Time Management in Workflow Systems

Johann Eder[1]
AT&T Labs – Research
Florham Park, NJ 07932
hans@research.att.com

Euthimios Panagos
AT&T Labs - Research
Florham Park, NJ 07932
thimios@research.att.com

Heinz Pozewaunig
Department of Informatics Systems
University of Klagenfurt, Austria
hepo@ifi.uni-klu.ac.at

Michael Rabinovich
AT&T Labs – Research
Florham Park, NJ 07932
misha@research.att.com

Abstract

Management of workflow processes is more than just enactment of process activities according to business rules. Time management functionality should be provided to control the lifecycle of processes. Time management should address planning of workflow executions in time, provide various estimates about activity execution durations, avoid violations of deadlines assigned to activities and the entire process, and react to deadline violations when they occur. In this paper we describe how time information can be captured in the workflow definition, and we propose a technique for calculating internal activity deadlines with the goal to meet the overall deadlines during process execution.

1 Introduction

Time plays an important role in the management of business processes. For instance, business process reengineering projects typically try to reduce turnaround times and improve process execution duration estimates in order to improve competitiveness. In addition, many business processes have time-related restrictions, including bounded execution durations for activities and sub-processes and absolute deadlines associated with activities and sub-processes.

---

[1] On leave from Department of Informatics Systems, University of Klagenfurt, Austria, eder@ifi.uni-klu.ac.at

Consequently, time management should be part of the core management functionality provided by workflow systems to control the lifecycle of business processes.

Even though existing workflow management systems offer sophisticated modeling tools to specify and analyze workflows, their time management functionality is rudimentary [PEL97, JZ96]. In particular, their time management functionality mainly addresses process simulations (to identify process bottlenecks, analyze the execution durations of activities, etc.), assignment of activity deadlines, and triggering of process-specific exception-handling activities (called escalations) when deadlines are missed during run-time [LR94, InC, Flo, Ult, CS96, CSE98, SAP97]. Moreover, few research activities about workflow and time management exist in the literature.

For process-centered organizations, time management is essential for the management of the processes themselves. Typically, time violations increase the cost of a business process because they lead to some form of exception handling. Therefore, a workflow management system should provide the necessary information about processes and their time requirements. In particular, the following requirements should be satisfied.

- Workflow modelers need means to represent time relevant aspects of business processes (e.g., duration of activities, time constraints, etc.) and means to check these timing conditions;

- Process managers need support to adjust time plans (e.g., extend deadlines) according to time constraints, and they need means to be warned of possible violation of time constraints so early that they can act accordingly to avoid time failures;

- Workflow participants need information about urgencies of the tasks assigned to them to manage their personal work plans;

- If time failures (i.e., deadline misses) occur, the workflow system should trigger exception handling to regain a consistent state of the workflow instance;

- Business process reengineers need information about the time consumption of workflow executions to improve business processes;

- Also controllers and quality managers need information about when and for how long activities of a workflow instance were performed.

The latter two aspects are usually provided by workflow systems via workflow documentation (also referred to as workflow history or workflow log) and monitoring interfaces. In this paper, we are mainly interested in the first three aspects. In particular, we address the following issues.

- Modeling of time at process build time to capture the available time information;
- Pro-active time calculation to raise alerts in case of potential future time violations;
- Time monitoring and deadline checking at run-time;
- Handling of time errors.

However, a word of caution ahead: the effectiveness of time management depends on the nature of the workflow, how detailed its description is, and whether there are external causes for time relevant events. For highly structured production workflows with only interorganizational events, time behavior and consumption of a workflow can be calculated precisely. In administrative workflows, which span different organizations with several external events (e.g., waiting for a costumer to reply), time calculations are rough. Nevertheless, management of time, time planning and controlling has to be done, and, to our experience, it is being done in current business processes. Typically, time planning relies on estimates based on experience. Time management during the execution of a process becomes even more important in such an environment, where time monitoring is essential for adjusting plans to avoid deadline misses. Our approach tries to model and represent the knowledge about time issues and make the best use of existing time knowledge during the execution of workflows.

