Supervising Remote Task Execution in Collaborative Workflow Environments

Michael von Riegen* and Sonja Zaplata

University of Hamburg, MIN Faculty, Department of Informatics,
VSIS, Vogt-Kölln-Straße 30, D-22527 Hamburg

Abstract. A key problem of collaborative workflows is the - sometimes questionable - assumption that remote tasks will be executed as agreed on before. In order to enforce compliance to given requirements, evidence of the correct execution of a remote task has to be produced. However, participants of inter-organizational workflows are autonomous and information about their private business processes is often (intentionally) unavailable to service consumers. This contribution identifies and discusses distinct levels and approaches to enforce the correct execution of tasks which are executed remotely. In addition to that, specific supervising mechanisms are presented, which are able to create evidence of correct execution while preserving the autonomy of participating business partners. Finally, the identified mechanisms are integrated into a flexible architecture to support monitoring and controlling of remote services.

1 Introduction

Collaborative workflows between organizations and cross-border processes are playing an important role in today's business processes, for example for eGovernment applications. To achieve collaboration in virtual enterprises or between two or more business partners, there are reasonable strategies like outsourcing of tasks, execution of remote activities or the entirely distributed processing of workflows [18]. Nevertheless the acceptance for using these opportunities is still weak: As of today, service providers deliver their services on their own behalf. Outsourcing of tasks always means to distribute the responsibility of task execution, and a task executed remotely probably leaves the provider's sphere of control. Therefore, an unreliable execution of tasks through external subcontractors does not only weaken the customer's trustfulness, but can as well have serious legal consequences, for example if a subcontractor delivers too late.

In majority, partners of business contracts want to be autonomous and heterogeneous. Heterogeneity problems can be solved by using established Web Service technologies and service oriented architectures. But, due to their autonomy and because their internal business functions are concealed behind well-defined interfaces, the participants can change or behave arbitrarily [10]. Technically,

* The work was sponsored by the EU IST-2004-026650 project "R4eGov"

paradigms like encapsulation and information hiding help to develop and integrate standard and easy-to-use functional access points, but on the other hand they also imply the behavior of a black box. The lack of insight into private processes of cooperating business partners declines confidence and causes doubts on reliability. Therefore, there is a need for mechanisms to ensure that an external service behaves just like the consumer has expected it to do, and that the results received are correct. Especially in an open environment, the enforcement of contracts and the proof of compliance to predefined requirements is a challenge. Therefore, this paper addresses the question how to enforce the execution of tasks by producing evidence that a remote task was executed as agreed on before.

Enforcement of remote task execution implies on the one hand judicial aspects, where in case of failures the aggrieved party can go to court and enforce its rights. To be able to do this, the partners have to prove the correct execution of the objected task, for example with an audit log. On the other hand, the step enforcement implies technical aspects where, for example, transactions or monitoring and controlling mechanisms can be used to create evidence during or directly after the execution of a task. Here, the goal is to detect failures or discrepancies during and immediately after the execution of a task in order to be able to react at once.

This contribution concentrates on technical aspects and therefore identifies different approaches to task enforcement and defines various levels of integrating monitoring and controlling capabilities into existing collaborative workflows. The actual decision which technology to adapt for a specific scenario depends significantly on the application, the kind of outsourced tasks, the involved business parties, as well as the sensitivity and the relevance of the execution. Therefore, this contribution introduces an approach for a *flexible* supervision platform which is able to negotiate appropriate enforceability mechanisms for various applications.

The remainder of this paper is organized as follows: Section 2 introduces a classification of possible solutions for step enforcement and creation of evidence on remote task execution. Existing approaches and techniques are reviewed and discussed in Section 3. Section 4 presents important requirements and prerequisites, and section 5 proposes a coarse architecture for an integrated approach of step supervision and enforcement. The paper concludes with a summary and an overview of future work.

