In International Journal of Agile Manufactoring (IJAM), Special issue on 'Information Management for Productivity Enhancement', 2000 (forthcoming).

FLOW.NET: WORKFLOW SUPPORT FOR INTER-ORGANIZATIONAL ENGINEERING AND PRODUCTION PROCESSES

Gregor Joeris, Holger Wache, Otthein Herzog
Intelligent Systems Department, TZI, University of Bremen,
PO Box 330 440, D-28334 Bremen, Germany
joeris|wache|herzog@tzi.de

Britta Gronemann
VSS GmbH, Am Fallturm 9, D-28359 Bremen, Germany
bgronema@vss.com

Abstract. On the basis of new information technologies, today's business evolves towards distributed and cooperative service processing resulting in a network of co-operating organizations. This paradigm also influences engineering and production processes. Both the characteristics of these processes and the interoperability between different organizations challenge the development of enabling technologies to support flexible and inter-organizational product development. Currently available workflow and document/data management technologies fail to support these distributed and collaborative processes. They lack abilities for supporting inter-organizational processes on the level of workflow as well as of document management. Furthermore, only little support is provided for collaborative workflows.
In this paper, we present the *flow.net* approach which has been developed within the MOKASSIN project. It is based on a workflow component which has been designed to support distributed, dynamic and collaborative process management across several organizations. The standard workflow management architecture is extended with integrated document management services including version and workspace control. In order to support inter-organizational processes, our approach provides a distribution layer which supports both decentralized workflow and document management.

Keywords. workflow management, collaborative workflow, inter-organizational processes, concurrent engineering, document management

1. INTRODUCTION

The trend towards global markets and the increasing customer orientation give rise to new business paradigms like agile manufacturing, virtual enterprises or business-to-business e-commerce. Due to the growth of production in temporary networks these paradigms require flexible communication, coordination, and cooperation facilities where information technologies act as enabler (cf. [17, 2, 20, 16]). The basic communication infrastructure (e.g. the internet) is nowadays available to nearly all companies. However, the characteristics of product development processes and the interoperability between different organizations make certain requirements on process-supporting software systems (e.g. workflow management systems) [15]. Available workflow and document/data management technologies fail to support these distributed and collaborative processes. They lack abilities for supporting inter-organizational processes on the level of workflow as well as of document management (cf. [5, 12]).

In this paper, we present the *flow.net* approach to distributed collaborative workflow management which has been developed within the MOKASSIN project. The approach aims at supporting *dynamic processes across several organizations.* In order to provide comprehensive coordination and cooperation support, we extend the standard workflow management architecture [23] with integrated document management services including version, workspace, and document life-cycle support. These services are not only integrated on an architectural level but also on a conceptual level. Therefore, interdependencies between workflow and data models can be defined. In particular, this results in a flexible dataflow semantics which takes into account versioning and different types of data interchange between the activities' workspace.

In order to support *inter-organizational processes*, our approach provides an import/export model for process interfaces and a distribution layer which supports distributed and decentralized workflow and document management. The distribution layer allows nested as well as asynchronous invocations with synchronization points among different workflow servers. Furthermore, workflows can be monitored across different workflow servers. Finally, different data interchange models are provided between different repositories on the basis of replication services.

Section 2 illustrates a real-world case study and identifies the main challenges. Section 3 gives an overview of the advanced workflow and document management services. Section 4 extends this view to cross-organizational settings. Finally, section 5 gives a short conclusion.

2. CASE STUDY AND REQUIREMENTS

At the beginning of the MOKASSIN project inter-organizational processes of different companies have been analyzed in order to identify the requirements for an advanced process management system (cf. [13]). In one case study within the aerospace industry, a company *Prod* assembles a large unicum, whose parts are manufactured by itself and by suppliers, e.g. company *Supp*. *Prod* defines product and process properties for the parts manufactured by *Supp*. During this inter-organizational process a lot of information must be exchanged. Figure 1 sketches the processes and their document dependencies. In the two company boxes the circles indicate high-level process steps. The arrows describe control flow between the process steps including the cross-organizational control flow (solid) and information/document interchange (pointed).

Before we discuss the main challenges, we describe the existing process in more detail. Normally the process should follow an idealistic producer/supplier interaction. *Prod* designs the unicum including all of its parts to a certain degree of refinement *(development task;* the interactions during this engineering task is not shown in this example but also of high interest). Then all parts are manufactured and integrated *(production task)*. In case of an external production an *order* is sent to the appropriate supplier (1). The sub-process in *Prod* is pending until the desired piece is delivered to *Prod*. On the supplier's side the received order (1) initiates the *order acceptation task.* According to the specification which is included in the order the requested piece is manufactured *(manufacturing task)* and delivered to *Prod (delivery task)*.

In reality this production process is often disturbed by incomplete or erroneous instructions. These problems are detected mostly during the manufacturing task of *Supp* and make the process much more complex. Depending on the kind of problem *Supp* has two alternatives for his reaction *(change request initiation task).* If the instruction documents are incomplete, *Supp* directly requests the missing information (2). If an error is discovered, first a discussion with *Prod (problem discussion task)* is needed (3). During the discussion several documents including error reports are exchanged and the kind of error protocol is determined. The appropriate error sheet is filled out and

