Deriving Service Models in Cross-Organizational Workflows

Justus Klingemann, Jürgen Wäsch, Karl Aberer

Integrated Publication and Information Systems Institute (GMD-IPSI)
GMD – German National Research Center for Information Technology
Dolivostr. 15, D-64293 Darmstadt, Germany

{klingem, waesch, aberer}@darmstadt.gmd.de
http://www.darmstadt.gmd.de/~{klingem, waesch, aberer}

In Proceedings of
RIDE - Information Technology for Virtual Enterprises (RIDE-VE '99),
Sydney, Australia, March 23-24, 1999

Deriving Service Models in Cross-Organizational Workflows*

Justus Klingemann, Jürgen Wäsch, Karl Aberer
German National Research Center for Information Technology (GMD)
Integrated Publication and Information Systems Institute (GMD-IPSI)
Dolivostraße 15, D-64293 Darmstadt, Germany
{klingem,waesch,aberer}@darmstadt.gmd.de

Abstract

Competitive markets force companies to form virtual enterprises by outsourcing activities to external service providers. The workflow concept has been very successful in streamlining business processes by automating the coordination of activities, but has so far been limited to the use within single organizations. To address the problems of cross-organizational workflows we use a service-oriented workflow model. Within this approach, we present a technique how to derive a model of the external services, based on continuous-time Markov chains, by analyzing their externally observable behavior. This allows to assess the quality of external services, without compromising the autonomy of the service providers.

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 urges organizations to form virtual enterprises by outsourcing activities to external service providers. Hence, business links with other organizations have to be set up and managed. This has to be achieved in a fast and flexible way to guarantee a short time to market while allowing a dynamic reaction to new customer demands and changing offers of service providers in electronic commerce environments.

Information technology has provided different tools to address these requirements. The workflow concept [10, 3, 8, 5] 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 provides the possibility to efficiently exchange data and communicate with a large number of possible service providers. Thus, workflow management systems (WfMS) can limit their scope no longer to a single organization but have to exploit the network infrastructure to cross organizational boundaries.

However, the extension of workflows beyond the borders of a single enterprise raises new challenges. One important challenge is the necessity to choose among different services that potentially satisfy the customers requirements. In particular, it has to be decided which activities or group of activities should be outsourced to which business partners. Relevant criteria with regard to that decision are the required time, cost or the adherence to domain-specific quality of service parameters.

The initiator of the workflow has only limited control over the outsourced activities due to the autonomy of the participating organizations. On the one hand, 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 extent. On the other hand, 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, together with the limited means of control, requires that the service requester has to make its own observations about the behavior of the service provider and take these observations into account when making outsourcing decisions. However, the service requester is not able to monitor directly the internal processing of the service provider. Therefore, he has to rely on the externally visible behavior and derive from that his model of the service.

In [15] we have presented a service-oriented approach to cross-organizational workflows that models a typical interface between service requester and service provider. In this paper, we apply this approach to present a technique how to derive a model of the external services, based on

*This work was partially done in the context of the ESPRIT-IV project CROSSFLOW. 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).

the continuous-time Markov chain model, by analyzing the externally observable behavior of a service. This allows to assess the quality of external services, without compromising the autonomy of the service providers.

The paper is organized as follows. 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 present our continuous-time Markov chain approach. We describe how the parameters are computed from the log of previous service executions and how the model can be applied to asses services in cross-organizational workflows. After discussing related work in section 5, we summarize our results and discuss future work.

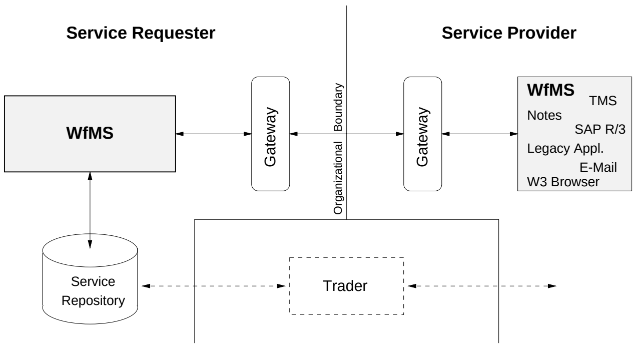

Figure 1. A simplified architecture for cross-organizational workflows.

2. Cross-organizational workflows

Workflow management is a rapidly evolving technology which is being increasingly exploited in a variety of industries [10, 24, 19, 5]. Its primary mission is to handle business processes. A *business process* is a set of one or more interconnected activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships [29]. A *workflow* is the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another, according to a set of predefined rules [29]. 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 [4].