2 Step Enforcement in Collaborative Workflows

Collaborative workflows are often realized with approaches based on loosely coupled components. One of these approaches is the *workflow view approach* which is inspired by the database view concept. The workflow view approach can be described as a distributed workflow running between organizations, where the organizations involved are able to refine their parts of the global workflow to build their private (sub-)workflows. This peer-to-peer like approach requires that the

private workflows of participating organizations will satisfy the public workflow as agreed on before [18]. To build a workflow based on the workflow view approach, the following steps can be performed: The organizations involved agree on a common public workflow, which serves as a contract between them. Each task of the public workflow is mapped to one of the participating organizations. This public part of the workflow is referred as their public view. Each organization can now create a new private workflow or adapt their existing workflows regarding the public view. In order to satisfy the overall correctness of the collaborative workflow, each organization may only specify a private workflow as a subclass of its public view.

While a private view implements its corresponding public view and hides company specific implementation details, the public view itself interconnects and interacts with the public views of the other organizations. Within the next section, the need for concepts to enable enforceability in collaborative workflows is discussed against the background of the workflow view approach.

2.1 Enforcement of Step Characteristics

A key problem within collaborative workflows is the enforcement of contracts and the proof of compliance to predefined requirements. The goal is to produce evidence for the correct execution of remote tasks, without reducing the autonomy of the business partners.

Normally, the notion of transactions can be used to keep the consistency within a process and to provide evidence that all partners sent a positive confirmation on their execution. But how can a business partner verify semantically whether a remote partner upheld the agreements or not? In this case, transactions are not strong enough to produce evidence, so the outcome of a task itself has to be analyzed in more details. In order to do this, functional as well as non-functional aspects of the remote task have to be supervised.

The supervision of the execution of a task can take place either during (*monitoring*) or after a task is being executed (*controlling*) [11]. During task execution, monitoring can include intermediary results, status information or quality of service aspects. This information can be used to make predictions about success or failure of the execution, for example about the compliance to time-constraints. After task execution, the given results can be verified against agreements that have been specified beforehand.

2.2 Example

To motivate the need for a flexible and adaptable enforcement and supervision mechanism, a short example from the domain of public procurement is presented:

> *A company offers business services, which can be requested by both private customers and public authorities. Because of legal obligations, the company has to provide detailed supervising mechanisms in order to collaborate with*

a public authority. For example, the public authority needs a confirmation for each task performed by the company and also a deeper insight to some of the company's private processes. On the other hand, when dealing with private customers, requirements for enforcement become more casual, whereas performance issues gain more importance in order to be competitive with other providers of similar services.

This example displays different degrees of evidence and data in order to supervise remote tasks. Therefore, the next section presents different levels of enforcement by displaying several solutions for each level.

2.3 Levels of Enforcement

Depending on the processes' sensitivity and the effort to be spent in order to check compliance, either a pessimistic or an optimistic enforcement strategy can be applied. In a pessimistic scenario, everything processed by the remote task should be controlled. This can be done for instance by checking input values against output values and comparing them to contractual invariants or even install a trusted computing environment on the remote site. In contrast, an optimistic controller assumes that the remote partner operates like expected and applies just a minimum of checking. Between these two extreme positions, more detailed levels of enforcement can be identified:

(Level 0) Confidence In this level no mechanisms of enforcement are provided. The outsourcing party trusts on the partner's capabilities and relies completely on his integrity. There is no review of results and no guarantees are expected. Runtime errors may remain unhandled and the overall process might fail because of failures in subsystems. This level of enforcement is at the most appropriate for already established business relationships, intra-enterprise collaborations or self-maintained subsystems.

(Level 1) Confirmation The successful execution of an outsourced task will be notified by sending a confirmation, which could be, for example, a digital signature or a certification document. Collecting confirmations is most valuable for tasks and subprocesses, which do not return result values to indicate that *anything* has happened to perform the task. Although within this level the remote runtime behavior cannot be monitored and therefore errors cannot be detected in advance, failures determined *later* can be associated with their origin and, for instance, legal steps can be taken to deal with non-compliance.