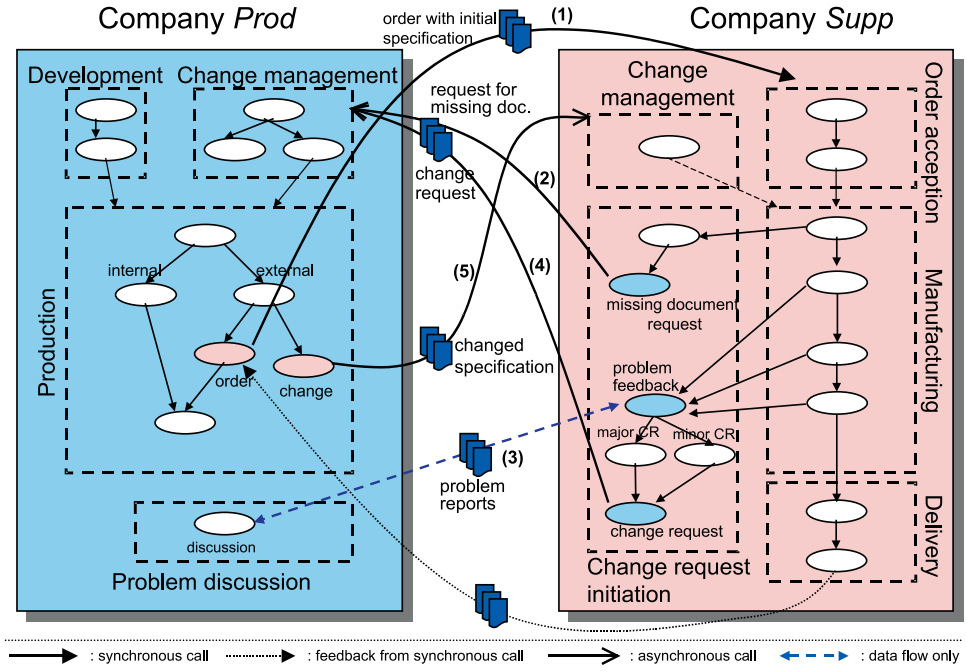

Fig. 1: Case study of inter-organizational workflows

sent to *Prod* (4). Both cases (2)+(4) are reasons for initiating the *change management task* in *Prod*. In this task *Prod* analyzes the change request. Sometimes a change request may be rejected because e.g. the changes would be too expensive. But normally the documentation is revised and the changed specification is re-sent to the supplier (5). However, during the change management task in *Prod* the work by *Supp* can be at least partially continued. If then an updated specification is received, the engineering and production data is revised according to the changes in the specification.

A system which supports such inter-organizational processes and document interchanges has to meet several requirements. The case study shows different types of process interactions and document interchanges. A process in the receiving company is normally initiated with an information interchange. E.g., the external production task of *Prod* initiates the order acception task at *Supp* (1). These two processes operate *synchronously*, i.e. the initiating process can only be continued when the initiated process is finished successfully (in this case, when the delivery task has been finished).

But also an *asynchronous* process interaction between the companies is needed: a process initiates another process but continues in its normal process flow without waiting for any answer. For example, the change request initiation task is started by the manufacturing task and invokes the change management task in *Prod* asynchronously (2)+(4). It does not wait for any direct result but terminates after the initiation. A synchronous interaction, i.e. waiting until the revised specification is received from *Prod*, is not appropriate because not every change request implies a revision of documents. For example several different change requests can be combined resulting in one set of revised documents, or documents can be changed without any change requested by *Supp*.

Furthermore, document interchange must not always imply the initiation of a new process. Rather, documents are interchanged between *running* tasks and synchronize the tasks implicitly. An example can be seen in the discussion task, where several documents are exchanged until the control flow continues (3).

Finally, in an inter-organizational environment the documents have to be managed over distributed repositories owned by each company. For optimal support, integration of the repositories in one logical document management system is required which provides a global view on the interchanged documents – which are still under local control – and manages the changes. Furthermore, these services have to be integrated into the workflow management system in order to provide integrated coordination for control flow and data flow.

3. PROCESS MANAGEMENT IN *FLOW.NET*

The *flow.net* process management system has been designed to support dynamic and collaborative processes across several organizations. The backbone of the system is a flexible and adaptable workflow management component which has been extended by document management services in order to support cooperative activities within a workflow-driven process. The components, the modeling and enacting approach, the flexibility aspects, and the integration of document management services are discussed in the sequel.

3.1 Components

Support for concurrently acting humans and systems in engineering and manufacturing domains cannot be reduced to the workflow level. Work among different actors has to be integrated on workflow, actor, and document level and has to take organizational boundaries into account. With respect to these requirements, the reference architecture of the Workflow Management Coalition (WfMC) [23] has several limitations which we outline in the following.

The reference architecture of the WfMC distinguishes five different components: A component for the definition of the workflows (*ProcessDefinitionTool)*, the central component to execute workflows (*WorkflowEnactmentServices)*, the clients which handle the personal work list of an actor (*Workflow Participant Client)*, the external applications which are invoked by the WFMS (*Invoke-*

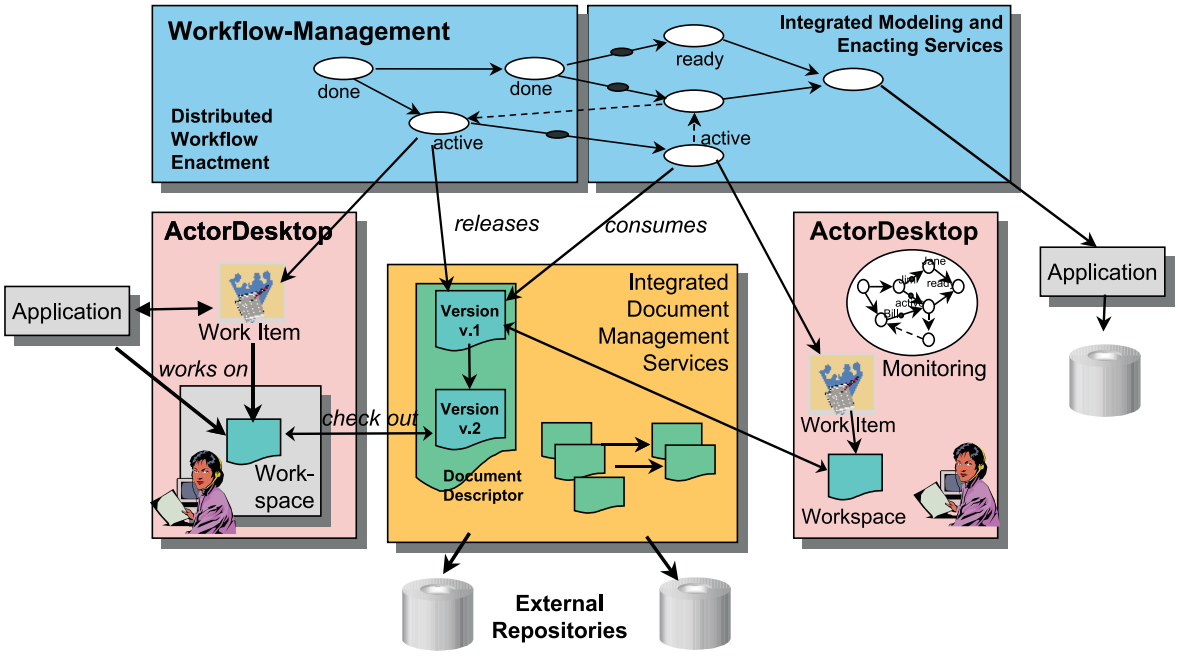

Fig. 2: Extended workflow management services

dApplications), and finally the component for workflow administration and control (*Administration&Monitoring).* The limitations of this architecture in our scenario are:

