Avoidance of Deadline-violations for Inter-organizational Business Processes

Johann Eder
University of Vienna, Austria
Department of Knowledge and Business Engineering

Horst Pichler and Stefan Vielgut
University of Klagenfurt, Austria
Department of Informatics-Systems

Abstract

Workflow time management allows the prediction of eventually arising deadline violations and enables proactive initiation of evasive actions. This saves time, avoids unnecessary task-compensations and therefore decreases costs. Current time management approaches assume that communication with external tasks, which are enacted by the process, is conducted in a blocking or synchronous manner. As inter-organizational processes frequently communicate in a non-blocking manner we examined basic asynchronous communication patterns and provided a mapping for each pattern in order to use it for time management purposes.

1. Introduction

Workflow management systems, are used to improve processes by automating tasks and getting the right information to the right place for a specific job function. As automated business processes often span several enterprises, the most critical need in companies striving to become more competitive is a high quality of service, where the expected process execution time ranks among the most important quality measures [11]. Additionally it is a necessity to control the flow of information and work in a timely manner by using time-related restrictions, such as bounded execution durations and absolute deadlines, which are often associated with process activities and sub-processes [7]. However, arbitrary time restrictions and unexpected delays can lead to time violations, which typically increase the execution time and cost of business processes because they require some type of exception handling [17].

Although currently available commercial products offer sophisticated modelling tools for specifying and analyzing workflow processes, their time-related functionality is still rudimentary and mostly restricted to monitoring of constraint violations and simulation for process re-engineering purposes [4, 10]. Workflow time management deals with these problems and allows for instance the prediction of response times or proactive avoidance of constraint violations. In research several attempts have been made to provide solutions to advanced time management problems (e.g. [3, 4, 6, 8, 10, 15]).

Nowadays business processes spread over the boundaries of companies and integrate customers, suppliers and partners to achieve inter-organizational business goals. Inter-organizational workflows may therefore be assembled from several external processes and services [1]. More than ever slow external services will have a disastrous impact on the overall process response time, cause deadline violations and increase the cost of the process. It seems to be an obvious idea to apply time management approaches to avoid these problems. The problem is that asynchronous communication patterns are not supported by existing time management approaches and that inter-organizational business process communication is asynchronous by nature. In this paper we examine basic asynchronous process communication patterns [1, 16, 19], show how they affect the process execution and duration, and adapt our time management algorithms and calculation operations to cope with the new requirements.

The paper is organized as follows. In Section 2 we describe basic workflow and time management concepts. Section 3 list several synchronous and asynchronous communication patterns. Then we define some necessary prerequisites in Section 4 in order to define mappings of the communication patterns for time management calculations, as explained in Section 5. How to apply the concepts to a workflow system is dealt with in Section 6 and finally the paper finishes with some conclusions and a brief outlook in Section 7.

2. Workflow Time Management

2.1. A Workflow Model

A workflow basically describes a structured process consisting of activities and control flow dependencies between them (see also [18]). Figure 1 shows the graphical representation of elements using a directed graph. Activities, represented by rectangles, correspond to individual steps in a process. Dependencies, displayed as edges, determine the execution sequence of activities. At runtime an activity will instantiated and executed as soon as its predecessor node is finished. Control nodes, visualized as circles or ovals, represent workflow control structures. The label of a control node identifies its type. A circle with a dot depicts the start of a workflow and an empty circle depicts its end. A circle labeled with OS represents an or-split, which decides, based on a run time evaluated condition, which successor node will be executed next. In our model this node uses xor-semantics, as exactly one successor must be selected. The or-join, labeled with OJ, marks the end of an or-structure and allows the continuation as soon as one predecessor is finished. And-splits and and-joins, labeled with AS and AJ, are used to define parallel execution of several branches. All branches after the and-split will be executed in parallel and the and-join synchronizes all these branches, such that continuation is allowed if and only if all its predecessor nodes are finished. For our time management-related examinations on synchronous and asynchronous communication patterns we restrict the graph to be full-blocked [18] without cycles. Full-blocked workflows can be decomposed to basic building blocks (sequence, and, or) that may be nested but must not overlap. Workflows adhering to this concept eliminate the possibility of deadlocks, unwanted multiple instances, or livelocks during execution, but they also constrain the degree of modelling freedom. Therefore we plan to relax these restrictions in further publications (see also Section 7).