(Level 2) Controlling by Process Design In case the control flow returns to the calling entity or output data is available, time constraints can be checked or return values can be verified against specified conditions. This level makes use of existing control flow constructs to check return values, deadlines and conditions. It is therefore part of the process modeling to specify what should happen if an error occurs during execution or if the result values do not meet given requirements.

(Level 3) Agreements and Policies Additional agreements can be made in order to define QoS requirements, policies or functional constraints. Agreements are mostly made prior to execution and are often defined separately from the business logic. Therefore, requirements and service properties are negotiated and finally result in a contract which is binding for both parties. But the act of checking the compliance with a defined agreement is outside the scope of defining contracts and needs support of other mechanisms, like monitoring and controlling tools. Therefore, this level as a single solution is not sufficient to detect errors in time to be able to react accordingly, e.g. by exchanging service providers in case the compliance with functional or non-functional requirements is at risk.

(Level 4) Monitoring and Controlling This level makes use of advanced monitoring and controlling tools either from the side of the service consumer, the service provider, or a trusted third party. For instance, consumer-side management systems can be extended in order to check incoming results and monitor the conformity of remote execution with predefined agreements. Furthermore, the service consumer can exploit monitoring information made available by the service provider. This way, either the foreign service provider can be monitored at runtime or the performed task is controlled after its completion. Provider-side monitoring and controlling requires a deeper look into external processes and allows for a more detailed view on quality parameters, faults and intermediate results, while still keeping the autonomy of each business partner. However, this level cannot prevent intentional malicious behavior of business partners.

(Level 5) Dual Control This level requires that a task will be effectively processed by at least two individuals or software programs (*4-Eyes-Principle*). This could for example mean that the execution is performed in a two-way manner. Results can either be compared to assure correct execution or the execution can be observed in a very detailed way, e.g. the result of each single step is controlled. The problem with this procedure is that it might be unreasonable and expensive: In most cases, the verification of a task's execution is at least as complex as executing the task itself. Furthermore, it appears that, if the outsourcing partner has the capability to rework every single step of a task, he could have done the job all by himself and there is no need for outsourcing it. Another point is that some activities with an *exactly once* semantic (e.g. a bank transfer) cannot be performed several times to compare results or to prove that the desired effect has occurred.

(Level 6) Direct Control Direct Control means to supervise the conformity of a remote task at the location of its execution. Technically, this can be achieved by a software agent or an external (hardware) module to control every step performed by the partner, for example with a trusted computing platform. Level 6 is the strongest way of enforcing functional and non-functional requirements. However, the problem is that normally autonomous business partners do not allow external software to be run within their private business systems. This paper does not deal with direct control mechanisms, because their realization would reduce the partners' autonomy.

A combination of the mentioned levels of enforcement is possible to achieve stronger individual evidence. For example, agreements to assure non-functional properties may be combined with means of confirmation to create legal evidence for the functional part. Another alternative is to integrate the results of advanced monitoring tools (Level 4) with the business process itself (Level 2) in order to allow an alternative control flow in case of dis-compliance.

3 Related Work

Most of existing languages to model business processes, for example *BPEL4WS* [1], *WSCI* [3] or *XPDL* [15], offer feasible constructs to check return values, monitor deadlines or define scopes for error handling in order to solve (external) failures. These elements allow to model controlling mechanisms right from the stage of process design and to integrate an alternative control flow in dependence of monitoring results. Additionally, some approaches integrate externally defined monitoring rules into existing processes by specifying assertions and by checking pre- and post-conditions to detect violations of functional contracts [5] [6].

The issue of defining service level agreements (SLA) and policies to obtain contracts is addressed, for instance, in the research area of Web Services where a couple of specifications have already evolved, like *WSLA* [14], *WS-Agreement* [2] or *WS-Policy* [4]. But, in the mentioned approaches, the problem of checking the compliance to these agreements remains unaddressed. Therefore, initiatives like the *Cremona* framework [13] also deal with the monitoring of agreements. Cremona introduces a middleware layer to create agreements and to access the current state of an agreement at runtime. Lazovik, Aiello and Papazoglou [12] propose another monitoring framework, which expresses contracts via assertions and checks them with a monitoring algorithm.