- Build- and run-time components are separated. Therefore, process schema changes during runtime are not supported.
- Document management or repository services are not taken into account. However, in our scenarios document management functionality is very important.
- The 'workflow participant clients' provide only a limited view for the actors since they only handle their work lists. Information about the process context, monitoring functions, and user intervention capabilities are located in a different component.
- Interoperability between different WFMS is defined on a technical level between their enactment services. On the process definition level, integration concepts for independently and decentralized specified inter-organizational workflows are missing (cf. [14]).

The architecture of the *flow.net* process management system was designed taking the above mentioned requirements into consideration. Therefore, it includes some extensions in comparison with the reference architecture (illustrated in figure 2). The system has been built as a distributed object system using the CORBA standard and a relational database for persistency services.

- The architecture has been extended *by document management services* which provide uniform data retrieval and storage as well as access control. Furthermore, these services are in charge of version control and - in conjunction with the ActorDesktop (see below) - of work space management. The document management services maintain only information about documents and versions as well as their application within a task (so-called meta-information), and delegate the physical storage of documents to external repositories. This ensures a logically integrated view on the documents produced and used within a workflow and supports the integration of heterogeneous document storage systems.
- The so-called *ActorDesktop* extends the limited functionality of worklist management in order to support collaborative workflows. It provides the virtual environment in which an actor performs its activities. This includes managing the personal work list (task space) as well as all needed documents for performing a task (the workspace). A workspace follows the desktop paradigm and provides uniform and local access to all relevant documents needed for performing a task. The ActorDesktop also provides monitoring functionality in order to control the execution states of subtasks and to visualize the process context. Finally, it supports dynamic process changes.
- External application systems are integrated into the architecture by means of the SystemCallAgent (not further addressed in this paper).
- *Build- and runtime* components have been *fully integrated* in order to support process schema evolution.
- *Inter-organizational processes* can be defined and executed by means of *additional distribution services* which have been added for process definition, workflow enactment, and document management (see Section 4).

3.2 Process Modeling

The building block of modeling a process is a *task definition* (or task type) which consists of a *task interface,* that specifies 'what is to do', and potentially several *workflow definitions,* which specify how the task may be accomplished. Among other things, the task interface is defined by pa-

rameter definitions and by a task behavior definition which specifies the external context-free behavior of a task (e.g., transactional or non-transactional). The context-dependent behavior of a task is defined by its application within a workflow definition. A workflow definition may be atomic consisting only of a task description or system invocation, or complex. A complex process is defined in an activity-oriented manner by a *task graph*. The decision which workflow definition of a task definition is used to perform a task is taken at run-time (late binding). This late binding mechanism also allows the creation of a new workflow definition at run-time (late modeling).

A *task graph* (an example is shown in figure 3) consists of task components, connectors, start and end nodes, and data inlets and outlets, which are linked by control and data flow dependencies: A *task component* is an applied occurrence of a task definition within a workflow definition. For every task component a split and join type (and, or, xor) can be specified. Furthermore, connector components are predefined which just realize splits and joins.

Task and connector components are linked by *control flow dependencies*. Iterations within this task graph are modeled by a special predefined feedback relationship. A condition can be associated to every dependency to support conditional branches. Rather than providing a limited set of predefined control flow dependency types, arbitrary control flow dependency types can be defined by a process engineer and can then be applied and reused within several workflow definitions (see [11] for details). This is a very powerful mechanism to define flexible and less-restrictive workflows as we demonstrate in the next sub-section.

Finally, task components can be linked by *dataflow relationships* according to the input and output parameters of their task definitions. Furthermore, a data inlet (or outlet) is used as a data source (or sink) in order to realize a vertical dataflow between the parameters of the task definition and their use within the workflow.

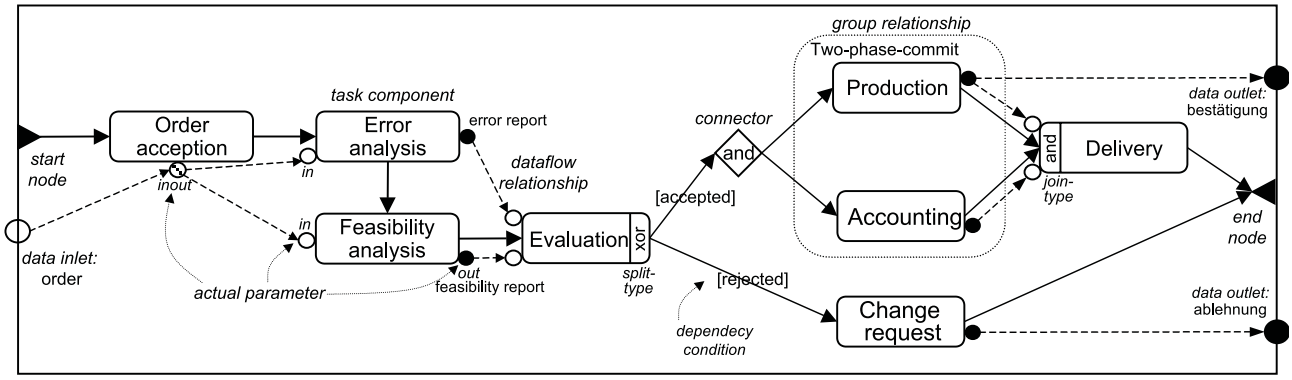

Fig. 3: Sample task graph

3.3 Distributed Process Enactment

Our enactment model is based on treating tasks as *reactive components* which encapsulate their internal behavior and interact with other tasks by message/event passing. This is a natural basis for a *distributed enactment* of workflows: instead of interpreting a workflow instance by a (centralized) workflow engine, a workflow is directly enacted by distributed task instance agents. Every task agent behaves as defined in the schema by its task type and by the component of a workflow definition it corresponds to. A task has several built-in operations categorized into state transition, actor assignment operations, operations for handling of (versioned) inputs and outputs, and workflow change operations. For every operation, the task has the knowledge about when to trigger the op-

eration, a condition that must hold for executing the transition (guard), and a list of receivers to which events are passed. We denote this approach as light-weight agent-oriented, since the knowledge of how to react on events is explicitly represented and decoupled from the built-in operations.