The remainder of the paper is structured as follows. Section 2 describes the workflow model we use in this paper, addresses activity durations and deadlines, and covers the phases when time calculations take place. Section 3 presents the time calculations at process build time. Section 4 presents the time calculations at process instantiation and the actions taken during run-time. Section 5 offers a comparison with related work and, finally, Section 6 concludes our presentation.

2 Time Information in Workflow Schemas

In this section, we describe the workflow model we use in this paper, discuss assignment of activity and process deadlines, and present the various points during the lifecycle of a process where time calculations can take place.

2.1 Workflow model

We use a generic workflow model that employs the structures typically found in existing workflow models. In particular, a workflow is a collection of activities, agents, and dependencies between activities. Activities correspond to individual steps in a business process. Agents are responsible for the enactment of activities, and they may be software systems (e.g., database application programs) or humans (e.g., custumer representatives). Dependencies determine the execution sequence of activities and the data flow between these activities.

Activities can be executed sequentially or in parallel, and the possible parallel executions are: *unconditional* , i.e., all activities are executed, *conditional* , i.e., only the activities that satisfy a given condition are executed, and *alternative*, i.e., any activity among many alternative activities can be executed. In addition, a workflow may contain *optional* activities. Optional activities are typically executed during the execution of workflow instances. However, they may be dropped during the execution of a particular workflow instance when existing time constraints can only be satisfied by omitting the execution of some activities.

There is an important difference between conditional execution and alternative execution of activities. In the former case, the activity that is executed next depends on data and state that are generated during the execution of the workflow process instance. In the latter case, the activity that is executed next depends on policies and information that is shared by all instances of the same workflow process. This means that any alternative will lead to a correct workflow execution, and for time management, when the schedule is tight, the alternative with the shortest execution time can always be chosen.

In order to represent time information, we need to augment our workflow model with the following basic temporal types: time points, durations, and deadlines. For the sake of simplicity we assume that all time information is given in some basic time units. For applications, the time information has to be given in application specific temporal units. For build time and workflow schemas, time information is always given relative to the beginning of a workflow. For workflow instances, this time information is mapped to an actual calendar.

2.2 Execution Durations and Deadlines

Given a workflow schema, a workflow designer can assign execution durations and deadlines to individual activities and to the whole workflow process [LR94, InC, Flo]. Usually, execution durations correspond to estimated or projected activity execution times. In addition, many duration values may be specified for activities and used during simulation. These durations can be either calculated from past executions, or they can be assigned by specialists based on their experience and expectations. Typically, the most common duration values used include minimum, maximum, and most frequent execution times.

Activity and process deadlines, on the other hand, correspond to maximum allowable execution times for activities and processes, respectively. In the remainder of this paper, we refer to these deadlines as explicit deadlines. At process build time, these deadlines are specified relative to the beginning of the process. At process instantiation time, a calendar is used to convert all relative deadlines to absolute time points, modify the assigned deadlines, or assign new deadlines.

Deadlines do not have to be associated with every activity in a given workflow schema. In fact, no deadlines may be assigned to activities at all. However, it is beneficial to associate deadlines with activities. The most compelling reason for doing so is the monitoring of the execution progress of activities and processes containing these activities so that preemptive actions are taken when delays are developed. We present how these deadlines, called internal deadlines, are computed at process build and instantiation times and how we use them at run-time in later sections of this paper.

It is important to note that activity durations and deadlines may not be the same, which is how they are always treated by some of the existing workflow management systems. Distinguishing between the two is extremely beneficial for cases where the actions taken when a deadline is missed may have considerable costs associated with them (e.g., rollback of the entire process). In such cases, when an activity takes longer to execute than the duration assigned to it in the workflow schema, pre-emptive steps can be taken to assess deadline satisfiability, modify workflow parameters, and alert appropriate agent(s) and process manager(s).

2.3 Time Calculations

Given the activity durations and deadlines assigned in a workflow schema, time calculations are needed for computing optimistic and pessimistic start and finish times of activities within processes, available slack time for activities, updating existing deadlines, converting time information to absolute time points, and so on. In particular, the following three phases are used for these time calculations.

- At build time, time information (durations and external deadlines) is recorded, and general time information on the workflow type is computed.