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.

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 [15]. 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 [12, 13] 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. 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. [13, 4].

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. However, we confine ourselves to the essential concepts which are necessary to understand the approach presented in this paper. For further details see [15]. We consider a workflow as a collection of activities which are related by certain dependencies. A workflow is modeled as a graph with activities as nodes and edges which represent the control and data flow. Formally, a *workflow* W has the following constituents:

- Formal *input* parameters $input(W)$ and *output* parameters $output(W)$. These parameters are used to provide the input for starting activities or are generated as output from terminating activities, respectively.
- A set of *constituent activities* $A_W = \{A_1, \ldots, A_n\}$. Each activity represents a unit of work in the workflow. Activities can have certain Quality of Service (QoS) parameters assigned.
- A directed graph $CF_W = (A_W, C_W)$, $C_W \subseteq A_W \times A_W$ that describes the *control flow* in the workflow. Each edge $c = (A, B) \in C_W$ 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.
- 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. Each edge $d = (A, B) \in D_W$ 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.
- A list of *quality of service* parameters $QoS(W)$. Examples are the maximal duration and the maximal cost allowed for a workflow. 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*.

As indicated above, the basic entity in our model is a *service* [15]. Informally, a service is 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 covers. In general, a service is not restricted to execute a single activity of a workflow, it can span multiple activities. With each service a *service provider* is associated. This can be either an internal resource or an *external organization*. 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, the service requester has only a limited possibility to get information on 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 *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.

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 form of control and notification events [15]. Control events (specified in the service description) can be sent by the service requester to the service provider in order to influence the processing of the service. Notification events are used to inform the service requester about the state of the processing of the external service.

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.

Figure 3 illustrates an example of the structural part (which is complemented with attributes like quality of service parameters and an identifyer for the service provider) of a 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

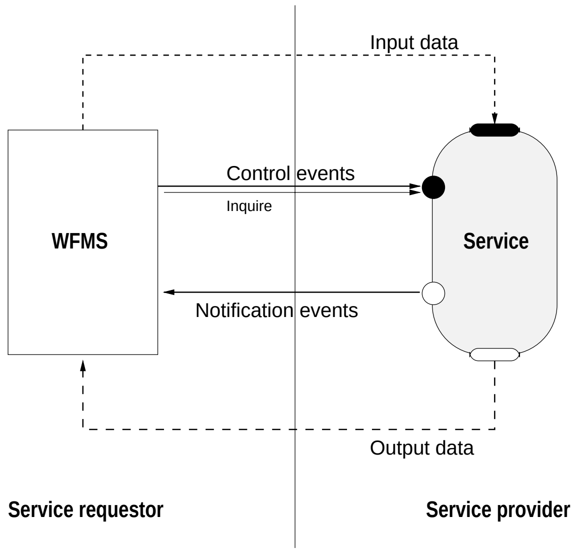

Figure 2. Interfaces of a service and interaction with the WfMS

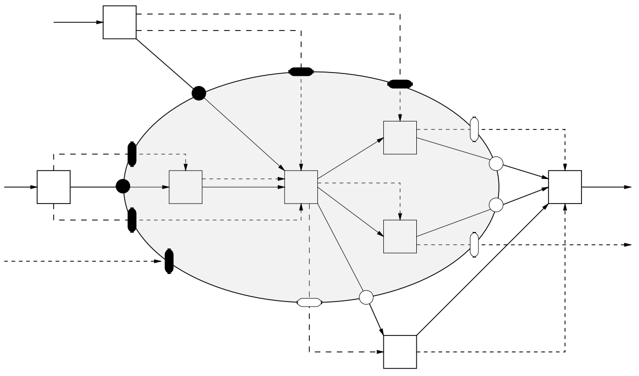

Figure 3. Service and its interfaces to a workflow

service and the rest of the 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.

For a description of the execution model for cross-organizational workflows and the possible service selection policies, we refer the reader to [15].

4. Modeling services with continuous-time Markov chains

In this section we describe how the external behavior of a service can be used to derive a model of this service. To build a model we need information collected during former performances of the service. Hence, the model is an abstraction of the observed performances. A general assumption to use the model as a means to make predictions into the future is therefore, that the observed behavior represents the future behavior, especially, that the behavior of the service is reasonably stable.

