GMD Report 30

GMD –
Forschungszentrum
Informationstechnik
GmbH

Justus Klingemann, Jürgen Wäsch,
Karl Aberer

Adaptive Outsourcing in Cross-Organizational Workflows

August 1998

© GMD 1998

GMD –
Forschungszentrum Informationstechnik GmbH
Schloß Birlinghoven
D-53754 Sankt Augustin
Germany
Telefon +49 -2241 -14 -0
Telefax +49 -2241 -14 -2618
http://www.gmd.de

In der Reihe GMD Report werden Forschungs- und Entwicklungsergebnisse aus der GMD zum wissenschaftlichen, nichtkommerziellen Gebrauch veröffentlicht. Jegliche Inhaltsänderung des Dokuments sowie die entgeltliche Weitergabe sind verboten.

The purpose of the GMD Report is the dissemination of research work for scientific non-commercial use. The commercial distribution of this document is prohibited, as is any modification of its content.

Anschrift der Verfasser/Address of the authors:
Justus Klingemann
Jürgen Wäsch
Dr. Karl Aberer
IInstitut für Integrierte Publiukations- und Informationssysteme
GMD – Forschungszentrum Informationstechnik GmbH
Dolivostraße 15
D-64293 Darmstadt
E-mail: {klingem,waesch,aberer}@darmstadt.gmd.de

ISSN 1435-2702

Abstract

Competitive markets force companies to minimize their costs while at the same time offering solutions which are tailored to the needs of their customers. This requires organizations to outsource at least parts of the overall effort and to rely on services which have to be provided by external partners, leading ultimately to virtual enterprises.

The workflow concept has been very successful in coordinating and streamlining business processes but is so far limited to a single organization. Any attempt to distribute workflows among different organizations has to face the problems posed by the complex relationship among autonomous organizations and their services. To address these problems we propose a service-oriented model for cross-organizational workflows. Modeling the workflow execution as a co-operation of services allows different organizations to interact via well-defined interfaces. We further show how the execution can be optimized by selecting services depending on their contribution to quality criteria of the workflow.

Keywords: Workflow, Virtual Enterprise, Outsourcing, Optimal Service Selection

1 Introduction

Competitive markets force companies to minimize their costs while at the same time offering solutions which are tailored to the needs of their customers. This requires organizations to outsource at least parts of the overall effort and to rely on services which have to be provided by external partners, leading ultimately to virtual enterprises. Hence, business links with other organizations have to be set up and managed. This has to be achieved fast and flexible to guarantee a short time to market while allowing a dynamic reaction to new customer demands or changing offers of service providers in electronic commerce environments.

Information technology has provided different technologies to address these requirements. The workflow concept [GHS95, AAAM96, Moh98, DÖBS98, CHRW98] has been very successful in coordinating and streamlining business processes but is so far limited to a single organization. On the other, hand the tremendous growth of global networks like the internet provide the possibility to efficiently exchange data and communicate with a large number of possible service providers. The extension of workflow management systems (WfMS) to manage cross-organizational business processes therefore seems to be a promising solution.

However, the extension of workflows beyond the borders of a single enterprise raises new challenges. Usually different possibility exist to achieve the customer requirements. This implies that a decision has to be made which choice is most appropriate for a specific workflow instance. Especially, which activities or group of activities should be outsourced to which business partner. Possible criteria are the required time, cost, or the adherence to certain other quality of service parameters.

The autonomy of the participating organizations implies that the initiator of the workflow has only limited control over the outsourced activities. This requires that both sides agree on an interface which allows the service requester to monitor and probably control the outsourced activities to a certain extend.

To address these problems we propose a service-oriented model for cross-organizational workflows. Modeling the workflow execution as a cooperation of services allows different organizations to interact via well-defined interfaces. We further show how the execution can be optimized by selecting services depending on their contribution to quality criteria of the workflow.

The work presented in this paper will contribute to the ESPRIT-IV project CROSSFLOW [CC97, CC98] starting in the third quarter of 1998[1]. The overall goal of CROSSFLOW is to develop and implement the mechanisms for connecting WfMS and other WfMS-like systems of different organizations in cross-organizational workflows and electronic commerce settings. In particular, CROSSFLOW is developing cooperative support facilities that allow the operational management of cross-organization workflows. These include:

- A *service-oriented model for cross-organizational workflows*. This includes a basic architecture, an abstract workflow model, a service model, and an execution model.
- *Adaptive outsourcing and flexible change control* in cross-organization workflows: flexible mechanisms for the selection and invocation of services in cross-organizational workflows.

[1]Members of the CROSSFLOW consortium are AGF Irish Life Holdings (Ireland), GMD-IPSI (Germany), IBM-Germany, IBM-France, IBM Zurich Research Laboratories (Switzerland), KPN Research (The Netherlands), Sema Group sae (Spain), and the University of Twente (The Netherlands).

- *Monitoring* of the outsourced parts in cross-organizational workflows: to provide information about service progress, resources consumed and the achieved quality of service.
- *Level of Control:* to allow the service requester some control over outsourced parts of cross-organizational workflows that are executed by an external service provider, e.g., to react on quality of service defects.

In this paper, we concentrate on the first two issues. First, we describe our view on cross-organizational workflows. In section 3, we explain how workflows and services are modeled and can interact. After that, we describe how the assignment of activities to services is adapted to the available service offers. We discuss different alternatives when to outsource activities and present appropriate algorithms. This allows the optimized execution of a cross-organizational workflow according to the specified quality of service parameters. After discussing related work in section 5, we summarize our results and discuss future work.

2 Cross-Organizational Workflows

Workflow management is a fast evolving technology which is increasingly being exploited by businesses in a variety of industry [GHS95, SGJ+96, Moh98, CHRW98]. Its primary mission is to handle business processes. A *business process* is a set of one ore more linked activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships [WfM96c]. A *workflow* is the automation of a business process, in whole or part, during which documents, information, or activities are passed from one participant to another, according to a set of predefined rules [WfM96c]. A *workflow management system* (WfMS) defines, creates, and manages the execution of workflows.

We call a business process *cross-organizational* if there is the possibility that at least one of its activities is outsourced to a different organization. An example is a motor claim process in the insurance industry. Usually, in this process several organizations are involved besides the insurance company. For example, the initial notification and collation of accident details may be outsourced to an accident management company. The examination of the damaged car vehicle is done by an independent engineer. In a liability situation, the outsourced parts of the workflow may also include medical examinations, a solicitor, a consulting engineer, and legal counsel [CC98].