Furthermore, several start and finish activity times are computed based on the above time information and the process structure. This information can be used by the process designer(s) to locate temporal bottlenecks and candidates for further optimization efforts;

- At process instantiation time, a process instance and its own time information are created, and the time information is mapped to absolute time points using a calendar. Furthermore, a deadline for the entire process is specified and internal activity deadlines are computed;

- At run-time, the execution progress of a process is monitored and activity execution times are recorded together with the decisions made at various conditional and alternative execution points. This information is subsequently used to adjust internal deadlines when activities are launched to worklists.

Typically, the assignment of external deadlines is an iterative process. The designer first assigns activity durations. The time calculations at process build time are then used to compute the duration of the whole process and the relative position of all activities. The designer can then choose to set external deadlines to some of the activities and recompute the time information. If external deadlines cannot be met, the designer might modify the workflow structure, or change the deadlines.

From this information, the duration of complex activities or sub-processes consisting of sequences, alternative, conditional, optional, and parallel executions can be calculated as we show in the sequel. However, for loops, we need additional information about the number of iterations. The designer can provide this information in three ways.

1. For a fixed number of iterations, the designer assigns exactly this number;

2. The designer can provide minimum, maximum, and mean number of iterations to calculate expectancy and variance as above, and use it for multiplication with the duration of the loop body;

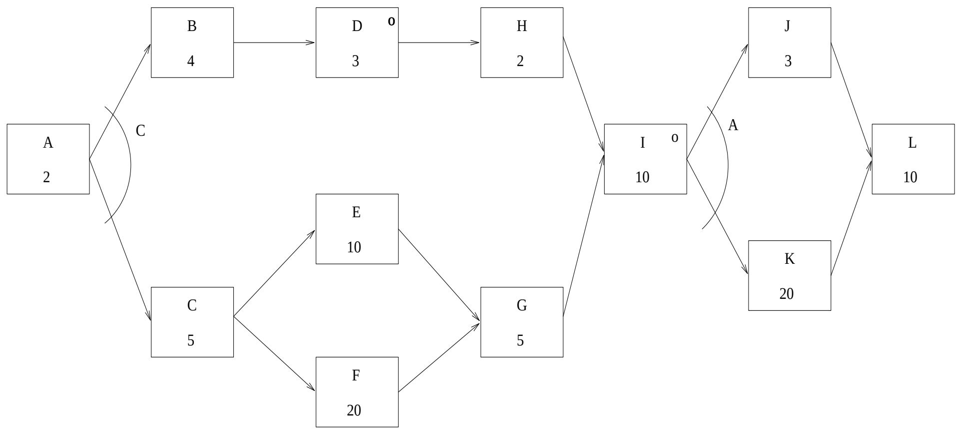

Figure 1: Example workflow process schema

3. The loop can be considered as a complex activity and the designer defines the duration of this whole complex activity.

In the following we will not consider loops in detail, but assume that they are treated as (complex) activities.

3 Time Management at Build Time

Typically, execution durations for activities are available at process build time. These durations are either assigned by the process modeler(s) based on various estimates, projections, and expectations or derived from past process executions. Figure 1 shows an example workflow schema having durations attached to activities. Activity A is the start activity and its duration is set to 2 time units. After A, there is a conditional split, and after activity C, there is an unconditional split (parallel execution). Activities D and I are optional. After I, there is an alternative split, i.e., either J or K will be executed (both are valid choices). Finally, L corresponds to the final activity of the workflow.

Using the workflow process schema and the durations assigned to the activities in the schema, we can calculate the relative start and end times for all activities (with respect to the beginning of the process), as well as the finish time of the entire process. In the remainder of the section, we describe how such calculations are carried out and how they can be used.

3.1 Timed Graph Construction

Every workflow activity A has a start and an end event associated with it. The start event denotes the start of the activity, while the end event denotes the completion of the activity. Assuming that any start-up costs are already incorporated in activity durations and, in addition, there exist no time constraints between activities (e.g., activity B can start 5 time units after activity A ends), the end event of an activity and the start event of all its successor activities are the same. When start-up costs exists or external timing constraints are present, dummy activities can be used to make the end event of an activity be the same as the start event of the next activity. Therefore, we only consider end events for the remainder of the paper.