Information about past performances of a service is collected in a *log* which is a list of events with a timestamp. We assume that the visible behavior of a service consists of events which denote the start of an activity and the corresponding end of an activity. Note, that not all activities of a service need to be present at the external interface. Which information is made available by the service provider depends on the service description. With $\mathcal{A}(S)$ we denote the set of activities whose start and end is signaled by the service S. If the service is clear from the context we will just use $\mathcal{A}$. Formally, a log for a service S contains for each service execution a list of records (A, E, T), where A denotes the activity, $E \in \{start, end\}$ is the type of the event and T is the time the event occurred.

4.1. The continuous-time Markov chain model

A continuous-time Markov chain (CTMC) is a stochastic process that proceeds through different states in certain time epochs. The Markov property states that the probability of entering a state depends only on the current state and not on the previous history (we only consider first-order Markov chains). Continuous-time Markov chains extend discrete Markov chains by adding mean residence times for states. Thus, the Markov model which is derived from the log consists of a set of states $\mathcal{S}$ and transitions between these states. Each transition is labeled with a transition probability p_{ij} and each state s_i is labeled with a holding time h_i which denotes the mean amount of time the service resides in this state. For modeling services we proceed as follows. To capture parallel activities, each state represents the set of activities of a service which is active at a certain point in time, i.e., the elements of the powerset of all visible activities $\mathcal{A}$ correspond to possible states of the Markov chain. Thus, $\mid \mathcal{S} \mid = 2^{|\mathcal{A}|}$ and $Act(s_i)$ denotes for each $s_i \in \mathcal{S}$ the set of activities which are active in this state. For practical cases we can assume that only a limited number of states are reached. In general, the number of states is small for structured workflows as they impose restrictions on what activities can be active at the same time.

In order to introduce a simple QoS model, we allow that each activity $a_i \in \mathcal{A}$ is assigned a cost c_i. The cost incurs when an activity is started. Therefore, each transition is labeled with the sum of the costs of all activities which are active in the destination node but not in the source node. Formally, the cost c_{ij} assigned to the edge from s_i to s_j is the sum of the costs of all activities in $Act(s_j) \setminus Act(s_i)$. The ability to assign costs to activities allows to model services which do not have a fixed price but are charged depending on what activities actually have to be executed. In this case the costs per activity are made available by the service provider. A special case of this situation is the service provider itself. If he wants to calculate the price of the service and needs information on how expensive a service execution is, he can use the model to represent his own service. The costs can also be used to represent possible contributions of the service requester. Such costs occur if the service provider and the service requester have to cooperate on certain activities.

4.2. Determining the parameters

In this subsection, we describe how the parameters of the Markov model are calculated. A continuous-time Markov chain is uniquely determined by the values p_{ij} of transition probabilities between states and the holding times h_i of the states. These parameters are estimated using the information collected in the log. A log $\mathcal{L} = \{L_1, \ldots, L_m\}$ consists of m elements called *service logs* each describing a single execution of the service to be modeled.

Each service execution starts with $s_\emptyset$, i.e., the empty set of active activities. We assume that the service ends when $s_\emptyset$ is entered again. Starting in the state $s_\emptyset$, we sequentially scan the log of each service execution as follows: If the next log entry denotes the start of an activity a_k, a state transition from the current state s_i to the state s_j with $Act(s_j) = Act(s_i) \cup \{a_k\}$ occurs. If the next log entry denotes the end of an activity a_l, a state transition from the current state s_i to the state s_j with $Act(s_j) = Act(s_i) \setminus \{a_l\}$ occurs. We allow that more than one event can occur at a certain point of time. In this case a transition into a state which combines the effects of all events occurs.

While scanning the log, we build up a matrix (t_{ij}) which records all state transitions. The rows and columns which correspond to the states are added dynamically to keep the matrix as small as possible. Remind, that usually only a small fraction of state combinations actually occurs, thus the resulting matrix is sparse. Each newly added row or column vector is initialized with zeros and indexed with the identifier of the state. Initially the matrix consists only of one row and one column for $s_\emptyset$. When a state transition from s_i to s_j occurs, the following happens: If the state s_j is already present in the matrix, we increment the entry t_{ij} by one. Otherwise, we first have to extend the transition table by a row and a column for s_j and then increment the entry, t_{ij}, too.

