A. (eds.) Workflow Management Systems and Interoperability. ASI NATO Series, Series F: Computer and Systems Sciences, vol. 164, pp. 145–164. Springer, Berlin (1999)
49. Casati, F., Ceri, S., Paraboschi, S., Pozzi, G.: Specification and implementation of exceptions in workflow management systems. ACM Transactions on Database Systems 24(3), 405–451 (1999)
50. Conradi, R., Fuggetta, A., Jaccheri, M.L.: Six theses on software process research. In: Gruhn, V. (ed.) EWSPT 1998. LNCS, vol. 1487, pp. 100–104. Springer, Heidelberg (1998)
51. Nagl, M. (ed.): Building Tightly Integrated Software Development Environments: The IPSEN Approach. LNCS, vol. 1170. Springer, Heidelberg (1996)
52. Cugola, G., Ghezzi, C.: Software processes: a retrospective and a path to the future. Software Process: Improvement and Practice 4(3), 101–123 (1998)
53. Vesperman, J.: Essential CVS. O'Reilly, Sebastopol (2006)
54. Collins-Sussman, B., Fitzpatrick, B.W., Pilato, C.M.: Version Control with Subversion. O'Reilly, Sebastopol (2004)