2.2. Time Management in a Nutshell

We claim that the control flow structure, the average or estimated durations of activities and overall process deadlines (maximum allowed workflow duration) can be utilized for time management purposes. The main concepts and benefits of workflow time management are best explained by means of a simple example. Figure 1 shows a workflow consisting of three activities A, B and C which must be sequentially executed. In many workflow systems activities have additional attributes holding expected execution durations, which are mainly used for simulation and process re-

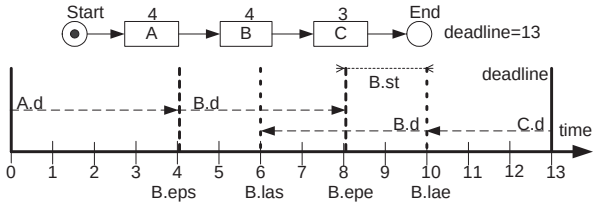

Figure 1. Time properties of activity B.

engineering purposes [18, 14]. In the example the durations are $A.d = 2$, $B.d = 3$ and $C.d = 5$ time units (e.g. hours). Assume that control nodes, like *Start* or *End*, usually have a duration of 0. Additionally an overall process deadline of $\delta = 12$ is defined, which must not be exceeded.

For predictive or proactive time management [17, 7] we have to calculate additional time information for each node at build time, which will later on be used at run time. The basic concepts are rooted in project planning methods like the Critical Path Method (CPM) or the Program Evaluation an Review Technique (PERT) [7]. They determine, among other things, expected and valid execution intervals for nodes. The interval is delimited by the earliest point in time an activity can start and the latest point in time it must end. The time line in Figure 1 shows these execution intervals for activity B. It uses a relative time, where 0 denotes the start time of the workflow. All other points in time are declared or calculated relative to this start time [12].

2.2.1. Sequences Figure 1 visualizes a workflow consisting of three activities executed in sequence. Explicit time properties are the estimated duration of activities in basic time units, which are $A.d = 4$, $B.d = 4$ and $C.d = 3$ and a deadline of $\delta = 13$, stating that the overall workflow execution must not exceed 13 time units. Based on this information four implicit time properties can be calculated for each node (e.g. for activity B): a) Considering the sum of durations of preceding activities the *earliest possible start* of activity B is $B.eps = 4$. b) The according *earliest possible end* is $B.epe = B.eps + B.d = 8$. Since we assume that there is no delay between activities (which may be relaxed using lower bounds, see 4.1) we can state that EP-values are calculated in a forward pass: $Start.eps = 0$, $Start.epe = Start.eps + Start.d$, $A.eps = Start.epe$, $A.epe = A.eps + A.d$, $B.eps = A.epe$ and so on. c) To take the deadline of 13, into account, the point of view has to be reversed, now starting from the end of the workflow. By subtracting the du-

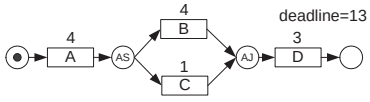

Figure 2. Workflow with parallel execution.

rations of succeeding activities from the deadline, the *latest allowed end* $B.lae$ of activity B is determined as $B.lae = 13 - 3 = 10$. That means if at process run time B ends at 10 it is still possible to reach the overall deadline of 13. d) Analogously the *latest allowed start time* is $B.las = B.lae - B.d = 6$. Therefore we can state that LA-values are calculated in backward pass: $End.lae = \delta$, $End.las = End.lae - End.d$, $C.lae = End.las$, $C.las = C.lae - C.d$, $B.lae = C.las$, and so on.