Let the set of states which actually occurred during the service executions recorded in $\mathcal{L}$ be $\mathcal{S}(\mathcal{L})$. $\mathcal{S}(\mathcal{L})$ is calculated during the construction of (t_{ij}). To calculate the transition probabilities we have to normalize the transition frequencies. Therefore, we compute the total number of transitions leaving the corresponding state by calculating the sum of each row in (t_{ij}). With $t_i = \sum_{s_j \in \mathcal{S}(\mathcal{L})} t_{ij}$ for $s_i \in \mathcal{S}(\mathcal{L})$ we get $p_{ij} = \frac{t_{ij}}{t_i}$ for $s_i, s_j \in \mathcal{S}(\mathcal{L})$.

Due to the Markov property, the holding time for each state is exponentially distributed. Since the maximum likelihood estimator for an exponentially distributed variable is the mean we can easily calculate this value from the log by taking the mean of the amount of time the service resides in this state. Each event results in a state transition. Thus, when we scan the log as described above, we can calculate the residence time by taking the difference between the timestamp the state is entered and the succeeding timestamp. The mean of these values is then an estimate of the holding time h_i for the corresponding state s_i.

In the following, we give an example how to construct a continuous-time Markov chain from a set $\mathcal{L}$ of service logs. For the sake of simplicity, we write $A_i(t_s, t_e)$ to denote the execution of activity A_i starting at time t_s and ending at time t_e. For example, in L_1 activity A starts at time 0 and ends at time 2. Assume, the set $\mathcal{L} = \{L_1, L_2, L_3, L_4\}$ consists of the following service logs:

$$
\begin{aligned}
L_1 &= [A(0,2),\ B(2,3),\ D(3,4)] \\
L_2 &= [A(0,4),\ C(4,8),\ D(8,10)] \\
L_3 &= [A(0,4),\ C(4,8),\ B(8,11),\ D(11,12)] \\
L_4 &= [A(0,3),\ C(3,7),\ B(4,8),\ D(8,10)]
\end{aligned}
$$

Table 1 shows the matrix $M_1 = (t^1_{ij})$ after service log L_1 has been processed. Since there are no “overlapping” activities in L_1, $\mathcal{S}(\{L_1\})$ consists only of four states, i.e., $\{S_\emptyset, S_{\{A\}}, S_{\{B\}}, S_{\{D\}}\}$. For each state transition occurring in L_1 there is an entry in M_1. Additionally, Table 1 shows the average duration for each activity occurring in the log and the average residence time for each state induced by the service logs processed so far.

Table 2 shows the matrix $M_4 = (t^4_{ij})$ after processing all service logs $L_i \in \mathcal{L}$. State $S_{\{B,C\}}$ in M_4 results from the overlapping of activities B and C in L_4. Note that due to this overlapping the average activity duration for activity B and C differs from the mean residence times for $S_{\{B\}}$ and $S_{\{C\}}$, respectively.

Figure 4 shows the continuous-time Markov chains derived from Tables 1 and 2, respectively. For each row in the matrices, there is a state in the corresponding CTMC. For each entry in the matrices, there is a transition in the corresponding CTMC. The transition probabilities p_{ij} are equal to the matrix entries t_{ij} divided by the sum of all entries in row i. For example, $p_{S_{\{A\}}S_{\{B\}}}$ is set to $\frac{1}{1+3}$. The holding times h_i for the states s_i are equal to the average residence times as shown in Tables 1 and 2.

(t_{ij}^1)	$S_\emptyset$	$S_{\{A\}}$	$S_{\{B\}}$	$S_{\{D\}}$	avg. residence time	avg. activity duration	activity
$S_\emptyset$		1			–	–	
$S_{\{A\}}$			1		2/1	2/1	A
$S_{\{B\}}$				1	1/1	1/1	B
$S_{\{D\}}$	1				1/1	1/1	C

Table 1. Matrix M_1 derived from service log L_1.

(t_{ij}^4)	$S_\emptyset$	$S_{\{A\}}$	$S_{\{B\}}$	$S_{\{D\}}$	$S_{\{C\}}$	$S_{\{B,C\}}$	avg. res. time	avg. act. duration	activity
$S_\emptyset$		4					–	–	
$S_{\{A\}}$			1		3		13/4	13/4	A
$S_{\{B\}}$				3			5/3	8/3	B
$S_{\{D\}}$	4						6/4	6/4	D
$S_{\{C\}}$			1	1		1	9/3	12/3	C
$S_{\{B,C\}}$			1				3/1	–	

Table 2. Matrix M_4 after processing all service logs $L_i \in \mathcal{L}$.