Workflow management technology can also be used for cooperation and coordination of the work in cross-organizational business processes. A *cross-organizational workflow* is the implementation of a business process that crosses organizational boundaries.

The realization of cross-organizational workflows requires some kind of interoperability among the various WfMS and other computer systems run by the different organizations [GHS95, SGJ+96, Geo98, MWW98a]. Any attempt to connect the WfMS of two organizations will run into multiple problems posed by the complex relationship between autonomous organizations, their authorities, administrations, and information systems. Systems that reside in different organizations belong to different administrative, technological, and security domains. If the systems are to be interlinked and are to cooperate, any difference has to be taken into account and dealt with. A common approach is to insert appropriate *gateways* at the organizational boundaries, thereby ensuring that security related information is added or removed where necessary, that

appropriate security checks are performed, monitoring and logging can take place, differences in management procedures are translated, and that measuring, costing and accounting are dealt with, etc. [Hof96, HC97, CC98].

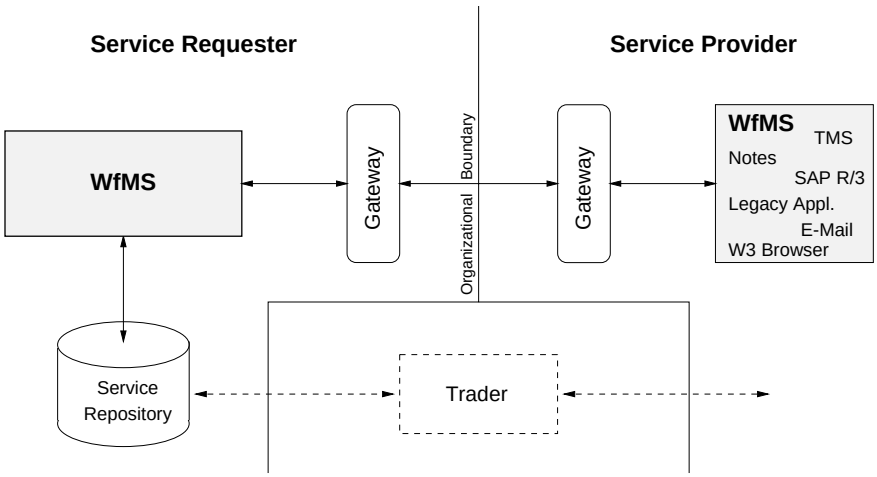

Figure 1: A (simplified) architecture for cross-organizational workflows.

A central concept in our notion of cross-organizational workflows is a *service*. A service comprises any action done by one party, i.e., the *service provider* on behalf of another party, i.e., the *service requester*. The service requester may use the services offered by various external service providers to outsource parts of a cross-organizational workflow.

Figure 1 shows a (simplified) architecture of a system for realizing cross-organizational workflows. In this paper, we abstract from the possible instantiations of this architecture. The WfMS of the service requester contains the cross-organizational workflow description and runs the cross-organizational workflow. A service repository contains the description of all available services offered by service providers. The service providers may run any kind of "WfMS-like system". This can range from a full-fledged WfMS over task management systems, standard business software, legacy applications to even simple WWW browsers or E-mail clients, hence, enabling the inclusion of small organizations in cross-organizational workflow implementations. The gateways are used to connect the WfMS of the service requester and the system used by the service provider, to homogenize the differences, and to add functionality to the WfMS or other systems when required. Optionally, there might be a (third-party) trader system which fills up the service repository. Note that the trading of services is outside the scope of this paper. For the sake of simplicity, we assume that all the available services are contained in the service repository together with *service level agreements*. A service level agreement is a "contract" among the different organizations about service provisioning and contains terms of responsibility, accountability and liability, etc. [HC97, CC98].

3 A Service-Oriented Model for Cross-Organizational Workflows

3.1 Specifying Workflows

In this section, we describe our conceptual model for cross-organizational workflows. The model we present is influenced by the FLOWMARK meta model [LA94] since FLOWMARK will be the primary platform in the CROSSFLOW project. However, we confine ourselves to the essential concepts since the ideas and algorithms presented in this paper are not restricted to a specific WfMS. For our purposes, a *workflow* W consists of the following constituents described below.

- formal *input* parameters $input(W)$ and *output* parameters $output(W)$,
- a set of *constituent activities* $A_W = \{A_1, \ldots, A_n\}$,
- a directed graph $CF_W = (A_W, C_W)$, $C_W \subseteq A_W \times A_W$ (with transition conditions p_c assigned to the edges in the graph) that describes the *control flow* in the workflow,
- a directed graph $DF_W = (A_W, D_W)$, $D_W \subseteq A_W \times A_W$ that describes the *data flow* in the workflow, and
- additionally a list of *quality of service* parameters $QoS(W)$.

Constituent activities. Activities $A_i \in A_W$ are described by formal input and output parameters $input(A_i)$, $output(A_i)$. Activities can be either *elementary* or *complex* activities. Activities can be considered as relations on $input(A_i) \times output(A_i)$. Complex activities are further described by another workflow W_{A_i}. Hence, the internal functioning of a complex activity is known whereas elementary activities are viewed as "black boxes" at the workflow level. By allowing complex activities it is possible to compose a workflow in a hierarchical way. Additionally, activities have certain QoS parameter assigned. Some of them are mandatory while others should be fulfilled as good as possible.

Each activity has a state $\in \{not\,ready, ready, active, completed, dead\}$. Initially, an activity is in the state $not\,ready$. The activity will become $ready$ if the control flow dependencies enable its execution (see below). An activity which is currently running is $active$. When it is finished it takes the state $completed$. An activity is $dead$, when it will never be executed (see below).

Control flow graph. Each edge $c = (A, B) \in C_W$ in the control flow graph CF_W is associated with a boolean predicate p_c that determines the activation of activity B when A terminates. For the sake of simplicity, we assume in this paper that the control flow graph CF_W is acyclic.