We skip the definition of the execution semantics of the tasks and the task graphs (which are defined by a statechart variant and event-condition-action (ECA) rules; see [11] for details) and give an example instead: A typical workflow enactment with traditional control flows is as follows: A task is started and in most cases a workflow is selected automatically creating sub-tasks in the case of a complex workflow. An event is passed to the first tasks of this workflow, triggering the evaluation of the 'enable' transition. When a task is enabled, role resolution is activated if no actor has yet been assigned explicitly. In the case of automatic tasks, the start transition will be directly triggered by the enable event. When a task is terminated, a corresponding event is sent to all successor tasks, which are connected by a signaled dependency, i.e., whose dependency condition evaluates to true. This again results in the evaluation of the 'enable' transition. Furthermore, for all tasks connected to the end node, the finish event is also sent to the super-task, triggering the termination of the super-task when all sub-tasks have been terminated.

3.4 Support for Flexible Workflow Management

Support for flexible workflows in schema-based workflow management systems has to cope with two fundamental challenges (cf. [3, 9, 18]): First, *a-priori flexibility* focus on the specification of a flexible workflow execution behavior in advance; flexible and adaptable control and data flow mechanisms have to be taken into account in order to support ad hoc and cooperative work at the workflow level. Second, *a-posteriori flexibility* (flexibility by dynamic adaptation) is provided by the change and evolution of workflow models in order to modify workflow specifications on the schema and instance level due to dynamically changing situations of a real process. In *flow.net*, comprehensive support is provided for both aspects (as illustrated figure 4): the first is supported by defining less-restrictive workflow and by providing cooperation support on the basis of flexible and user-adaptable control flow dependency type and a versioned data flow semantics; the second is supported by on-the-fly instance changes as well as complex workflow schema evolution policies which are based on an integrated architecture for representing workflow schema and instance elements and on workflow schema versioning.

Dynamic changes: Based on the object-oriented modeling and enacting approach, dynamic changes can be handled as any ordinary state transition (see [7, 10] for details). Every change primitive is encapsulated by a pre-condition which restricts its application, and by raising a corresponding event which is handled by the affected task instances in order to ensure the behavioral consistency of the execution states. Thus, dynamic changes can be supported also in the context of distributed workflow enactment. Whether a change is allowed and how to react on it highly depends on the particular situation and the behavior of the involved tasks. Therefore, this behavior can be adapted in our approach.

Process schema evolution: Besides changing a single process instance, we support also different process schema evolution strategies by schema versioning and migration rules (see [10]).

Support for collaborative workflows: Collaborative workflows are supported in our approach by integrated document management services and flexible dataflows which provide controlled cooperation on the workflow level. We discuss these issues in the next section.

Defining less-restrictive workflows: A-priori less-restrictive workflows, i.e. workflows where a certain degree of freedom is left open to the actor, can be realized by the concepts of user-defined

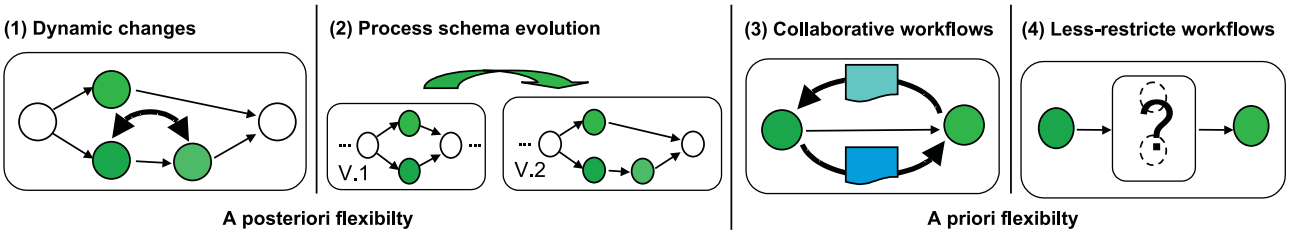

Fig. 4: Support for flexible workflows in *flow.net*

control flow (cf. [8]) and group relationship types. Rather than focussing on details, we outline some sample dependency types which can be realized by this concept:

- An *exclusive* group relationship type forces its members to execute mutually exclusive. In conjunction with parallel branches, we can define that certain tasks should be executed sequentially, but without defining the actual order of the tasks. So, the user can choose which task he or she wants to perform next.
- The *try* dependency type supports parallel branching with final selection (cf. [18]). In this case, several alternatives are activated, but when the first branch has reached the synchronization point the other alternatives are stopped and compensate if necessary and possible.
- A *conditional or soft synchronization* dependency supports synchronization between branches whose activation cannot be determined at build-time but is evaluated at runtime. Moreover, it allows to skip a task without causing indefinite waiting of the dependent task (cf. [18]).
- When using a *simultaneous* dependency type, dependent activities can overlap and may pass intermediate results to subsequent activities. Up to the final version, the supplier may produce several versions of its output and the consuming tasks may obtain several input versions. To avoid chaotic processes, the termination condition is defined as an end-end dependency guaranteeing that all preceding tasks have finished and hence no new input versions will occur (cf. [7]).

3.5 Integrated Document Management and Support for Collaborative Workflows

The integrated document management approach in *flow.net* distinguishes between logical and physical document management. From the logical point of view, documents are managed by the *flow.net* system by maintaining (meta) information about the documents, whereas the physical storage is provided by external repository services. These systems can be integrated using an open and simple repository interface. Furthermore, a default repository is provided within *flow.net* by a relational database. This meta level approach is the basis for providing inter-organizational document management services since this approach naturally supports physically distributed document storage in repositories under the control of different organizations.