In [PEL97], the authors used the PERT-net technique to formulate time constraints in workflow systems. Here, we extend their work and associate the following relative time information with the end event of an activity A: E_{BS}, E_{WS}, E_{BF}, E_{WF}, L_{BS}, L_{WS}, L_{BF}, *and* L_{WF}. In the above time information, E stands for the earliest point in time A may end, while L stands for the latest possible point in time A can finish to ensure minimal execution time for the entire process. Since conditional branches may require different execution times, we use B to denote the best case and W to denote the worst case. Finally, optional and alternative activity executions are "captured" by F and S. F corresponds to an execution where optional activities are not executed and the fastest alternative is always selected. S corresponds to an execution where all optional activities are executed and any alternative can be selected.

For example, E^{A}_{WS} corresponds to the earliest point in time A may end when in the preceding process execution the worst conditional branches were followed, all optional activities were executed, and the slowest alternatives were chosen. On the other hand, L^{A}_{WS} corresponds to the latest possible point in time activity A has to finish in order to minimize the execution of the entire process, assuming that the worst conditional branches will be followed, all optional activities will be executed, and any alternative can be selected in the remaining of the process.

Depending on the control dependencies between activities, E_{BF}, E_{BS}, E_{WF}, E_{WS}, L_{BF}, L_{BS}, L_{WF}, and L_{WS} are computed in the following way. First, a forward traversal of the workflow schema is required for computing E_{BF}, E_{BS}, E_{WF}, and E_{WS}. At the beginning of this traversal, the E-values of all activities without predecessors are set to 0. Next, a backward traversal of the workflow schema is required for computing L_{BF}, L_{BS}, L_{WF}, and L_{WS}. At the beginning of this traversal, the L-values of all activities

Forward Computation Procedure	
sequential	$E^{j}_{BF} = E^{i}_{BF} + d'(j),\ i \rightarrow j$ $E^{j}_{BS} = E^{i}_{BS} + d(j),\ i \rightarrow j$ $E^{j}_{WF} = E^{i}_{WF} + d'(j),\ i \rightarrow j$ $E^{j}_{WS} = E^{i}_{WS} + d(j),\ i \rightarrow j$
unconditional	$E^{j}_{BF} = \max\{E^{i}_{BF} + d'(j) \mid \forall i : i \rightarrow j\}$ $E^{j}_{BS} = \max\{E^{i}_{BS} + d(j) \mid \forall i : i \rightarrow j\}$ $E^{j}_{WF} = \max\{E^{i}_{WF} + d'(j) \mid \forall i : i \rightarrow j\}$ $E^{j}_{WS} = \max\{E^{i}_{WS} + d(j) \mid \forall i : i \rightarrow j\}$
conditional	$E^{j}_{BF} = \min\{E^{i}_{BF} + d'(j) \mid \forall i : i \rightarrow j\}$ $E^{j}_{BS} = \min\{E^{i}_{BS} + d(j) \mid \forall i : i \rightarrow j\}$ $E^{j}_{WF} = \max\{E^{i}_{WF} + d'(j) \mid \forall i : i \rightarrow j\}$ $E^{j}_{WS} = \max\{E^{i}_{WS} + d(j) \mid \forall i : i \rightarrow j\}$
alternative	$E^{j}_{BF} = \min\{E^{i}_{BF} + d'(j) \mid \forall i : i \rightarrow j\}$ $E^{j}_{BS} = \max\{E^{i}_{BS} + d(j) \mid \forall i : i \rightarrow j\}$ $E^{j}_{WF} = \min\{E^{i}_{WF} + d'(j) \mid \forall i : i \rightarrow j\}$ $E^{j}_{WS} = \max\{E^{i}_{WS} + d(j) \mid \forall i : i \rightarrow j\}$