To determine the value of the predicate $p_{(A,B)}$ of a control flow edge (A, B) at a certain time t we define the *evaluation function* $E(p_{(A,B)}, t)$. The range of E is $\{true, false, undefined\}$. For each activity $A \in A_W$ we define the set of all preceding activities as $Pred(A) := \{B \in A_W \mid (B, A) \in C_W\}$. This allows us to formally define when an activity is ready or dead: An activity $A \in A_W$ is $ready$ at time t if $A \in Ready(W, t)$ which is defined as follows:

$$\begin{aligned} Ready(W,t) \;:=\; & \Big\{ A \in A_W \mid \Big(\exists B \in Pred(A) : B\,completed \wedge E(p_{(B,A)}, t) = true\Big) \wedge \\ & \Big(\forall B \in Pred(A) : B\,dead \vee \Big(B\,completed \wedge E(p_{(B,A)}, t) = true\Big)\Big) \Big\} \end{aligned}$$

An activity $A \in A_W$ is *dead* at time t if $A \in Dead(W,t)$ which is defined as follows:

$$Dead(W,t) := \Big\{ A \in A_W \mid \Big(\exists B \in Pred(A) : B\, completed \wedge E(p_{(B,A)},t) = false\Big) \vee (\forall B \in Pred(A) : B\, dead) \Big\}$$

Data flow graph. Each edge $d = (A,B) \in D_W$ in the dataflow graph DF_W is associated with a partial mapping $f_c : dom(output(A)) \rightarrow dom(input(B))$. The mapping f_c specifies which output parameter of activity A is mapped to which input parameter of activity B.

Of course, some structural constraints apply to the control flow graph CF_W, to the data flow graph DF_W, and their combination to ensure the correctness of the workflow. For example, for each $d = (A,B) \in D_W$ there must exist a path in the control flow graph from A to B. We do not elaborate this issue further since validation of workflow types is out of the scope of this paper. We assume in the following, that the workflow type has a meaningful correspondence to a business process and is validated with respect to some criteria specific by an organization.

Quality of service parameters. We assume that it is possible to specify certain quality of service parameters (QoS) for a workflow. It is possible to "overwrite" QoS parameter values of a workflow type by supplying more specific QoS parameter values at invocation time. Examples are the maximal duration and the maximal cost allowed for a workflow, and the services usable in the workflow (see below). Despite these more or less application-independent QoS parameters, domain-specific QoS parameter can exist.

3.2 Modeling Services

In order to allow the execution of activities at runtime, we need to define a mechanism that assigns activities to particular "agents" that are responsible for the execution of the activity instances. Usually, in intra-organizational workflows agents are considered to be human beings or computer programs. If a workflow is allowed to span different organizations, there is a third kind of agents that can be involved in the execution of a cross-organizational workflow instance, namely *external service providers.*

In commercial workflow systems like FLOWMARK, there is a ***fixed*** assignment of activities within a workflow to a description of the agents that can execute the activity. This can be done for example by means of organizational units, roles, or users. All agents that fit the description can potentially execute the activity. In most systems, all these agents are notified about the activity that is ready to run. The first agent that responds is then responsible for the execution of the activity. Of course, more advanced resource resolution schemes can be used, e.g., considering the current workload and availability of an agent [WMW98].

We introduce a more flexible *service-oriented model*. We believe that this model is better suited for cross-organizational workflows while at the same time being applicable in ordinary enterprise-wide workflows. The basic entity in our model is a *service*. Informally, a service in an abstract specification of the amount of work that a resource promises to carry out with a specific quality of service. A service specifies which part of a workflow it fulfills. In general, a service is not restricted to execute a single activity, it can span multiple activities. With each service there is a *service provider* associated. This can be either an internal resource or an *external organization*.

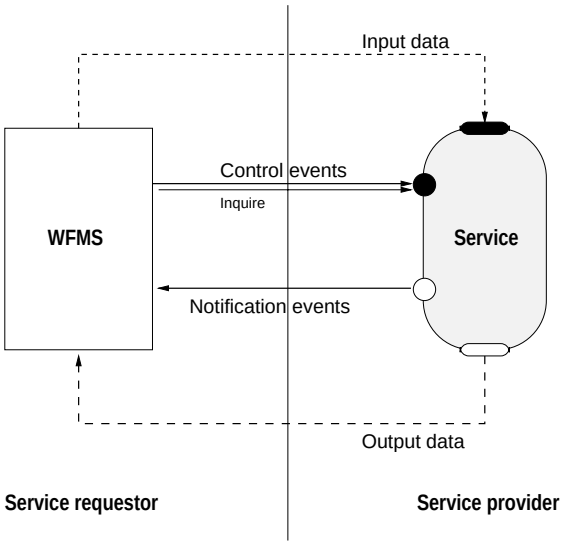

Figure 2: Interfaces of a service and interaction with the WfMS.

A service offered by an external organization is called an *external service*. Otherwise, it is called *internal service*.

In contrast to internal services, external services are not executed under the control of the service requester, i.e., the organization that runs and controls the cross-organizational workflow. For example, it might not be possible for a service requester to inquire the state of an external service execution. Moreover, the internal work process of an external service might not be known to the service requester due to the autonomy of the service providers. Thus, while they are executed, services have to be treated as "black boxes" from the viewpoint of a service requester. Only the interfaces to and from the services are known by the service requester and which activities are subsumed by the service. This includes a specification how an external service can be invoked, which parameters have to be supplied, etc.

Besides the interfaces, a service description contains the quality of service and the level of control offered by the service provider to the service requester. Within a level of control specification we can distinguish two different parts. One is concerned with the *monitoring* the execution of services, i.e., the degree of information flow from the service provider to the service requester. The other is concerned with the *control* of the services, i.e., the possibilities a service requester has to influence the execution of the external service by the service provider. Examples are cancelling, interrupting, or speeding up external services.

Additionally, a service description specifies a set of quality of service parameters that a service provider promises for an external service. Examples are the maximum costs and duration for the execution of a service instance. Again, there might be a variety of other, domain-specific QoS parameters.

Figure 2 illustrates our view on a service. A service has four interfaces. Two of them are concerned with the data flow and two of them are concerned with control flow.

In this paper, we define services based on a specific workflow. The service provider specifies which part of the workflow he offers to execute. This is done by identifying the activities

comprised by the service. Implicitly, this defines a subworkflow with the control and data flow dependencies of the corresponding part of the workflow (see also Figure 3). Another possibility to specify services would be to specify a service as a complete workflow of its own, including for example explicit control and data flow dependencies. To avoid the duplication of specifications we have chosen the former possibility but we are well aware that a service can be applicable in different workflows.