Document meta model: The following meta information is managed according to the schema and instance level, respectively:

On *schema level*, different document types can be defined. A document type may be atomic or complex. An atomic document type represents the structure of a certain kind of document (e.g., ChangeRequest), and a document (instance) template can be specified which can be used as the initial version when instantiating a document of this type. A complex document type is modeled in an ER-like manner defining the component document types and their interdependencies. Beside some predefined document attributes also user-defined attributes can be specified for document types. Fi-

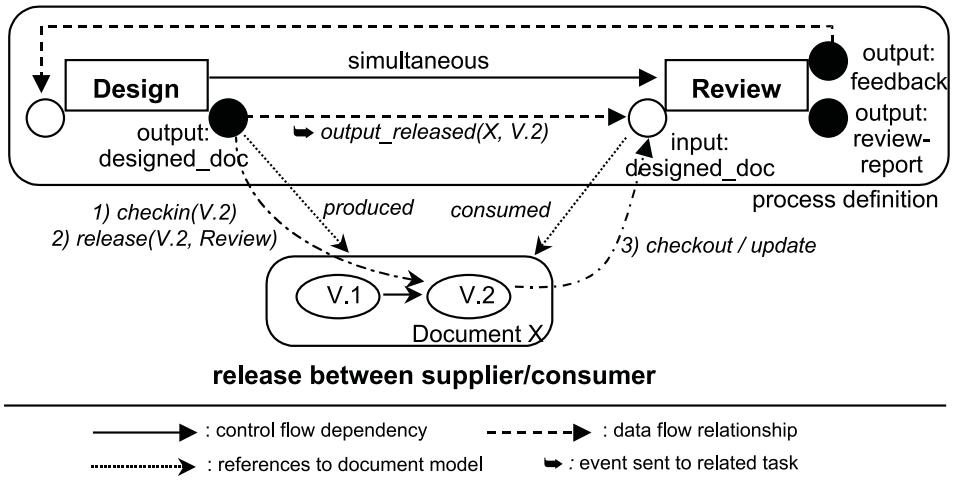

Fig. 5: Extended dataflows

nally, a state transition diagram can be defined for every document type which is used to control the document lifecycle. Document types can be used within a parameter definition of a task type defining the input or output type of the parameter. A parameter definition also permits the specification of some additional access rights. Beside read or write permissions for input or output parameters, respectively, also exclusive and shared access are distinguished.

On *instance level*, information about the actual document is managed, i.e. its state within the lifecycle, the actual structure of complex documents, the history of different document versions, the user's working versions, and so on. In addition, information about the process context are maintained in which documents are used, i.e., which versions have been produced and consumed by a task (see figure 5). The workspace management even provides detailed information about every access made on a specific version of the document. Finally, the dataflow relationships of a workflow definition show clearly information channels between tasks.

Support for collaborative workflows: Collaborative workflows are supported by the version and workspace control capabilities (cf. [4]) of the integrated document management services. Versions can be checked out into the individual workspace of a user for further editing and checked in afterwards. Both pessimistic and optimistic locking are supported. Furthermore, versions can be released for dedicated tasks (as proposed by [7, 21]). In combination with the dataflow specification, this allows for different access modes and data exchange types (see [9]). Beside passing the original document to another task, also intermediate results can be exchanged between running tasks (e.g., using the simultaneous dependency type) and users can work concurrently on private copies of a document (since we neither can assume specific document formats nor a central data/document management merging is provided only for those document types for which a merge operator has been defined). This goes beyond a simple black box view of tasks within a workflow.

Furthermore, the data flow can be used also for control flow purposes. The availability of input data can be checked and the operations for releasing outputs generate events as any other transition so that tasks can react on these events. E.g., the event 'output_released' triggers the evaluation of the enable transition of the consumers (see figure 5). Rather than supporting free cooperation by information sharing, the cooperation can be controlled on workflow level. For example, the above mentioned simultaneous dependency coordinates collaborative tasks and avoids a chaotic process.

4. INTER-ORGANIZATIONAL PROCESS MANAGEMENT

So far, we have focussed on the distributed enactment of centrally modeled processes, but not on the modeling and enacting of inter-organizational processes which become more and more significant. In this case, decentralized process models of different process management systems have to be integrated. Furthermore, document management services for collaborative workflows are also needed in inter-organizational settings.

4.1 Extended Architecture

In order to support the integration of decentralized modeled processes and to provide workflow and document management support for inter-organizational processes we have extended the architecture of our process management system by a distribution layer. This layer supports the interoperation of several process management servers without the need for any central services. It consists of the following enhancements (illustrated in figure 6):

Import/export services: First, on the schema level, import and export services for task types as well as document types are provided. Task and document types can be exported globally or to a dedicated process management server of another organization. Only exported types are visible for an importing server. Furthermore, in the case of task types, their task body is hidden; only the task interface is visible. When a type is imported all type information are replicated in the local process knowledge base. After an import, the imported type can be used within the local process management system like any other type (except that it must not be modified) and disconnected from the exporting server. For example, an imported task type can be used within a workflow definition of the local system. In our case study, for example the task which handles change requests at *Prod* is imported by *Supp* and integrated in the problem handling process. Of course, the semantics is different to a local task, since an imported task will be enacted at the process management system from where it has been imported. This is realized by gateways of the task agents:

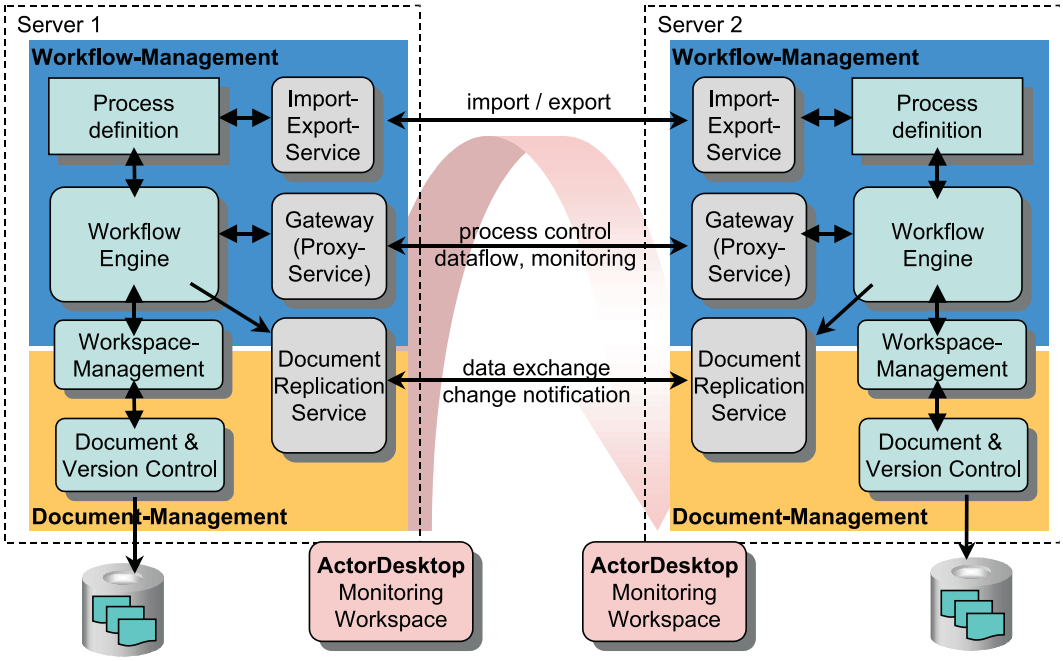

Fig. 6: Extended architecture for inter-organizational process management

Task gateway: A task gateway provides the services of enacting processes across different process management systems. Because of the extended control and data flow capabilities in our approach, this goes beyond simple invocation of a workflow at a remote site (cf. [12]). Rather, process control, dataflow, and monitoring capabilities are provided in both directions and remain transparent. Our approach of distributed enactment of centrally modeled processes by task agents is a very good starting point for the execution and synchronization of processes across different servers, since the task agents already have the knowledge of how to coordinate tasks. The task gateway extents the task agents such that they can inter-operate also in inter-organizational settings, without changing their interface. We explain this in the next sub-section.

Document replication services: Consequently, document management capabilities have to be provided also in inter-organizational processes. In this case, a uniform and secure access, a logically integrated view, but also local control and local management of exchanged documents are needed. The document replication services meet these requirements and provide different policies of distributed document management, as introduced in section 4.3.

4.2 Inter-Organizational Workflow Enactment

In order to support inter-organizational workflow enactment, i.e. inter-server workflow enactment, we have extended the design of the task agents as follows (as shown in figure 7(1)).

Meta model for inter-organizational processes: The abstract class TASK defines the common interface of every task agent; the class LOCALTASK represents task agents which operate on a single server; this is what we have referred to as a task so far. In contrast, the new abstract class DISTRIBUTEDTASK defines the interface for tasks which are distributed across different servers and extends the class TASK by the task gateway. Two types for distributed tasks are separated:

First, the class REALTASK represents the tasks which are really performed at a server (or its corresponding organization). The task definition of those tasks have been exported (but not every instance of an exported task definition has to be a distributed task because the task definition can still be used locally).

Second, the class SHALLOWTASK acts as a surrogate of the real task in the *context of a workflow* at a different server. A shallow task is always an instance of an imported task definition and vice versa. A shallow task is similar but not identical to a proxy service. The shallow task does not only pass requests on to the real task, but also provides services needed by the real task. Since the SHALLOWTASK is embedded in the workflow, it determines the context-dependent behavior of the task, i.e. for example, when the task is ready for execution. Thus, there is a tight and bi-directional interrelationship between shallow task and real task which are always located at different servers.

Invocation types of inter-organizational processes: We distinguish and support two main kinds of invocation of inter-organizational processes (which are also useful for local processes; see figure 7(2); cf. [23]). First, *synchronous* or nested invocation where the caller waits for the termination of the invoked task. In this case, a distributed task is invoked as a normal sub-workflow. This is used in our case study for the main ordering task. Second, *asynchronous* invocation where the invoked task runs concurrently with the calling process. The change management tasks reflect this invocation type in our scenario. Furthermore, synchronization points are useful to control the concurrent execution and can be realized by event passing. A special case of this invocation type is a chained structure in which the last task of a process invokes the next process and hence only one process is active at a time.

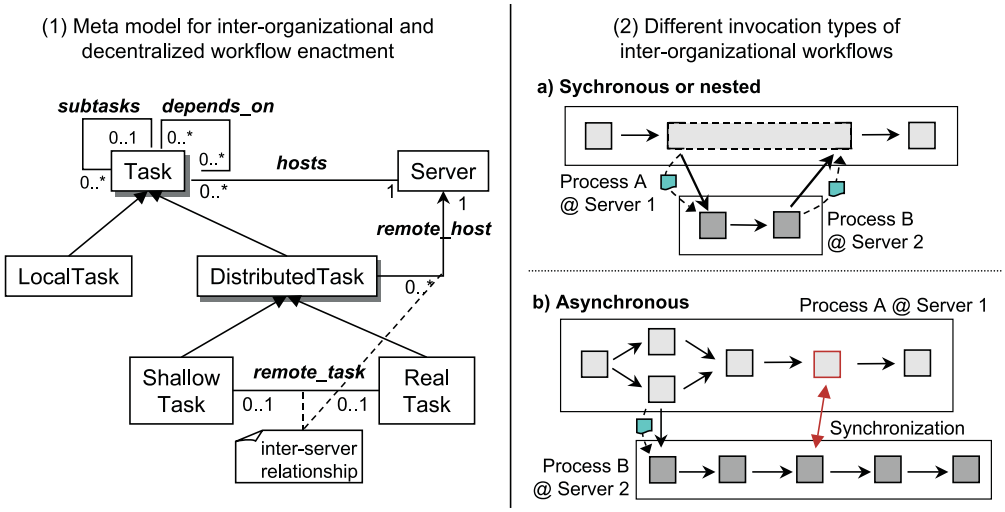

Fig. 7: Meta model and invocation types for inter-organizational workflows

The asynchronous invocation is realized as follows. First, shallow and real task are instantiated and interact as in the case of synchronous invocation; this is needed because the task may be enabled and disabled several times and the work lists of the actors have to be updated. This already shows that a simple remote invocation of a process is not sufficient for inter-organizational workflows. When the real task is started, input data is transferred and the connection between shallow and real task is broken. From now on, both tasks act independently from each other. The state of the shallow task is changed to 'done'. Note, that in the case of an iteration, a new real task is generated for a shallow task so that a new task is initiated and invoked.