Implicit time properties can be utilized to specify a valid time interval for the execution of each activity. The EP-values may, during workflow execution, for instance be used to notify a participant about the expected start time of his future tasks or predict the overall workflow (rest) duration. The LA-values may be used to predict possible future deadline violations. If for example B would end at 11 it is likely that the deadline will be violated after finishing C. As 11 exceeds the LAE of B immediately evasive actions can be enacted in order to hold the deadline (e.g. assign additional manpower); the system predicts a deadline violation and can proactively avoid it. This saves costs and increases the quality of service.

2.2.2. Parallel Execution In workflows with parallel structures the longest path between and-split and and-join determines the duration and the intervals. In the example in Figure 2 the activities B and C will be executed in parallel. The and-join must wait for both activities to finish, therefore its EPS is determined by the maximum EPE of the predecessors: $B.epe = 8$ and $C.epe = 5$ thus $AJ.eps = 8$. For the backward calculation of LA-values the same applies for the and-split, the path to the end-node with the longest duration determines its LAE-value, therefore the minimum LAS-value of the successors is used: $B.las = 6$ and $C.las = 9$ thus $AS.lae = 6$.

2.2.3. Conditional Execution At build time the workflow modeler can not know which decisions will be made at or-splits during workflow execution. Additionally activity durations will differ from expected average values. Therefore some time management approaches (e.g. [10, 15]) already incorporated minimum and maximum bounds for durations, EP- and LA-values . In order to further increase the forecast precision we additionally introduced a probabilistic model which captures duration distributions and expected branching behavior for or-splits [8]. In this paper we use, for the sake of simplicity, average duration values. Nevertheless a bounding or probabilistic approach may be applied. For a detailed description of our probabilistic model and how to calculate EP- and LA-values we refer to [9].

3. Communication Patterns

3.1. Problem Statement

So far existing time management approaches are based on one assumption: activities, may they be atomic or complex, are interpreted as basic execution units which must be finished in order to proceed with workflow execution. But in real business processes one is often confronted with the situation that external services, applications or sub-processes may be started, using a blocking (synchronous) or non-blocking (asynchronous) communication model. These models are especially needed for inter-organizational business processes (IOBPs) which may nest several sub-processes or services from different organizations. To enable our time management algorithms to cope with these models it is necessary to examine possible structural scenarios and how they affect calculation of EP- and LA-values. Recent publications on web service communication and web service composition (e.g. [1, 16, 19]) already identified several basic synchronous and asynchronous communication patterns, which we utilized for our purposes. Note that the communication is realized by exchanging messages between processes. Therefore we explain some of the patterns in the context of the main process, which we focus on, that sends or receives messages to or from an external process.

3.2. Synchronous Patterns

In a synchronous or blocking model the requester waits for the response of the provider before it continues execution. The advantage of this model is its simplicity, as the process state does not change until the response has been received. Additionally it is much easier for modelers to comprehend the process structure hence debugging is eased. The obvious disadvantage is that blocking the execution of the main process, especially when long running external processes are involved, increases its execution duration tremendously.

SCP1 - Request/Reply is the basic model for synchronous communication. The main process sends

a request to the external process and blocks until a response is returned.

SCP2 - Solicit Response is an inverted SCP1. The external process sends a request to the main process, which sends a response after the request has been processed.

SCP3 - Synchronous Polling is an extension of SCP1. Again the main process sends a request and blocks. Subsequently it checks in defined intervals if the response has been returned, then it stops further polling-attempts and proceeds with execution.

3.3. Asynchronous Patterns

Although synchronous communication is appropriate in many situations it may be suboptimal when long-running external services or sub-processes are called. In an asynchronous or non-blocking model the requester sends a request to the provider and continues execution without delay. At a later point in time it receives a response (callback[1]) from the provider, which of course implies that the main process contains an activity which waits to receive this response. Asynchronous communication loosely couples sender and receiver. This accelerates process execution and compensates communication problems (e.g. network problems).

ACP1 - One-way (Message Passing) The main process sends a message to the external process and as it does not expect a reply it immediately continues with execution.