The specification of a service describes only the visible behavior of the service, i.e., the *external view* of the service. However, since the service requester has only a limited possibility to supervise the service execution as specified in the service level agreement, the service provider is free to execute internally a different subworkflow as long as the results are equivalent at the externally visible interfaces. A *service specification* S (for a workflow W) has the following constituents:

- The set $A(S) \subseteq A_W$ of activities describing the part of W that is "implemented" by the service.
- $Resource(S)$ denotes the service provider that is responsible for the execution of the service.
- $QoS(S)$ describes the quality of service parameters offered by the service.
- $Input(S)$ describes the input values of the service. When a service is invoked by a service requester $input(S)$ contains the data values that are needed for the execution of a service. Note that it might not necessary that all the input values are immediately available when the service is invoked. $Input(S)$ is specified by means of the data flow edges.
- $Output(S)$ contains the data values that are the result of the service execution. Note that it might be the case that some result values are available before the service execution terminates. $Output(S)$ is specified by means of the data flow edges.
- $Control(S)$ defines the *control events* (or control messages) that can be sent by the service requester to the service provider in order to influence the processing of the service. Of course, control messages may contain parameters.
- $Notification(S)$ defines the *notification events* (or notification messages) that can be sent by the provider in order to inform the service requester about the state of the processing of the external service. These events correspond to the observable "states" [RKT$^+$95, RS95] of the service. We assume that at least a termination event is available that is sent to the service requester when the service execution is finished. Like control events, notification messages might include data values, e.g., data necessary for the service requester to evaluate the predicate assigned to a control flow edges in a workflow W. Note that it is not required that the service provider actively sends the notification events. The service requester can inquire about the current state of the processing (as shown in Figure 2). Thus, the service provider then sends notification events "on demand" of the service requester.

Figure 3 illustrates the service specification. The shaded part of the workflow is offered by the service. Rectangles denote activities. Control flow edges are represented by solid lines and data flow edges by dashed lines. The interaction points between the service and the rest of the

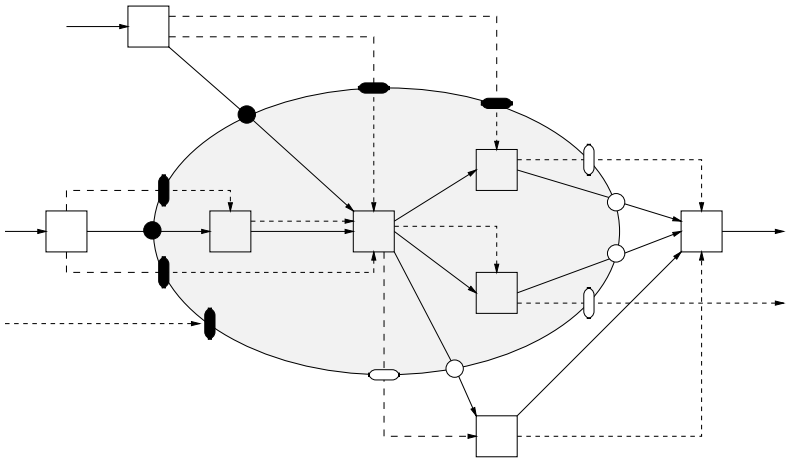

Figure 3: Service and its interfaces to a workflow.

workflow are depicted by circles and rounded boxes. Circles denote event channels and rounded boxes data interfaces. Interaction points in black denote interactions that go into the service and those in white denote interactions that go out of the service.

3.3 Execution model

In this section, we briefly describe the execution model we assume for cross-organizational workflows. The execution of the workflow follows the control flow graph CF_W. When the workflow is started, all activities that have no predecessor get into the state $ready$. An activity that is in the state $ready$ can be executed. The start of the execution changes the state to $active$. When the execution terminates, the state changes to $completed$. This triggers the evaluation of predicates at outgoing edges and thus can result in state transitions of succeeding activities.

In traditional approaches, for each activity that is executable a user (or computer program) has to be identified that is the responsible for the execution. This is done by evaluating the resource the activity is assigned to. If one agent is identified, the activity instance is started. In our model, this is more complex since services can cover more than one activity and there might be different services available with intersecting activity sets. Therefore, we introduce different "phases" in the execution:

- Service *selection*: The system selects a set of services covering one or more activities. There are different strategies when an activity is ready to be assigned to a service. These strategies together with algorithms to perform the service selection are described in section 4.

- Service *invocation*: A selected service is invoked. A $startService(S)$ control message is sent together with (some) data of $input(S)$ to the service provider.

- Service *execution monitoring*: During a service execution the WfMS listens to notification events to monitor the progress of the service execution. This includes events that signal the completion of certain activities or the availability of parts of the output data. If a

notification is sent that corresponds to a control flow edge pointing to an activity outside the service, the WfMS system evaluates the associated predicate. We assume that the data necessary for the evaluation of the predicate is made available as part of the control event. Moreover, it is assumed that other parameter values are set in $output(S)$ accordingly. Other notifications might indicate deviations from the QoS parameters. Appropriate action will then be undertaken by the WfMS considering the available level of control for the service.

- Service *termination*: The service notifies the WfMS about its termination. All the data in $output(S)$ is then available. If a service S terminates, all the activity instances in $A(S)$ are marked as $completed$ if no other message concerning this activity has been received.

4 Selection of Services in Cross-Organizational Workflows

In this section, we will discuss how services will be selected to execute the activities of the workflow. We assume that all relevant information about services is made available in the service repository. Therefore, we do not consider issues like finding out which services are available.

The workflow and its activities have certain QoS parameters. Some of them are *mandatory*, others should be fulfilled as good as possible and are optimized with the following algorithms. For the sake of simplicity we assume that the latter parameters can be aggregated into a single value which can be used to compare different service offers, i.e., each service S has assigned a value q which we call the *price* of the service. We further assume that this price can be calculated for all services offered.

Since the edges of the control flow graph are labeled with conditions, it is usually not clear at the start of a workflow which activities actually have to be executed. For example, a workflow can contain branches with alternative sequences of activities for certain parts of the workflow. Conditions can be evaluated when all necessary parameters are available due to the execution of previous activities. Service selection mechanisms can be differentiated depending on when activities are assigned to services. In the first part of this section, we will consider the case where we only assign those activities to services from which we know in the respective state of the workflow that they have to be executed. In the second part of this section, we extend this mechanism to more complex subworkflows which contain decisions yet to be made on which parts of the subworkflow have to be executed.