4.3 Distributed Document Management

Support for inter-organizational processes is already quite complex. However, inter-organizational engineering and production processes require also distributed document management services in order to support collaborative workflows in these settings. Therefore, a simple functional black box view where the input documents are passed to a remote task, when the task is invoked, and the output documents are returned upon termination of the task is inadequate. Rather, documents have to be exchanged also between running tasks of different organizations. This substantially increases the complexity of inter-organizational workflow and document management.

The *document replication services* extend our integrated document management approach and provide the capabilities needed for distributed document management. Rather than formally introducing the details of these quite complex services, we explain them by giving an example. We concentrate on the management of document instances rather than on the import of document types as already discussed in section 4.1.

In the example shown in figure 8, task B and its input document type, e.g., a change request form CR, are modeled on server 2. They are imported by server 1 and embedded in the local workflow. In this workflow, a local task A precedes the local task C and the imported task B which is represented by its shallow task B'. Task A creates an output of the imported document type which should be passed to task B as its input. This is modeled by a dataflow relationship to its shallow task B'.

Assume, that task A has created a first document version X.v1 and released it for the succeeding tasks. The version is still checked out by task A. The integrated document management services

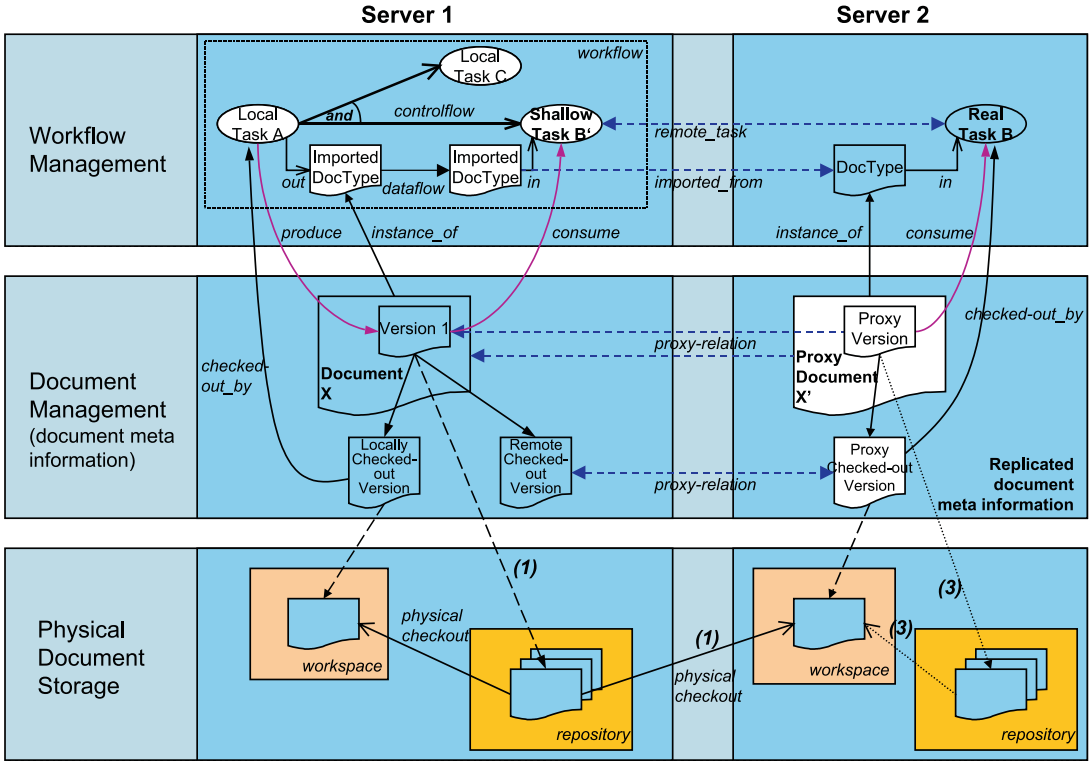

Fig. 8: Distributed document management

maintain all relevant meta information of this situation, whereas the physical document is stored in the repository of server 1 and a copy is located in the personal workspace of the actor of task A.

The release and dataflow information are propagated to task B by means of the shallow task B' so that task B can consume the document version as input. In the case of inter-organizational processes there exist three different ways of managing incoming documents:

1) Check out from producing server ("share"): First, the actor of task B checks out the document from the server 1 (marked in figure 8 by '(1)'). In this case, first a "dry" check out is performed on server 1, i.e., the information about the check out is recorded and marked as a remote check out, and the physical document is provided for data interchange for the consuming server. Because of security reasons and access rights server 1 cannot place the document into the user's workspace at another organization. This is done by the consuming server which transfers the document by ordinary ftp-services. The consuming server also creates local proxy-objects for all relevant entities, i.e. documents, versions, and checked-out versions, which refer to the source objects on server 1. Thus, rather than replicating the physical document, parts of the meta information are replicated and establish a local view of server 2 on the documents managed by server 1. These proxy objects allow for an easy access to new versions of the document and are fundamental for inter-server workspace control.

2) Create new local document ("send"): Second, a new local document is created at the consuming server as a copy of the main document on server 1 (this case is omitted in figure 8). The transferred version is recorded as the first version of this new local document and is stored in the local repository. The new document resides under its own name and is managed at server 2 independently from the producer and the original document. In particular, the different servers exchange no change notification.

3) Replication: The third case is a combination of the above types. Server 2 also creates a local document by copying the original document, but the relation between the two documents is kept. The document is stored in the local repository, which allows independent access control, but the possibilities of change notification, workspace updates, and comparison of the two documents remains. This policy combines local control and local management of exchanged documents with a uniform and logically integrated view on the documents.

The same document management policies are provided when an output is created by a remote task at another server and is transferred back to the calling workflow. For example, assume that task B produces an output. This output is visible by means of the shallow task B' within the local workflow at server 1. Then, a succeeding task, say task D, can consume this output document from the producer via task B'.