ACP2 - Notification is an inverted ACP1. The external process sends a message to the main process, without expecting a response.

ACP3 - Request/Response is the combination of ACP1 and ACP2. The main process sends a message to the external process. The external process processes the request, which may take some time, and sends a notification (callback) when it is finished. In the meantime the main process proceeds execution. Naturally a mechanism must exist which receives the notification.

ACP4 - Request/Multiple Response is an extended ACP3. The main process sends a message to the external process, which returns several notifications, each at a different point in time.

1 For details on the implementation of callbacks, for instance with correlation-ids, we refer to [1].

ACP5 - Publish/Subscribe is topic-based messaging. In a publish-subscribe system, senders label each message with the name of a topic ("publish"), rather than addressing it to specific recipients. The messaging system then sends the message to all subscribers that have asked to receive messages on that topic ("subscribe"). This form of asynchronous messaging is a far more scalable architecture than point-to-point alternatives. Senders need only concern themselves with creating the original message, and can leave the task of servicing recipients to the messaging infrastructure. It is a very loosely coupled architecture, in which senders often do not even know who their subscribers are.

ACP6 - Broadcast can be interpreted as an inverted ACP5. A message is send to the messaging infrastructure, which delivers a message to a selected or subscribed list of receivers. Each of the receivers decides how to further process this message. The sender does not expect any acknowledging responses. Similar to ACP5 broadcast may significantly increase network traffic, therefore it should be used carefully and for very specific purposes only.

ACP7 - Request/Response with polling is used if callbacks from external processes are not possible or allowed. The main process calls an external process using a blocking SCP1 Request/Reply, where the reply is an acknowledged-message from the receiver. Afterwards the main process continues. At a later point in time, when the main process needs the results, it calls the external process again; this time it uses SCP3 synchronous polling until it receives the response. Subsequently the main process may proceed. In order to find out if these operations correlate, which means that they belong to the same communication sequence, each message must be augmented with the same correlation-id, which is unique in both participating systems.

ACP8 - Request/Response with posting is needed for business-critical or confidential messages. It consists of a SCP1 request/reply followed by SCP2 solicit response. The main process sends a message to the external process, blocks and waits for the immediate acknowledging response. Later the external process sends an answer which must again be immediately acknowledged by the main-process. Again each message must be augmented with a correlation-id.

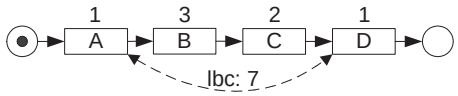

Figure 3. Workflow with a lower bound constraint.

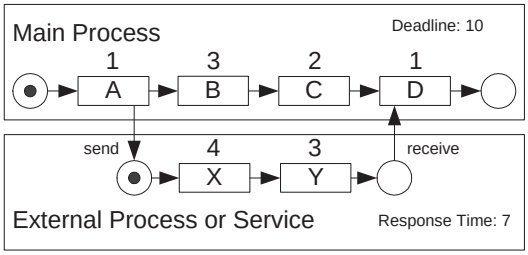

Figure 4. Calling an external process.

4. Prerequisites

4.1. Lower Bounds

Our solution is based on *lower bound constraints* [7]. A lower bound is defined between a source and a target activity which are not necessarily adjacent. It demands that when the source activity has been finished the target activity must not start until a defined time span has passed. Imagine a simple workflow with activities A, B, C and D which are to be executed in a sequence (see also Figure 3). Additionally a lower bound of $lbc = 7$ has been defined between A and D. Assume that after the execution of activity A its successors B and C have been executed and that the execution of B and C lasted 5 time units. Now D has to wait another 2 time units before it may start. For time management calculation the lower bound constraint behaves like a parallel execution path. The EPS of the "joining" activity D is determined by the sum of the duration of the longest path from the "splitting" activity A and $A.epe$, which results in $D.eps = 8$ (as the longest "path" is determined by the lower bound). Analogously for the backward calculation the longest path between A and D determines $A.lae = 2$. Please note that in this scenario a lower bound less than 5 hours (which is the sum of durations of B and D) would have had no effect on the execution duration of the workflow and the execution intervals of its activities, as in this case the longer regular path via B and D would determine the interval values.