4.1 Selection of Services for a Fixed Set of Activities

To outsource activities, parts of the workflow, or even the whole workflow we have to determine which part of the workflow is ready to be outsourced. In this subsection we assume that we do not want to outsource control flow decisions. Therefore, the part of the workflow which can be assigned to services at a certain stage of the workflow execution is restricted to those activities, which are known to be executed. We therefore define the set of activities which can be outsourced at time t as those which are $ready$ or will reach this state as the necessary predicates already evaluate to $true$:

$$
\begin{aligned}
Outsourcable(W,t) \;:=\; & \{A \in A_W \mid A \in Ready(W,t) \vee \\
& (A \in Not\,Ready(W,t) \wedge (\forall B \in Pred(A) : B\,dead \vee \\
& B \in Outsourcable(W,t) \wedge E(p_{(B,A)},t) = true))\}
\end{aligned}
$$

In the following, we describe an approach how to select services for the activities in $Outsourcable(W,t)$. Similar techniques can be applied if the assignment of services for some activities should be postponed, i.e., we select services only for a subset of activities in $Outsourcable(W,t)$.

Let Σ be the set of services which offer to cover an appropriate part of the workflow in line with the specification of the workflow. In particular, these services fulfill all mandatory QoS parameters and execute the activities in an execution order which is in line with the workflow specification[2]. Additional restrictions can be included, e.g., the service requester might want to exclude the services of a certain service provider. The problem can then be formulated as finding a set of services from Σ which covers all activities in $Outsourcable(W,t)$ exactly once while optimizing the non-mandatory QoS parameters. Mathematically, this problem is a minimal weighted exact set cover, i.e., can be formalized as follows: Find a subset O of Σ, that forms an exact cover, i.e., $\bigcup_{S \in O} A(S) = Outsourcable(W,t)$ and $\forall S_i, S_j \in O : A(S_i) \cap A(S_j) = \emptyset$ and such that the *cost* of the cover $\sum_{S_i \in O} q_i$ is minimized. This problem is NP-complete [GJ79, Kar72, Hoc97]. However, since the set activities to be outsourced is usually small, we expect that our approach is applicable to many real-world workflows.

4.1.1 Solving the Minimal Weighted Exact Set Cover

In the following we describe the basic algorithm how an optimal choice of services can be selected.

1. Generate for each activity A_i in $Outsourcable(W,t)$ a set $Serv_i$ containing all services which encompass activity A_i, i.e., $S \in Serv_i$ **iff** $A_i \in A(S)$. Define $SERV := \{Serv_i\}$. Note, that the sets $Serv_i$ need not to be disjoint as a service can cover multiple activities. A correct assignment of services consists of a selection of one service out of each set $Serv_i$ whereas all services have to cover disjoint sets of activities.

2. Sort the sets in $SERV$ in ascending cardinality resulting in the list $Slist$. Sorting the sets of services is a heuristic to improve the efficiency of the algorithm. The choice of one service excludes other services with a non-disjoint set of activities from the selection. By starting with activities that are covered by few services (in the extreme case only one) we can quickly narrow the search space. Various other tuning possibilities exist, e.g., using only the best service S out of all services that cover the same set of activities $A(S)$, but we omit them here to keep the description of the algorithm simple.

3. call $SelectServices(Slist, 0, \emptyset)$

[2]Note, that this does not mean that the actual execution order has to be the same as in the workflow specification. Due to the autonomy of the service provider and the resulting limited monotoring capabilities we only require that the execution order cannot be distinguished at the interface of the service.

Function 1 Select an optimal assignment of services

$\boldsymbol{SelectServices(Slist, Cost, Cover) : [Cost, Cover]}$
if $Slist$ is the empty list **then**
  return $[Cost, Cover]$
end if
remove the first set $Serv_j$ of $Slist$
$currentOpt \leftarrow \infty$
for all $S_i \in Serv_j$ **do**
  $Result \leftarrow SelectServices(CleanList(Slist, S_i), Cost + q_i, Cover \cup S_i)$
  if $Result.Cost < currentOpt$ **then**
    $currentOpt \leftarrow Result.Cost$
    $currentCover \leftarrow Result.Cover$
  end if
end for
return $[currentOpt, currentCover]$

The recursive function $SelectServices$ performs the core task. In each recursive call one list element is considered to select a service for the corresponding activity. Note that the $currentCover$ is undefined if no solution is possible with the existing partial choice of services. This is indicated by the optimal cost of ∞.

Since the selected service can cover additional activities, we have to adjust the list of service sets. This is done in the function $CleanList$. We do not need to search a service for an activity which is already covered by another service. Hence, the corresponding list elements are discarded. In addition to this, we cannot use services which contain activities that are covered by already chosen activities. Thus, these have to be removed, too. Finally, the function $SelectServices$ returns the minimal costs to cover all activities as well as the corresponding set of services.

Function 2 Remove the effects of S from $Slist$

$\boldsymbol{CleanList(Slist, S) : Slist}$
for all list elements from $Slist$ **do**
  if list element belongs to an activity covered by service S **then**
    remove list element from $Slist$
  end if
end for
for all remaining list elements $Serv_j$ of $Slist$ **do**
  remove all $S_i \in Serv_j$ with $S_i \cap S \neq \emptyset$
end for
return $Slist$

Figure 4 shows an example of a simple subworkflow consisting of four activities. We assume that activity A is $ready$ and the predicates on all edges evaluate to $true$. There are five services $S_1, \ldots, S_5$ which offer to perform parts of this subworkflow. Table 1 shows the prices of these services. A correct selection of services would be, for example, $O_1 = \{S_3, S_4, S_5\}$ as each activity is covered by exactly one service. The cost of this selection is 7. Note that the cover

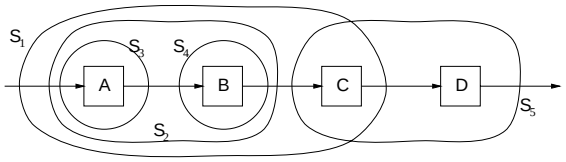

Figure 4: Services offered for a sub-workflow.

Service	S_1	S_2	S_3	S_4	S_5
Price	5	4	2	3	2

Table 1: Services and their prices.

$\{S_1, S_5\}$ is not allowed as the activity C is included in both services.