The CrossFlow Project [9] introduced monitoring mechanisms and emphasized its high relevance for inter-organizational workflows. Here is the focus on service contract establishment and enactment. The enforcement of contracts and the supervision of required Quality of Service parameters are considered only marginally in this project. An interesting aspect is the possibility to include terms of monitoring into the service contract, but integration with existing monitoring tools is an open issue.

More related work for monitoring and controlling data can be found in the research area of service management. OASIS provides the service management standards MOWS [16] and MUWS [17], which also involve aspects of service monitoring by using Web Service protocols. Likewise, Hewlett Packard presented the WSMF [7] framework for service management. In general, these management facilities were designed to be applied to controllable systems, not to remote services of autonomous business partners.

As to see, there are already many adequate concepts and technologies to define and integrate control flow, agreements and assertions. The presented concepts are feasible to define contracts on how to behave and to decide how to react in case of dis-compliance. A relevant insufficiency is that these techniques

- although well suitable on a higher level - lack basic support to create evidence about what has *actually* happened. Summarized, the presented concepts rely on observing and analyzing the *externally visible behavior* of the communicating parties.

4 Supervising Remote Task Execution

In collaborative workflows it is very likely that participants are autonomous and by this, their services and internal processes are mostly heterogeneous and may be based upon proprietary technologies. But not only service implementations are heterogeneous, also monitoring tools and log files are different amongst business partners. While services can be accessed by means of standard interfaces like WSDL, there is a need for allowing access to management tools, irrespective of the technology used for internally accessing monitoring, controlling and logging data. Additionally, security mechanisms to protect monitoring and controlling interfaces from unauthorized access have to be provided.

4.1 Requirements

Considering the problem of enforcement and the creation of evidence for task execution, a number of important requirements can be identified. First, validation for the execution of a task is necessary to disallow repudiation about which party has performed a certain object of agreement. Besides this merely judicial aspect, mechanisms to detect errors in any stage of an external task's execution are required. Most importantly, possible dis-compliances have to be detected immediately, so that partners can react as soon as possible. Reacting on dis-compliances can mean to either restart tasks or the whole process, change parameters or to select other partners in order to execute the task on a different system.

Because collecting of monitoring data and its evaluation can have negative effects on the performance of a task (or even the performance of the whole process), it is important to allow to separate monitoring capabilities from the business logic of the service [5]. Therefore, the consumer should decide (ideally dynamically) whether to integrate monitoring and controlling functionality or not. Cooperating parties can also decide or determine the amount of supervision in dependence on the task's sensibility or a business partner's identity and select the appropriate level of enforcement accordingly.

Nevertheless, if for instance public authorities need deeper insight into the partner's internal processes and need to include higher levels of enforcement, they have to prove their authorization level during a negotiation phase where partners can agree on supervising methods (See also section 2.2).

4.2 Discussion of Life Cycle Aspects

The question, what kind of supervision data should be accessible is also related to the question, *when* this data should be accessed. There are three possibilities,