4.2. External Processes

Now consider the processes in Figure 4. Activity A sends a message which activates an external process. The main process continues with its execution and, assuming that the execution duration of B and C again is 5 time units, has to wait for the results of the external process at activity D in order to proceed. One can easily see that the calculation of the duration of the main process and the execution intervals of its activities do not differ from the above explained variant using a lower bound. Since the external process may reside anywhere, and therefore its structure may be unknown, it must be treated as a black box. The only knowledge required is its expected execution duration or response time. Hence for time management calculations asynchronous external services can be treated like lower bound constraints. A required prerequisite is to connect the adhering sending activity A (exit-node) and the receiving activity D (entry-node) with a lower bound.

4.3. Response Time

In order to explain the integration of synchronous and asynchronous communication patterns into our time management calculation algorithms we have to clarify what we focus on. We are interested in the effects different types of communication patterns have on the duration of the main process and the delay they might produce between different activities of the very same. Please note that we calculate our time models for the main process only, therefore our primary interest in external services or processes is their response time. For time management it is not important to know how communication works in process automation systems. We assume that the system relies on a communication infrastructure which forwards request-messages from the main process and receives response-messages from external processes. This may be implemented using message queues [1].

Furthermore we do not differ between communication, execution or waiting time for external services; we use an overall response time which suffices for our needs. As messages are queued they may be retrieved before they are needed in the process. Therefore it is important to notice that the response time is measured as the time span between the outgoing message leaving the out-queue and the correlating ingoing message entering the in-queue[2]. Response times may be estimated

2 On details how messages are correlated in order to associate them to their appropriate process instances we refer to [1].

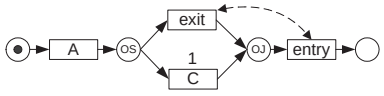

Figure 5. Unsound Workflow.

or gathered from empirical experience, for instance extracted from the message-log which holds information about prior communication-sequences or from a third party (see [11, 5]). The duration of entry and exit-nodes, which basically only send or receive messages to or from the messaging system, will usually be extremely low compared to other durations and response times, e.g. milliseconds vs. minutes, hours or even days. As their influence on other time properties is negligible assume them to be 0 if not stated otherwise.

4.4. Sound Asynchronous Processes

A workflow is sound if and only if, for any case, the process terminates properly, i.e., termination is guaranteed, there are no dangling references, and deadlock or livelock are absent. Although soundness has been defined on a petri-net based model it can analogously be applied on a graph based model. For further investigations about the soundness of workflows we refer to [2, 13]. Full-blocked workflows, which we concentrate on, will always be sound if synchronous communication patterns are used; this does not apply for asynchronous communication. E.g. the workflow in Figure 5 is full-blocked but not sound. In the upper conditional branch an external process is called and synchronized after the or-join. The problem is that during process execution the lower branch may be chosen which does not call the external process. This poses a problem as at the entry-node a message is expected which will never be received. To tackle this problem it is necessary to restrict the model to certain structures (see [19]) or to use tools to check the *soundness* of a process with asynchronous communication. In the following we assume that processes are sound.

5. Mapping of Patterns

This section describes how synchronous and asynchronous communication patterns can be represented in a workflow graph in order to apply time management calculations (on the main process). Figures 6, 7, 8 and 9 visualize these mappings.

5.1. Mapping of Synchronous Patterns

SCP1 - Request/Reply As the process blocks execution after sending a message, there must be no activities between the exit-node and the entry-node. The duration of the external process is defined by a lower bound. This pattern may also be represented by one *complex* activity [7]. A complex activity is used to abstract and hide process structures, which in this case are external. Its duration is defined by the overall duration of the hidden structure. Nevertheless we stick to the entry/exit-node notion as it may then be easily mapped to e.g. web service composition environments where request and reply are interpreted as atomic activities.