In the following, we sketch how the above algorithm processes this example. List 1 shows the initial status of $Slist$. Activity D is only included in service S_5. Activity C is included in S_1 as well as S_5. Activities A and B are each included in three services. The function $SelectServices$ starts with the first list element. Since only one service exists for D, S_5 is chosen. Before the next recursion level is entered, $Slist$ has to be adjusted to this choice. In addition to activity D, the service S_5 covers C, too. Therefore, the list entry for C is removed in the first loop of $CleanList$. Activity C is also included in service S_1. This precludes the choice of S_1. Hence, in the second loop S_1 is removed from the set of available services for A and B. The result is shown in List 2.

D	$\{S_5\}$
C	$\{S_1, S_5\}$
A	$\{S_1, S_2, S_3\}$
B	$\{S_1, S_2, S_4\}$

List 1: The initial status of $Slist$.

The next recursive steps reveal that besides O_1 there exists another exact cover namely $O_2 = \{S_2, S_5\}$ with cost 6. Thus, $SelectServices$ returns O_2 as the optimum.

4.1.2 Relaxing the Requirement for Exact Cover

So far we have assumed that each activity has to be covered by exactly one service. This implies that we cannot choose a service if it comprises an activity which is already covered by another service.

However, if we assume that a service provider that is offering a service S is willing to provide less comprehensive services, i.e., services that cover also subsets of activities $A(S)$, too, we can relax this restriction. In this case the requirement for disjoint services can be omitted and the problem we have to solve is a minimal weighted set cover. This problem is NP-complete, too [GJ79, Kar72]. However, many algorithms have been proposed in the literature for the solution of this problem, see for example [BH80, Bea87, BJ92, BC96a]. In addition to this, several heuristic

A	$\{S_2, S_3\}$
B	$\{S_2, S_4\}$

List 2: The status of $Slist$ after S_5 has been selected.

algorithms exist [BH80, Bea90, BC96b, CFT96, Pas97]. Thus we have a large set algorithms to compute a (nearly) optimal solution.

4.2 Outsourcing of Control Flow Decisions

In this section we describe how the complete workflow or more complex parts of the workflow which include control flow decisions can be outsourced in a single step. In this strategy, the set of activities which actually has to be performed is not completely known at the outsourcing time. Thus, services are assigned to activities which are not necessarily executed. This implies that for a correct execution it has to be ensured that each service executes only those activities, which are actually required, i.e., the selection of a service is only a reservation which ensures the potential execution of the activities. In addition to this, we allow that the price for a service depends on the set of activities which are actually executed. This helps to avoid situations in which otherwise a lot of resources have to be spent for services which are not executed. For example, in a workflow which requires the execution of exactly one of two expensive activities, it is not profitable to pay for the execution of both activities. Instead, it should be possible, e.g., in the case both activities are outsourced to different services, to cancel the execution of the activity which is not needed. The service which is affected by the cancellation would in this case charge only a limited fee for the reservation of the possibility to execute this activity.

4.2.1 The Optimization Algorithm

Similar as in section 4.1 the solution we require is a weighted cover or exact weighted cover, respectively. However, since the set of activities which are needed is not known when the outsourcing decision is made, the optimization criterion has to be different. Actually, we have to consider during the optimization the possible execution sequences and the probability of their occurrence.

In the following, we assume that we know the execution probability of each possible set of activities, respectively the conditional execution probability if the workflow already has executed certain activities. Estimations of these probabilities can be easily generated from the information which is available in the audit trail of a WfMS [AGL98]. The algorithm can then be sketched as follows:

1. Enumerate all possible (exact) covers of the workflow graph using algorithms similar as in section 4.1.

2. For each cover calculate the weighted mean of the costs of all possible execution sequences. The weight of each sequence is the probability of its execution.

3. Select the services in the cover with the lowest overall costs.

4.2.2 Application Considerations

Compared to the delayed selection of services at the time when an activity is known to be needed, i.e., the approach described in section 4.1, the early selection of services may result in additional reservation costs for those services which turn out to be (partially) unnecessary during the execution of the workflow. However, this does not necessarily result in higher overall costs. As the set of available services and their prices can vary, it might be advantageous to reserve services early when the prices are still low and early bird discounts can be used. A typical example for this situation are airplane tickets. In addition to this, bundle offers, which comprise several activities for a lower price, can be used. Such bundle offers would be excluded from the selection in section 4.1 when it is not yet clear at outsourcing time, whether all activities included in this service really have to be executed. Even if the early selection of services incurs higher costs, it might be desirable to choose this possibility as it guarantees at the start of the workflow that (1) services are available to execute the workflow and (2) an upper limit on the price of the execution of the workflow. These two guarantees might even be necessary to decide, whether the workflow should be executed at all.

5 Related Work

The need for workflows crossing the boundaries of organizations and for interoperability of WfMS has been identified widely in literature [GHS95, JLP+95, Geo98, SK98, MWW98a, WfM98]. Unfortunately, current WfMS products have several limitations with respect to interoperability [JLP+95, AAAM96, Geo98, Moh98]. Also, most research projects on the distributed execution of workflows in heterogeneous environments (like [SKM+96, MWW+98b, AHST98, Dog98]) do not consider federations of WfMS and the realization of cross-organizational workflows. The work in [CCPP96] addresses semantic workflow interoperability but assumes a global conceptual workflow model and a homogeneous environment which is not realistic in cross-organizational settings.

A notable exception is the Collaboration Management Infrastructure (CMI) project of MCC [MCC]. The vision of the CMI project is to improve the ability of real and virtual organizations to rapidly respond to continuous change and new opportunities by using collaborative process technologies for management of collaboration among people and software systems in large-scale, heterogeneous, ubiquitous and nomadic computing environments. Services in CMI are *single* activities with service agreements attached. They are modeled in the workflow definition as placeholder with service selection policy attached. In contrast to this, we offer a more flexible model, where whole parts of a workflow may be outsourced to different organizations. Services are not part of the workflow specification and can be selected based on flexible optimization criteria.