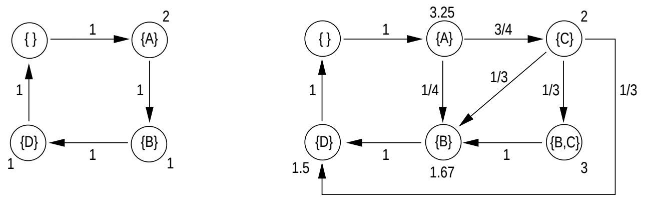

Figure 4. Continuous-time Markov chains computed from matrix M_1 (left) and M_4 (right).

From the CTMC shown in the right part of Figure 4, we can easily make some initial observations about the behavior of the service: Each service execution started with activity A and ended with activity D. Between A and D either B, C or both were executed. If both B and C were executed C always started before B started and ended before B ended.

4.3. Applications of the model

In general, the derived continuous-time Markov chain will be used to validate a service specification, i.e., to compare the observed behavior with the properties claimed by the service provider. This enables the service requester to assess a service offer independently from the reliability of the service provider. It can also be used by the service requester to determine the parameters of its own service and, hence, can support him in the generation of his service offer.

A CTMC can represent the externally visible behavior of a service in a concise way. This allows a graphical representation of the service which gives the service provider as well as the service requester an impression of the run-time behavior of the service. For example, it can be seen that certain activities were never executed in parallel or were always executed in a certain order. Since we calculate the transition properties, we can also make more subtle assertions, e.g., that certain transitions occurred rarely or frequently. In addition, several numerical characteristics are made available through the Markov model. On the one hand, the model can be used as a basis for simulations of the service. On the other hand, the theory of continuous-time Markov chains provides us with tools to perform transient analysis of the CTMC representation of the service. Due to space limitations we can only sketch the possibilities. For the actual algorithms we refer the reader to [26].

Our main interest is in predicting the expected time or cost of certain parts or the overall service. The CTMC allows to calculate the expected time until a state s_j or a member of a set of states $\mathcal{S}_j$ is reached the next time given that the current state is s_i. This can be used to calculate the expected duration of the overall service. It also allows to calculate the expected remaining time given that the service has already processed certain activities. Another interesting property which can be determined is the expected time until a certain activity is processed. Often it is sufficient that only a certain part of the service is carried out fast and that the corresponding results are delivered whereas the remaining part of the service is less urgent. This is for example the case if other activities outside the observed service depend on selected results. In a similar fashion the expected cost for parts or the overall service can be calculated.

The mean execution time of an activity itself is directly calculated during the estimation of the parameters.

The Markov model can also be used to make assertions about the order the states of a service are traversed. The transition probabilities can be used for this purpose. For example, the probability that a specific sequence of states is executed next is simply the product of the corresponding transition probabilities.

5. Related work

Although there is a common agreement that logging and analysis of workflow executions are important tasks in workflow management [10, 27, 24, 25, 21, 20, 9], little work has been done in the area of analyzing and mining the histories of workflows.

Agrawal et al. [1, 2] consider the problem to generate a workflow model from a log of executions produced by a preexisting system which uses a different (usually less formal) representation. the developed models differ due to the different application contexts. Whereas Agrawal et al. aim at a model which forms the basis for a later execution by a WfMS, our model aims at analyzing services to guide the service selection [15]. The difference becomes most evident when looking at the available data when using the model. During the execution of a workflow all runtime data like input and output parameters are available. On the other hand, if we use the external model of a service for analysis and simulation purposes this runtime data is not available and we have to rely solely on statistical data. In this scenario the transition probabilities are valuable since they allow us to make assertions about transitions without the evaluation of predicates. Moreover, the focus of Agrawal et al. is on the generation of edges and nodes of the workflow graph. Ideas how to learn the edge predicates are only sketched and left for further work.

The work of Cook and Wolf [6, 7] goes in a similar direction as the paper described above. The main differences are the workflow model used and the techniques to derive the model. Whereas Agrawal et al. use a simplified version of the FLOWMARK meta model [17], Cook and Wolf model the workflow as a finite state machine. Agrawal et al. generate their workflow model using a data mining technique whereas Cook and Wolf experiment with several techniques including neural networks, a deterministic algorithm which groups partial strings into equivalence classes depending on the future behavior, and Markov chains. Both approaches model only the possible sequences of activities. Information about the duration or cost of a workflow or parts thereof are not addressed.