SCP2 - Solicit response The main process must wait for a message from an external process at the entry point in order to process the request. As this pattern assumes that no prior request has been sent from the main process, the duration of the entry-node must be set to an average waiting time. It is, unlike the response time, determined by the average time span the activity in the process has to wait for the incoming message, which may already be in the in-queue. This information can be extracted from the workflow log or estimated. The succeeding activity A stands as placeholder for an arbitrary number of activities or control flow structures. For our purposes it is not important that the calling external process is blocked during the execution of A. The exit-node sends the response to the requesting external process. For time management calculations it is not necessary to know that the entry-node and the exit-node adhere to the same communication sequence, therefore no connecting construct similar to a lower bound is needed. This pattern is unlikely to occur in a driving main process, which treats other processes as mere service providers.

SCP3 - Synchronous polling For time management calculations it is not important to know the number of polling attempts. The only information needed is again the response time, which is the time span between the original request and the last successful polling attempt. Therefore it can be treated like SCP1.

5.2. Mapping of Asynchronous Patterns

ACP1 - One Way The main process sends a message to the external process at the exit-node. As it

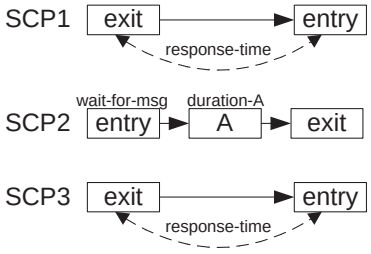

Figure 6. Mapping of Synchronous Patterns.

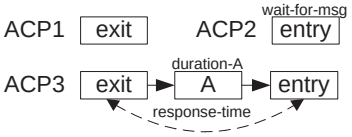

Figure 7. Mapping of Patterns ACP1-ACP3.

does not wait for a response there is no synchronization required.

ACP2 - Notification The main process must wait for message from an external process at the entry point. As this pattern assumes that no prior request has been sent from the main process, the duration of the entry-node must be set to an average waiting time. Like SCP2 this pattern is unlikely to occur in a driving main process, which treats other processes as mere service providers.

ACP3 - Request/Response As the process does not block execution after sending a message, there may be an arbitrary number of activities, represented by A, which are executed between the the exit and the entry-node. The response time of the external process is defined by lower bound between the exit and the entry-node. Time management calculations are to be conducted as described in Subsection 4.1.

ACP4 - Request/Multiple Response As the process does not block execution after sending a message, there may be an arbitrary number of activities, represented by A, after the exit-node. But now the main process expects multiple responses from the external process, which are received in multiple entry-nodes (see upper graph in Figure 8). Therefore a lower bound must be defined between the exit-node and each entry-node. Another possibility is visualized in the lower part of the figure. Here the messages are received in

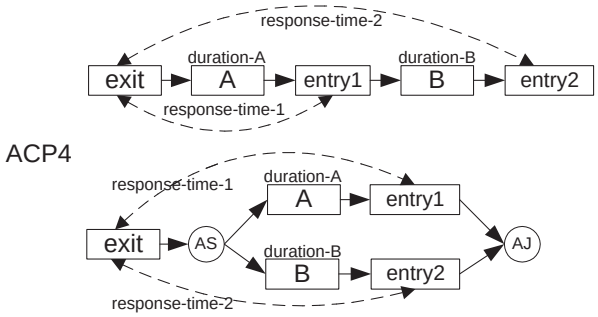

Figure 8. Mapping of Pattern ACP4.

parallel branches of the main process; although the structure is different the concept of applying lower bounds between each combination does not change (the same applies if the and-structure is exchanged by or-structure). The determination of a response time for each combination poses no problem if an entry-node expects a specific type of message, which can only be received by this entry-node. If some or even all entry-nodes are implemented to receive the same type of message an average response time may be used for each lower bound.