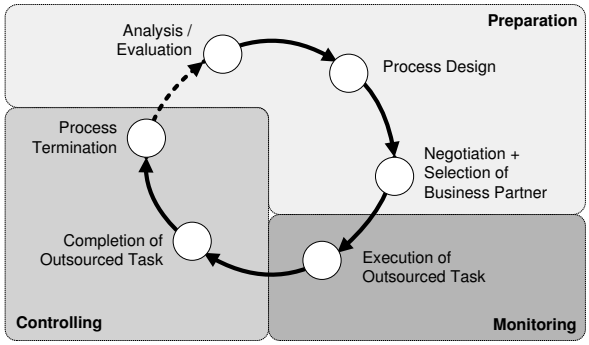

Fig. 1. Phases of Supervision

which are summarized in Figure 1: First, information can be accessed before the task is going to be executed (*Preparation Phase*). This could be, for example, within the preparation of service assembly to check the availability of a service. Second, information can be accessed at runtime (*Monitoring Phase*) in order to check the task's progress, its current state and the compliance to constraints and agreements. Third, results can be checked after the execution of a task (*Controlling Phase*). Here again, non-functional criteria like the abidance of deadlines, agreements and policies have to be evaluated. It is also possible to delay the controlling procedure and to wait until the entire process is finished. The results of this *late* controlling procedure do not longer affect the process instance itself, but can be integrated into statistics, interpreted for optimization purposes, or can be taken to initiate legal actions against partners who did not fulfill their tasks properly. This is often referred to as auditing which analyzes divergences between the as-is state of a system and the target state in which it should be or should have been.

4.3 Supervision Models

Besides the aspects regarding the life cycle, monitoring and controlling techniques can be distinguished by the way of how the supervision information is actually accessed: In the *Push Model* monitoring and controlling data is supplied by the service provider, for example by sending notifications in case a certain event occurs or a threshold value is exceeded. In the *Pull Model* the data is explicitly requested by the consumer, e.g. by asking for the current process state [8]. Both paradigms can be combined to design flexible and efficient monitoring and controlling models.

5 An Architecture to Supervise Remote Execution

Generally, monitoring and controlling can be considered from three different view points, depending on the involved party: A *service provider*, a *service requester* and an eventually involved *third party*.

A service provider supplies relevant data and (aggregated) information about the progress of a task via monitoring and controlling interfaces on the public view of a workflow. The provided interfaces encapsulate relevant internal data provided by the according private workflow and are able to negotiate the degree of information that can be supplied. They are also able to return results (if any) to a service requester, depending on the type of interaction between the business partners.

On the other side, a service requester can check results and information supplied by the service provider. This includes monitoring and controlling data and even the results of a request itself. Regarding the provided data, the requester can decide if this creates evidence on task execution or if the provider abides certain quality of service requirements.

Nevertheless, both service provider and service requester have the option to use a trusted *third party* in order to collaborate. This trusted third party monitors the execution, validates the results and ensures the abidance of given contracts. Apart from that, a third party has also the ability to monitor messages and to check whether the exchanged messages between business partners comply a given protocol or not.

Regarding these viewpoints, the next section gives an overview of our architecture to supervise remote task execution.

5.1 Overview

Figure 2 depicts the viewpoints as introduced in section 5 and shows starting-points for integrating monitoring and controlling capabilities into existing collaborative workflows: If a client invokes a method on the public view of business partner A, the execution of a private process is initiated. If this private process of business partner A includes interaction with other services regarding the public workflow, A will invoke a method of the public view of business partner B. Because A is in charge of the whole request and the integration of partner B remains transparent for customers, the enforcement of this step (executed remotely by B) is essential for A.

The key component of B is an additional service which offers an interface to negotiate enforcement mechanisms and provides monitoring and controlling capabilities. This *monitor and controlling interface* (m+c interface) encapsulates internal process data of the private view and possibly existing private monitoring mechanisms. It aggregates required data in order to abide privacy policies. In case an internal system monitor already exists, the m+c interface can act as a simple wrapper to integrate the monitoring functions. If there is no existing internal system monitor, the m+c interface has to monitor internal processes as well.