Efforts on standardizing workflow management technology are being made by the industrial coalition bodies WfMC and OMG, and by initiatives like SWAP. In the WfMC Interoperability Abstract Specification [WfM96a] the information and control flow between heterogeneous workflow engines are defined to be capable of supporting workflows that cross organizational boundaries. [WfM96b] defines a binding giving concrete type definitions and message formats for the realization of the abstract binding using E-Mail with MIME encoding as a transport mechanism. The jFlow (Joint Worflow Management Facility) submission [OB98] to the OMG Workflow Facility RFP is based on WfMC standards and addresses the requirements for interoperabil-

ity between different workflow object implementations in a CORBA environment. In this case, component WfMS should be able to interoperate via the CORBA middleware. A relatively new approach is the Simple Workflow Access Protocol (SWAP) [SWA98] which results from an initiative announced by Netscape, SUN, and HP in April 1998 and is now supported by the WfMC, too. SWAP is a simple protocol designed to allow for interoperability between between WfMS and other applications using HTTP. SWAP allows for the programmatic initiation of a workflow by an application via the Web, and the exchange of process data and state information between workflows and applications. Data exchanged is encoded using XML [W3C98].

All of these standardization efforts provide a technological infrastructure that can form the basis for realizing cross-organizational workflows. Which of them is most appropriate depends on the infrastructure available in the organizations participating in cross-organizational workflows (E-mail vs. CORBA vs. HTTP). However, none of them addresses really the problems and issues occurring in cross-organizational workflows like service modeling, service selection, and service monitoring. For example, the OMG jFlow submission defines an interface for associating resources with workflows and activities but states that the detailed specification is beyond the scope of the submission and realization of specific mechanisms and selection options is left to the implementations [OB98].

[GKT98] introduces a market-based approach to inter-organizational workflow management. Activities contained in a workflow are considered as goods traded on an electronic market consisting of service providers offering the execution of *single* activities within a workflow. A service broker together with a single-step bidding protocol is used to retrieve the available services at run-time. Based on information of the expected cost and duration of such a service execution and a fixed selection function (that combines costs and time of a service offering) a service is then selected for the execution of an activity. We do not propose a special selection function since it appears to be highly domain-dependent and specific for different workflow types. Moreover, the online-brokering approach seems not really appropriate for cross-organizational workflows since there will exist some more or less fixed business relationships with agreed service level among the cooperating organizations. [PR98] presents ongoing research in reducing the costs in workflow escalation but does not address cross-organizational workflow issues. Escalation refers to actions taken when workflow activities miss their deadlines. The approach may be useful to react to QoS deviations in cross-organizational workflows.

As already mentioned in section 3, most WfMS define the assignment of activities to resources, e.g., roles, in a fixed way. At run-time activities are assigned to agents that reply first to a work request. Only a few systems allow for a dynamic binding of resources and more systematic support for run-time management of resources [HA98]. In [MWW98a] several assignment strategies for flexible worklist management are described but they are only applicable to intra-organizational workflows. The strategies proposed implement specific business requirements such as the principle of dual control and balancing of the work assigned to an agent. Assignment strategies are modeled as (sub-)workflows which are then explicitly integrated into the overall workflow. We do not follow such an approach since we agree with [HA98] that there should be a separate model for describing the resources (i.e., services) and their allocation to activities in cross-organizational workflows. Separating the service model and the service selection strategies from the workflow specification allows for a flexible adaptation to changes in the business relationships.

6 Conclusions

In this paper we have proposed a service-oriented model for cross-organizational workflows. Modeling the workflow execution as a cooperation of services allows different organizations to interact via well-defined interfaces. We have applied this model to optimize the execution of cross-organizational workflows. Therefore, we have presented algorithms which allow to select services depending on their contribution to the quality criteria of the workflow.

In the context of the CROSSFLOW project [CC97, CC98], we plan to relax some the assumptions we made and complement the model with additional features. So far we have assumed that a full and reliable specification of each service including QoS parameters is available. However, external services might lack a complete specification and the service requester has only a limited a priory knowledge about the reliability of the service providers, i.e., the specification given by service providers about their services cannot be taken for granted. This requires that the service requester has to make its own observations about the behavior of the service and take this observations into account when making outsourcing decisions. Modeling the offered services based on the externally visible behavior is therefore one of our main topics for future work. These models will be constructed by analyzing previous executions and can form the basis for sound outsourcing decisions.

Besides the static analysis of previous service executions, a service which is currently running has to be monitored, too. Possible quality of service deviations and exceptional situations have to be detected and appropriate actions have to be taken. For example, we have to take into account that the execution of an activity by a service fails or can only be provided with a reduced quality. Modeling these situations, detecting them at runtime, and providing reaction policies using the available level of control for a service will therefore be investigated, too. The simplest reaction to deviations is for example to notify a person to handle the problem. Other deviations might be handled in an automatic way by using pre-specified rules, e.g, terminating the service and selecting an alternative one, select a new set of services for the overall workflow, or terminate the overall workflow.

References

[AAAM96] G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan. Functionality and limitations of current workflow management systems. *IEEE Expert Journal*, 12(5), 1996.

[AGL98] R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *Proc. 6th Intl. Conference on Extending Database Technology (EDBT'98)*, Valencia, Spain, March 1998.

[AHST98] G. Alonso, C. Hagen, H.-J. Schek, and M. Tresch. Towards a platform for distributed application development. In Dogac et al. [DÖBS98].

[BC96a] E. Balas and M. C. Carrera. A dynamic subgradient-based branch and bound procedure for set covering. *Operations Research*, 44:875–890, 1996.

[BC96b] J. E. Beasley and P. C. Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94:392–404, 1996.

[Bea87] J. E. Beasley. An algorithm for set covering problems. *European Journal of Operational Research*, 31:85–93, 1987.

[Bea90] J. E. Beasley. A lagrangian heuristic for set covering problems. *Naval Research Logistics*, 37:151–164, 1990.

[BH80] E. Balas and A. Ho. Set covering algorithms using cutting planes, heuristics and subgradient optimization: A computational study. *Mathematical Programming Study*, 12:37–60, 1980.

[BJ92] J. E. Beasley and K. Jörnsten. Enhancing an algorithm for set covering problems. *European Journal of Operational Research*, 58:293–300, 1992.

[CC97] CROSSFLOW-Consortium. CROSSFLOW: Cross-organizational workflow. Proposal for the European Programme for RTD in Information Technologies (ESPRIT), December 1997.

[CC98] CROSSFLOW-Consortium. CROSSFLOW: Project programme. ESPRIT Proposal No. 28635, July 1998.

[CCPP96] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Semantic workflow interoperability. In *Proc. 5th Intl. Conference on Extending Database Technology (EDBT'96)*, Avignon, France, March 1996.