ACP5 - Publish/Subscribe This pattern can be mapped using a combination of other patterns. With SCP1 or ACP1 we model the subscription, depending on wether the main process has to wait for acknowledgement or not. ACP2 is used for each point in the main process where a publication message is expected. The waiting time for ACP2 is the average time span between two publications. Of course it makes sense to differ between waiting times for specific topics, in case it is known which message to expect at an entry-node. In our opinion this pattern is unlikely to appear in the course of a business process. It may probably be used to start a process when such a message is received, which has no influence on time management calculations as the starting time of the workflow is always initialized with 0.

ACP6 - Broadcast The broadcast is mapped using a single exit-node, like in ACP1, as the selection of receivers and sending multiple messages is conducted by the messaging system.

ACP7 - Request/Response with Polling The first request with acknowledgement is modelled using a SC1 request/reply. When the main process needs the results it starts polling the ex-

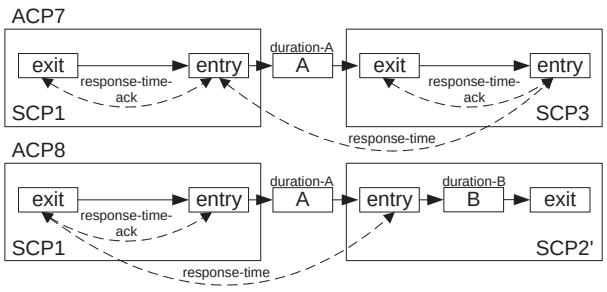

Figure 9. Mapping of Patterns ACP7-8.

ternal process, which is modelled using a SC3 request/reply with polling. The response time, representing the actual execution time needed by the external service to process the primary request, must be introduced as lower bound between the first and the second entry-node. At the first entry-node the main process is notified by the external process that the primary request arrived. At the second entry-node the final response is received by the main-process. Alternatively SCP1 and SCP3 may be modelled as complex activities, as they are blocking anyway. This simplifies the model a little bit, because the extra acknowledged-response times can simply be modelled as durations of the complex activities.

ACP8 - Request/Response with Posting Again the first request with acknowledgement is modelled using SC1 request/reply. After the execution of A, the main process has to wait for the response from the external process before it can return an acknowledged-message. Therefore SCP2', a solicit response variant, is used. The (complex) activity B may among other things for instance generate the acknowledged-message. The difference between SCP2' and the original SCP2 is that the entry node is not augmented with an average waiting time. The response message will be received when the external process finished its execution. Therefore it is sufficient to introduce a lower bound between the primary exit- and the second entry-node augmented with the response time, which represents the duration of the external process. Finally the main process will send an acknowledged-message at the second exit-node.

6. Application

The main purpose of time management is to express, process and interpret queries on different time properties. This information can be used to achieve several objectives, e.g.to make statements about the expected duration or the probability of violating time constraints (see also Section 2). Integration of time management for workflows with asynchronous communication patterns consists of the following major steps:

1. **Process modelling:** model the process according to the business needs and integrage selected external services or processes.
2. **Apply time management data:** enhance the workflow with time management-related data, like durations and deadlines. Additionally introduce lower bound constraints between entry and exit-nodes and augment them with appropriate response times. If the workflow has already been used for some time empirical information can be gathered, by extracting data from the workflow and messaging system logs.
3. **Calculate time management relevant information** according to Section 2.
4. **Utilize build time information.** The information calculated so far may for instance be used to predict the expected duration for a workflow or to check if deadlines are defined to tight.
5. **Process instantiation and run time.** For each workflow instance the actual *start time* (real calendar date and time) must be stored. At run time the time management component has to monitor the temporal status of all running workflow instances. To achieve this each running workflow instance must be synchronized with the time model. The time attributes of these activities must then be mapped to real calendar dates (by adding the instances start time). Now time predictive or proactive management methods may be applied in order to forecast and avoid eventually upcoming deadline violations as described in Section 2.

7. Conclusion and Future Work

The prediction and proactive avoidance of deadline violations decreases costs of processes and increases their quality of service. In this paper we examined basic asynchronous process communication patterns frequently used in inter-organizational processes, showed how they affect the execution and duration, and provided mappings for our workflow time management al-

gorithms and calculation operations in order to cope with the new requirements.