The proposed levels of enforcement (see section 2.3) can also be identified in this architecture. Level 0 (Confidence) does not use monitoring and controlling mechanisms at all. This is the trivial case where no m+c interfaces and system monitors are implemented. Within this architecture, Level 1 (Confirmation) is

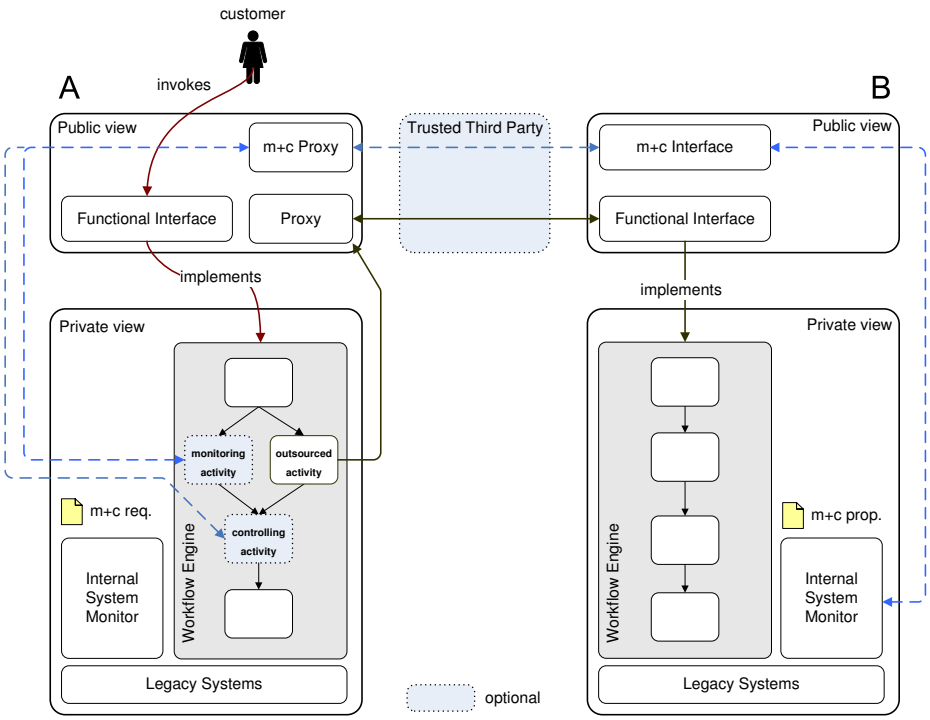

Fig. 2. Coarse architecture to enforce remote task execution

achieved by requesting a digital signature attached to the task's result after finishing the remote execution. Furthermore, activities to check the result values or time constraints are optionally included into the process description (Level 2, Controlling by Process Design). Existing concepts are used to realize this, e.g., the conditional branch construct or the deadline expression from BPEL4WS. Advanced requirements and factors which cannot be specified within a process description or which should be defined external to the business logic, are described by additional documents containing agreements and policies (Level 3, Agreements and Policies). For instance, quality of service parameters, like constraints on costs and response time or requirements on security issues can be defined. To access runtime data, like progress information or intermediary results, the selected business partner has to offer a special interface to present the results of internal monitoring and controlling tools to the consumer (Level 4, Monitoring and Controlling). The correctness of received results or the verification of the consequences of an execution can further be determined by another authority (Level 5, Dual Control). This authority could either be the consumer of the service himself or a trusted third party. Level 6 (Direct Control) is not applicable within this architecture, because it would reduce the autonomy of business partners.

5.2 Negotiation of Supervising Policies

Like described within the requirements (see section 4.1), the type of supervising mechanisms and the amount of insight provided must be negotiable. Besides the required level of enforcement (see section 2.3), formats and patterns (e.g. push vs. pull model) have to be settled before starting collaboration. In summary, there are five different areas of negotiable information: Status and progress information, Quality of Service data, intermediary results, certification data and log data. The requirements for particular monitoring information are specified within a *m+c requirements document*, whereas the properties of the remote m+c interface are defined in a *m+c property document*. Matching between these two policy documents can be included into the service discovery and selection procedure, so the m+c requirements can act as a non-functional requirement for service selection. To attach these requirements documents to specific business cases, they are either involved as process data in the workflow description itself or can be specified externally in a Service Level Agreement (SLA).