Other approaches in the area of workflow logging like [11, 16, 27] focus on the storage and querying of workflow histories. Both [11] and [16] only allow analysis of past workflow executions by posing OQL queries against the workflow log. Weikum [27] discusses different techniques to implement an application-level workflow log. Sophisticated analysis techniques for workflow logs are not addressed in these papers. The CMI project [18] is also concerned with process monitoring [23] but its focus is geared towards an awareness model instead of an analysis model.

The Audit Data Specification of the WfMC [28] defines the status changes to be reported by a WfMS and the form of the audit trail records to be produced when such a status change happens. The `WfEventAudit` interface of the OMG jFlow submission [22] is based on [28]. Both specifications only provide rudimentary querying capabilities for workflow histories, but they form a basis for building more complex analysis techniques as the one discussed in this paper on top of them.

6. Conclusions

In this paper, we have described a technique that allows to derive a model of external services in cross-organizational workflows from the externally observable service behavior. Our approach is based on continuous-time Markov chains that can be incrementally constructed from the log of the past executions of services. This allows the service requester to build up an external model of services and to asses their quality, without compromising the autonomy of the service providers. This assessment can guide the service selection policies and outsourcing strategies in cross-organizational workflows [15].

The work presented in this paper can also be applied to ordinary enterprise-wide workflows. Our model can be utilized to analyze and assess the efficiency, accuracy, and the timeliness of the (real) enterprise's business processes, too, or of the cross-organizational workflow as a whole. The information extracted provides the feedback for continuous business process engineering.

Future work includes the relaxation of some assumptions made in this paper. For example, it is currently required in our model that at least one activity of a service is active at any point in time during the service execution. This assumption can be easily avoided by introducing dedicated waiting states into the continuous-time Markov chain model. These waiting states represent time periods where no observable activity is present. Another extension addresses the requirement that for each activity that is externally visible both the start event and the end event have to be present in the service log. Currently, we are generalizing our model in a way such that it is able to cope with arbitrary events that can be externally observed.

References

[1] R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. Technical Report RJ 10100

(91916), IBM Almaden Research Center, December 1997.

[2] 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.

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

[4] CROSSFLOW-Consortium. CROSSFLOW: Cross-organizational workflow in virtual organizations – Project programme. ESPRIT Proposal No. 28635, July 1998.

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

[6] J. E. Cook and A. L. Wolf. Automating process discovery through event-data analysis. In *Proc. of the 17th Int. Conference on Software Engineering*, Seattle, Washington, April 1995.

[7] J. E. Cook and A. L. Wolf. Discovering models of software processes from event-based data. Technical report CU-CS-819-96, Department of Computer Science, University of Colorado, November 1996.

[8] A. Dogac, T. Öszu, A. Biliris, and T. Sellis, editors. *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.

[9] D. Georgakopoulos. World wide workflow: The vision and the state-of art in products and research. In Dogac et al. [8].

[10] 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.

[11] A. Geppert and D. Tombros. Logging and post-mortem analysis of workflow executions based on event histories. In *3rd Intl. Conference on Rules in Database Systems (RIDS)*, Skvde, Sweden, June 1997.

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

[13] Y. Hoffner and B. Crowford. Using interception to create domains in distributed systems. In *Proc. of the Joint Intl. Conference on Open Distributed Platforms and Distributed Systems*, Toronto, Canada, May 1997.

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

[15] J. Klingemann, J. Wäsch, and K. Aberer. Adaptive outsourcing in cross-organizational workflows. GMD Report 30, GMD – German National Research Center for Information Technology, August 1998.

[16] P. Koksal, S. Arpinar, and A. Dogac. Workflow history management. *ACM SIGMOD Record*, 27(1):67–75, 1998.

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

[18] 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, 1998.

[19] C. Mohan. Recent trends in workflow management products, standards and research. In Dogac et al. [8].

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

[21] 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. [8].

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

[23] M. Rashid and S. Helal. The CEDMOS reference architecture. Technical Report Technical Report CMI-136-97, MCC, Austin, TX, October 1997.

[24] 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.

[25] A. Sheth and K. Kochut. Workflow applications to research agenda: Scalable and dynamic work coordination and collaboration systems. In Dogac et al. [8].

[26] H. C. Tijms. *Stochastic Models – An Algorithmic Approach*. John Wiley & Sons, 1994.

[27] G. Weikum. Workflow monitoring: Queries on logs or temporal databases? In *Proc. High Performance Transaction Systems Workshop (HTPS'95)*, September 1995. Position paper.

[28] WfMC. Audit data specification. Technical Report WFMC-TC-1015, Workflow Management Coalition, November 1996. Version 1.0.

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