Table 1: Computations of E_{BS}, E_{WS}, E_{BF}, E_{WF}

Backward Computation Procedure	
sequential	$L^{i}_{BF} = L^{j}_{BF} - d'(j),\ i \rightarrow j$ $L^{i}_{BS} = L^{j}_{BS} - d(j),\ i \rightarrow j$ $L^{i}_{WF} = L^{j}_{WF} - d'(j),\ i \rightarrow j$ $L^{i}_{WS} = L^{j}_{WS} - d(j),\ i \rightarrow j$
unconditional	$L^{i}_{BF} = \min\{L^{j}_{BF} - d'(j) \mid \forall i : i \rightarrow j\}$ $L^{i}_{BS} = \min\{L^{j}_{BS} - d(j) \mid \forall i : i \rightarrow j\}$ $L^{i}_{WF} = \min\{L^{j}_{WF} - d'(j) \mid \forall i : i \rightarrow j\}$ $L^{i}_{WS} = \min\{L^{j}_{WS} - d(j) \mid \forall i : i \rightarrow j\}$
conditional	$L^{i}_{BF} = \max\{L^{j}_{BF} - d'(j) \mid \forall i : i \rightarrow j\}$ $L^{i}_{BS} = \max\{L^{j}_{BS} - d(j) \mid \forall i : i \rightarrow j\}$ $L^{i}_{WF} = \min\{L^{j}_{WF} - d'(j) \mid \forall i : i \rightarrow j\}$ $L^{i}_{WS} = \min\{L^{j}_{WS} - d(j) \mid \forall i : i \rightarrow j\}$
alternative	$L^{i}_{BF} = \max\{L^{j}_{BF} + d'(j) \mid \forall i : i \rightarrow j\}$ $L^{i}_{BS} = \min\{L^{j}_{BS} + d(j) \mid \forall i : i \rightarrow j\}$ $L^{i}_{WF} = \max\{L^{j}_{WF} + d'(j) \mid \forall i : i \rightarrow j\}$ $L^{i}_{WS} = \min\{L^{j}_{WS} + d(j) \mid \forall i : i \rightarrow j\}$

Table 2: Computations of L_{BS}, L_{WS}, L_{BF}, L_{WF}

Activity	
duration	*opt.*
Ebf	Lbf
Ebs	Lbs
Ewf	Lwf
Ews	Lws

Figure 2: An activity node in the timed workflow graph

without successors are set to their corresponding E-values, unless external deadlines are assigned to these activities. In this case, all L-values are set to these deadlines, which are assumed to be relative to the beginning of the process.

If during the backward traversal external deadlines are assigned to activities with successors, the L-values of these activities are set to their respective deadlines when they are greater than these deadlines. Tables 1 and 2 illustrate the formulas used for the above computations. For clarity, we have omitted external deadlines from these formulas. The notation $i \rightarrow j$ is used to represent the fact that j is an immediate successor of i. In addition, $d(j)$ denotes the execution duration of activity j. If activity j is optional, $d'(j)$ is set to 0. Otherwise, $d'(j)$ is set to $d(j)$. It is easy to see, that the following invariants hold: $E_{BF} \leq E_{WF} \leq E_{WS}$, $E_{BF} \leq E_{BS} \leq E_{WS}$, and $E_{BS} \leq L_{BS}$.

For the calculations presented in tables 1 and 2, the execution duration of an activity can be set to the minimum, maximum, or average execution duration, or to a duration that corresponds to some percentage of the already executed process instances. These values can be computed from the log records generated by existing workflow systems. Alternatively, they can be assigned by process modelers and be part of the workflow-specific data. For the remainder of the paper, we assume that execution durations correspond to average execution times.

3.2 Interpretation of the Timed Graph

Figure 3 shows the result of calculating the E- and L-values for the workflow schema shown in Figure 1. Figure 2 shows the meaning of these values for each activity node. The most important information in this timed graph is the activity E-values and, in particular, the E-values of activity L. This is because *L* is the last activity to be executed and, thus, its termination