6 Conclusion

This paper has motivated the need for techniques to support monitoring and controlling in order to create evidence of a remote task execution within collaborative workflows. In contrast to auditing mechanisms, this paper proposes supervising techniques that are able to check, validate and control the execution of a task immediately after and during its execution without tampering the remote site. We have presented different supervising categories that can be used as single solutions or as a combination to strengthen the evidence of execution. These supervising mechanisms to enforce a step within a collaborative workflow have been depicted in different phases of a process life cycle. In order to handle these categories, we have presented an architecture for supervising task execution by using the public-to-private workflow model to enable collaborative workflows between two partners. The introduced architecture is based on Web Service technologies and provides negotiation capabilities to determine the category of supervising by the involved partners prior to task execution. In the future, the amount of supervision is supposed to become dynamically adjustable, so that a possibly negative performance impact of supervision can be evaluated against the benefit of the enforced task execution.

Finally, this contribution only considers the creation of evidence of remote tasks executed between *two* partners (*local evidence*). Future work includes the extension of the concept to *all partners* participating in the collaborative workflow. To prove that all tasks of such a distributed workflow have been executed accordingly to agreements made before (*global evidence*), local guarantees have to be integrated. This also reflects the idea taken from transaction management, where local guarantees generate global guarantees on distributed transactions in heterogeneous federations [19].

References

1. T. Andrews et al. Business Process Execution Language for Web Services - Version 1.1. Technical report, BEA Systems, International Business Machines Corporation, Microsoft Corporation, SAP AG, Siebel Systems, May 2003.
2. A. Andrieux et al. Web Services Agreement Specification (WS-Agreement). Specification, 2005.
3. A. Arkin et al. Web Service Choreography Interface (WSCI) 1.0. Specification, World Wide Web Consortium, 2002.
4. S. Bajaj et al. Web Services Policy Framework (WSPolicy). Specification, 2006.
5. L. Baresi, C. Ghezzi, and S. Guinea. Smart monitors for composed services. pages 193–202, 2004.
6. L. Baresi and S. Guinea. Towards Dynamic Monitoring of WS-BPEL Processes. In B. Benatallah, F. Casati, and P. Traverso, editors, *ICSOC 2005, Third International Conference of Service-Oriented Computing*, volume 3826 of *Lecture Notes in Computer Science*. Springer, 2005.
7. N. Catania, P. Kumar, et al. Web Services Management Framework (WSMF) Overview Version 2.0. Specification, 2003.
8. Y. Hoffner, H. Ludwig, et al. An Architecture for Cross-Organizational Business Processes. *Proceedings of the Second International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2000)*, 00, 2000.
9. Y. Hoffner, H. Ludwig, et al. CrossFlow: integrating workflow management and electronic commerce. *SIGecom Exch.*, 2(1), 2001.
10. M. N. Huhns and M. P. Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, 09(1), 2005.
11. S. Jablonski, M. Böhm, and W. Schulze. *Workflow Management - Entwicklung von Anwendungen und Systemen*. dpunkt, 1997.
12. A. Lazovik, M. Aiello, and M. Papazoglou. Associating assertions with business processes and monitoring their execution. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, New York, NY, USA, 2004. ACM Press.
13. H. Ludwig, A. Dan, and R. Kearney. Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, New York, NY, USA, 2004. ACM Press.
14. H. Ludwig, A. Keller, et al. Web Service Level Agreement (WSLA) Language Specification. Specification, 2003.
15. R. Norin, M. Marin, and R. Shapiro. Workflow Process Definition Interface - XML Process Definition Language Version 2.0. Specification WFMC-TC-1025, Workflow Management Coalition, 2005.
16. I. Sedukhin. Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.0. Specification, 2005.
17. W. Vambenepe. Web Services Distributed Management: Management Using Web Services (WSDM-MUWS) 1.0. Specification, 2005.
18. W. van der Aalst. Loosely coupled interorganizational workflows: modeling and analyzing workflows crossing organizational boundaries. *Inf. Manage.*, 37(2), 2000.
19. G. Weikum and G. Vossen. *Transactional Information Systems*. Morgan Kaufmann Publishers, 2002.