4.4 A Brief Overview of Related Work

Inter-organizational workflow management is a new research area which is currently of increasing interest. Different projects like PhantaRei [6], WISE [1], CrossFlow [14], or [22] focus on this challenge, but they do not support collaborative workflows and do not address distributed document management. The WISE project provides IT services in order to support virtual enterprises (VU). Beside process definition and enactment services which are the backbone for powering a VU, also tools for cooperative browsing, a web catalogue of the services provided by a certain enterprise, and video conferencing tools have been integrated into the WISE architecture. The CrossFlow project investigates all issues which are concerned with business processes crossing organizational boundaries. A contract-based approach is used to define the business relationships between the organizations. Within this contractual basis, inter-organizational processes can be defined and performed. These surrounding organizational policies of enabling inter-organizational workflows is missing in *flow.net* so far. Finally, document-oriented inter-organizational workflows have been investigated by [22]. However, document-based WFMS are neither sufficient as a general-purpose process support system nor for enabling process-oriented business-to-business relationships.

On the other hand, commercial collaborative workflow management systems like InConcert [19] from InConcert Inc. or TeamWARE Flow from Fujitsu do not adequately support inter-organizational workflows. They rely on supporting the WfMC interoperability standard interface which is not sufficient for making inter-organizational workflows possible.

5. CONCLUSION

In this paper we have presented the *flow.net* approach to distributed collaborative workflow management which supports inter-organizational engineering and production processes by flexible and dynamic workflow management and integrated document management services. An extended CORBA-based architecture provides these services also in inter-organizational settings without the need of any centralized services. The application scenario presented in section 2 has been fully implemented within the *flow.net* system. We have realized a tight integration between different process management systems, but have neither focussed on the interoperability of heterogeneous workflow systems nor on a loosely coupled integration on the basis of messaging APIs as addressed by standardization efforts (e.g. [23]). This will be part of future work. Furthermore, we will investigate the integration of additional CORBA services in the implementation of our prototypical system.[1]

[1] This work has been supported by the German Ministry for Research and Technology (BMBF) , project MOKASSIN under grant number 01 IS 606 B

REFERENCES

[1] Alonso, G.; Fiedler, U.; Hagen, C.; Lazcano, A.; Schuldt, H.; Weiler, N. "WISE: Business to business E-Commerce." *9th Int. Workshop on Research Issues on Data Engineering (RIDE-VE'99)*, 1999.

[2] Borghoff, U. et al. (eds.) Information Technology for Knowledge Management, Springer: Berlin, 1998.

[3] Ellis C.A.; Nutt, G.J. "Workflow: The Process Spectrum" *NSF Workshop on Workflow and Process Automation in Information Systems*, 1996.

[4] Estublier, J. "Work Space Management in Software Engineering Environments." *Software Configuration Management (SCM-6 Workshop)*, LNCS 1167, Springer, Berlin, 1996, pp. 127-138.

[5] Georgakopoulos D., Hornick M., Sheth A. "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure." *Distributed and Parallel Databases*, 1995, 3(2), pp. 119-153.

[6] Groiss, H.; Eder, J. "Workflow Systems for Inter-Organizational Business Processes." *ACM SIGGROUP Bulletin 18*, 1997.

[7] Heimann, P.; Joeris, G.; Krapp, C.-A.; Westfechtel, B. "DYNAMITE: Dynamic Task Nets for Software Process Management" *18th Int. Conf. on Software Engineering*, 1996, pp. 331-341.

[8] Jablonski, St.; Bussler, Ch. Workflow Management - Modeling Concepts, Architecture and Implementation. International Thomson Computer Press, London, 1996.

[9] Joeris, G. "Cooperative and Integrated Workflow and Document Management for Engineering Applications." *8th Int. Workshop on Database and Expert System Applications, Workshop on Workflow Management in Scientific and Engineering Applications*, 1997, pp. 68-73.

[10] Joeris, G; Herzog, O. "Managing Evolving Workflow Specifications" *3rd Int. IFCIS Conf. on Cooperative Information Systems (CoopIS'98)*, 1998,pp. 310-319.

[11] Joeris G.; Herzog O. "Towards Flexible and High-Level Modeling and Enacting of Processes" *11th Int. Conf. on Advanced Information Systems Engineering*, LNCS 1626, Springer, 1999, pp. 88-102.

[12] Juopperi, J. et al. "Usability of some workflow products in an inter-organizational setting." *IFIP WG8.1 Working Conference on Information Systems for Decentralized Organizations*, 1995.

[13] Kuhlmann, T.; Lamping, R.; Maßow, Ch. "Agent-Based Interorganisational Workflow." *Int. Conference on Management and Control of Production Logistics*, Brazil, 1997.

[14] Ludwig, H.; Hoffner, Y. "Contract-based Cross-Organizational Workflows - The CrossFlow Project." *WACC'99: Workshop on Cross-Organizational Workflow Management and Co-ordination*, 1999.

[15] Morschheuser, S.; Raufer, H.; Wargitsch, C. "Challenge and Solutions of Document and Workflow Management in a Manufacturing Enterprise: A Case Study." *29th Annual Hawaii Int. Conference on System Sciences (HICSS)*, 1996, pp. 4-13.

[16] Parish, J.D. "Paradigm Shift to Agile Manufacturing." *Int. Journal on Agile Manufacturing*, 1999, 2(1), pp. 87-92.

[17] Prasad, B. Concurrent Engineering Fundamentals - Integrated Product and Process Organization. Prentice-Hall International Series in Industrial and Systems engine, Vol. I: Englewood Cliffs, NJ, 1996.

[18] Reichert, M; Dadam, P. "ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control" *Journal of Intelligent Information Systems - Special Issue on Workflow Managament*, 1998, 10(2), pp. 93-129.

[19] Sarin, S. K.; Abbott, K. R.; McCarthy, D. R. "*A Process Model and System for Supporting Collaborative Work.*" *Int. Conf. on Organizational Computing Systems COOCS'91*,.1991; pp. 213-224.

[20] Warnecke, G.; Zeihsel, F. "Systematic Product Development using Information and Communication Technology.": *Production Engineering*, 1998, 5(1), pp. 107-110.

[21] Westfechtel, B. "Integrated Product and Process Managemet for Engineering Design Applications" *Integrated Computer-Aided Engineering*, 3(1), John Wiley & Sons, New York, 1996, pp. 20-35.

[22] Wewers, T.; Wargitsch, C. "For Dimensions of Interorganizational, Document-Oriented Workflow: A Case Study of the Approval of Hazardous-Waste Disposal" *Hawai'i Int. Conf. on System Science*, 1998.

[23] Workflow Management Coalition "Reference Model - The Workflow Reference Model." Document No. WfMC-TC-1003. 1995.