[CFT96] A. Caprara, M. Fischetti, and P. Toth. A heuristic algorithm for the set covering problem. In *Proc. of the fifth Integer Programming and Combinatorial Optimization Conference*, volume 1084 of *LNCS*, pages 72–81, 1996.

[CHRW98] A. Cichocki, S. Helal, M. Rusinkiewicz, and D. Woelk. *Workflow and Process Automation - Concepts and Technology*. Kluwer Academic Publishers, Boston, 1998.

[DÖBS98] A. Dogac, T. Öszu, A. Biliris, and T. Sellis, editors. *Advances in Workflow Management Systems and Interoperability*. NATO-ASI Series. Springer Verlag, 1998. NATO Advanced Study Institute on Workflow Management Systems and Interoperability. Istanbul, Turkey. August, 1997.

[Dog98] A. Dogac et. al. Design and implementation of a distributed workflow management system: METUFlow. In Dogac et al. [DÖBS98].

[Geo98] D. Georgakopoulos. World wide workflow: The vision and the state-of art in products and research. In Dogac et al. [DÖBS98].

[GHS95] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview on workflow management: From process modelling to workflow automation. *Distributed and Parallel Databases*, 3(2), April 1995.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.

[GKT98] A. Geppert, M. Kradolfer, and D. Tombros. Market-based workflow management. In *Proc. IFIP Conference on Distributed Systems for Electronic Commerce*, Hamburg, Germany, June 1998.

[HA98] Y. Han and A.Sheth. On adaptive workflow modeling. In *Proc. 4th International Conference on Information Systems Analysis and Synthesis*, Orlando, Florida, July 1998.

[HC97] Y. Hoffner and B. Crowford. Using interception to create domains in distributed systems. Technical Report RZ 2906 (#91501), IBM Research Division, Zurich, Switzerland, February 1997.

[Hoc97] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing, Boston, 1997.

[Hof96] Y. Hoffner. Interoperability and distributed platform design. In *Proc. IFIP/IEEE International Conference on Distributed Platforms*, Dresden, Germany, February 1996.

[JLP+95] J. Juopperi, A. Lehtola, O. Pihlajamaa, A. Sladek, and J. Veijalainen. Usability of some workflow products in an inter-organizational setting. In *Proc. of IFIP WG8.1 Working Conference on Information Systems for Decentralized Organizations*, Trondheim, Norway, August 1995.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[Kim95] W. Kim, editor. *Modern Database Systems: The Object Model, Interoperability, and beyond*. Addison-Wesley Publishing Company, 1995.

[LA94] F. Leymann and W. Altenhuber. Managing business processes as an information resource. *IBM Systems Journal*, 33(2), 1994.

[MCC] MCC. CMI - Collaboration management infrastructure for comprehensive process and service management. Home page: www.mcc.com/projects/cmi; Overview slides: www.mcc.com/projects/cmi/overview/CMI_Overview/-sld001.htm.

[Moh98] C. Mohan. Recent trends in workflow management products, standards and research. In Dogac et al. [DÖBS98].

[MWW98a] P. Muth, J. Weissenfels, and G. Weikum. What workflow technology can do for electronic commerce. In *Proc. EURO-MED NET Conference*, 1998.

[MWW+98b] P. Muth, D. Wodtke, J. Weissenfels, G. Weikum, and A. Kotz-Dittrich. Enterprise-wide workflow management based on state and activity charts. In Dogac et al. [DÖBS98].

[OB98] OMG-BODTF. Workflow management facility. Technical Report bom/98-06-07, OMG Business Object Domain Task Force, July 1998. OMG BODTF RFP #2 Submission.

[Pas97] V. T. Paschos. A survey of approximately optimal solutions to some covering and packaging problems. *ACM Computing Surveys*, 29:171–209, 1997.

[PR98] E. Panagos and M. Rabinovich. Reducing escalation-based costs in WFMSs. In Dogac et al. [DÖBS98].

[RKT+95] M. Rusinkiewicz, W. Klas, T. Tesch, J. Wäsch, and P. Muth. Towards a cooperative transaction model: The cooperative activity model. In *Proc. of the 21st Int. Conference on Very Large Databases*, pages 194–205, September 1995. Zurich, Switzerland.

[RS95] M. Rusinkiewicz and A. Sheth. Specification and execution of transactional workflows. In Kim [Kim95], chapter 29, pages 592–620.

[SGJ+96] A. Sheth, D. Georgakopoulos, S. Joosten, M. Rusinkiewicz, W. Scacchi, J. Wileden, and A. Wolf. Report from the NSF workshop on workflow and process automation for information systems. Athens, GA, May 1996.

[SK98] A. Sheth and K. Kochut. Workflow applications to research agenda: Scalable and dynamic work coordination and collaboration systems. In Dogac et al. [DÖBS98].

[SKM+96] A. Sheth, K. Kochut, J. Miller, D. Worah, S. Das, C. Lin, D. Palaniswami, J. Lynch, and I. Shevchenko. Supporting state-wide immunization tracking using multi-paradigm workflow technology. In *Proc. 22nd Intl. Conf. on Very Large Databases (VLDB'96)*, Bombay, India, September 1996.

[SWA98] SWAP. Simple workflow access protocol (SWAP). Technical Report people.netscape.com/kswenson/SWAP/SWAP9805.pdf, SWAP Working Group, May 1998. Draft Proposal.

[W3C98] W3C. Extensible Markup Language (XML) 1.0. Technical Report www.w3.org-/TR/1998/REC-xml-19980210, World Wide Web Consortium, February 1998. W3C Recommendation 10-February-1998.

[WfM96a] WfMC. Workflow standard - Interoperability: Abstract specification. Technical Report WFMC-TC-1012, Workflow Management Coalition, October 1996. Version 1.0.

[WfM96b] WfMC. Workflow standard - Interoperability: Internet e-mail binding. Technical Report WFMC-TC-1018, Workflow Management Coalition, October 1996. Version 1.0.

[WfM96c] WfMC. Workflow standard - Terminology & glossary. Technical Report WFMC-TC-1011, Workflow Management Coalition, June 1996. Version 2.0.

[WfM98] WfMC. Workflow and Internet - Catalyst for radical change. White paper, Workflow Management Coalition, June 1998.

[WMW98] J. Weissenfels, P. Muth, and G. Weikum. Flexible worklist management in a light-weight workflow management system. In *Proc. of the EDBT Workshop on Workflow Management Systems*, Valencia, Spain, March 1998.