Currently we design and implement a Web Services-based time management framework. It will be used to examine complex asynchronous communication-sequences between multiple communicating processes and how they affect our time management algorithms. Another research objective is to extend the mappings for full-blocked and arbitrary cyclic structures. The integration of time management into process automation environments is subject of ongoing research.

References

[1] G. Alonso, F. Casati, H. Kuno, V. Machiraju. Web Services: Concepts, Architectures and Applications. Springer Verlag, ISBN 3-540-44008-9, 2005.

[2] W. M. P. van der Aalst and H. A. Reijers. Analysis of discrete-time stochastic petrinets. In *Statistica Neerlandica, Journal of the Netherlands Society for Statistics and Operations Research*, Volume 58 Issue 2, 2003.

[3] G. Baggio and J. Wainer and C. A. Ellis. Applying Scheduling Techniques to Minimize the Number of Late Jobs in Workflow Systems. In *Proc. of the 2004 ACM Symposium on Applied Computing (SAC)*. ACM Press, 2004.

[4] C. Combi and G. Pozzi. Temporal conceptual modelling of workflows. LNCS 2813. Springer, 2003.

[5] J. Cardoso and A. Sheth and J. Miller *Workflow Quality of Service.* Proceedings of the International Conference on Integration and Modeling Technology and International Enterprise Modeling Conference (IEIMT/IEMC'02), Kluwer Publishers, 2002.

[6] P. Dadam and M. Reichert. The adept wfms project at the university of ulm. In Proc. of the 1st European Workshop on Workflow and Process Management (WPM'98). Swiss Federal Institute of Technology (ETH), 1998.

[7] J. Eder and E. Panagos. Managing Time in Workflow Systems. Workflow Handbook 2001. Future Strategies Inc. Publ. in association with Workflow Management Coalition (WfMC), 2001.

[8] J. Eder and H. Pichler. Duration Histograms for Workflow Systems. In Proc. of the Conf. on Engineering Information Systems in the Internet Context 2002, Kluwer Academic Publishers, 2002.

[9] J. Eder and H. Pichler. Probabilistic Workflow Management. Technical report, Universitt Klagenfurt, Institut fr Informatik Systeme, 2005.

[10] J. Eder, E. Panagos, and M. Rabinovich. Time constraints in workflow systems. LNCS 1626. Springer, 1999.

[11] M. Gillmann, G. Weikum, and W. Wonner. Workflow management with service quality guarantees. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data.* ACM Press, 2002.

[12] H. Jasper and O. Zukunft. Time Issues in Advanced Workflow Management Applications of Active Databases. In *Proc. of the 1st International Workshop on Active and Real-Time Database Systems.* Workshops in Computing, 1995.

[13] B. Kiepuszewski, A. ter Hofstede, C. Bussler On Structured Workflow Modeling. In: Proceedings of the 12th Conference on Advanced Information Systems Engineering (CAISE). Stockholm, Sweden, June 2000.

[14] M. Laguna and J. Marklund Business Process Modeling, Simulation and Design. *ISBN 0-13-091519-X.* Pearson Prentice Hall, 2005.

[15] O. Marjanovic and M. Orlowska. On modeling and verification of temporal constraints in production workflows. Knowledge and Information Systems, 1(2), 1999.

[16] E. Newcomer. Understanding Web Services. Verlag: Addison-Wesley, ISBN 0-201-75081-3, 2002.

[17] E. Panagos and M. Rabinovich. Predictive workflow management. In Proc. of the 3rd Int. Workshop on Next Generation Information Technologies and Systems, Neve Ilan, ISRAEL, 1997.

[18] Workflow Process Definition Interface. *A Workflow Management Coalition Specification.* Document number WFMC-TC-1025, 2002.

[19] P. Wohed, W. M. P. van der Aalst, M. Dumas and A. H. M. ter Hofstede Pattern Based Analysis of BPEL4WS In: QUT Technical report, FIT-TR-2002-04. Queensland University of Technology, Brisbane, 2002.