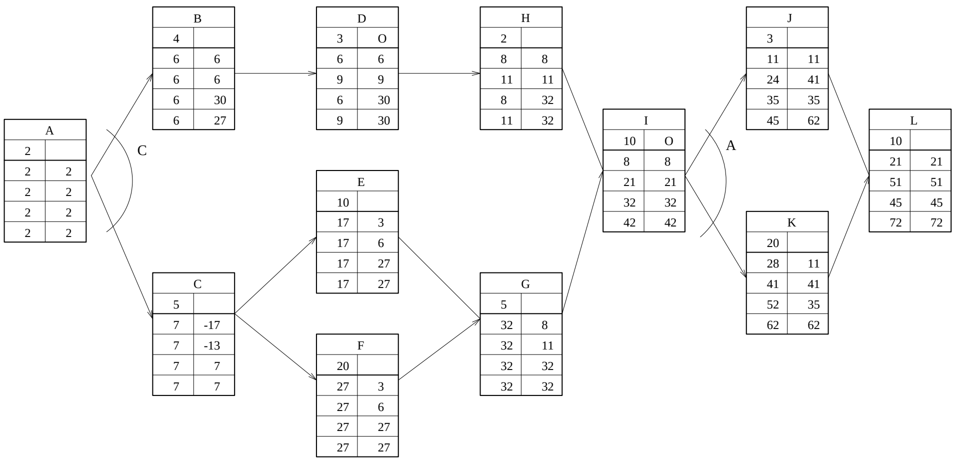

Figure 3: Workflow timed graph after the build-time calculations

time point corresponds to the termination time point for the entire process.

In particular, the earliest possible time for the entire workflow to terminate is 21, which corresponds to E^{L}_{BF} . This can happen when activities *D* and *I* are not executed, *J* is selected instead of *K*, and the conditional branch containing *B* is followed. On the other hand, if all optional activities are executed, the workflow can finish as early as 51 (i.e., E^{L}_{BS}) and as late as 72 (i.e., E^{L}_{WS}), depending on the alternative activity selection and the conditional branch followed.

Regarding the L-values of an activity, they indicate whether there is a path containing this activity that may lead to a time error at process execution. In particular, if all L-values of an activity are greater than their corresponding E-values, there exist execution paths containing this activity that are likely to avoid time violations. However, if there exist L-values that are less than their matching E-values, then there exist paths that may lead to time violations. For example, activity *C* has negative L_{BF} and L_{BS} values in Figure 3. Therefore, if *C* is executed at run-time and the deadline of the entire process is set to 21, the deadline will be violated.

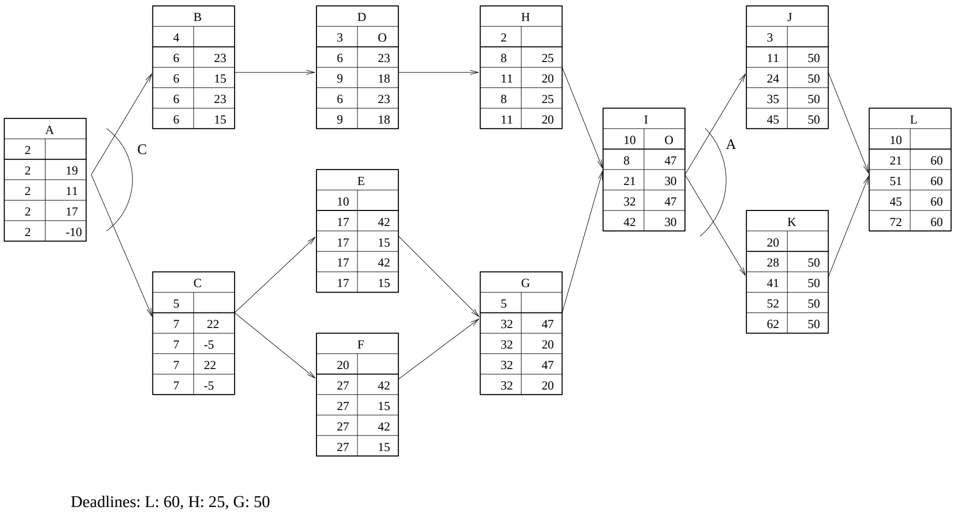

Figure 4: Example workflow process schema after process instantiation

4 Time Management at Run-Time

At run-time, relative time information contained in the timed graph created during build time is transformed into absolute time points, internal activity deadlines are monitored, and remedial actions are taken when deadlines are violated. In the remainder of the section, we address these issues in depth.

4.1 Time Fixing at Process Instantiation

At process instantiation time, an actual calendar is used to transform all relative time information to absolute time points. In addition, the timed graph may have to be recomputed if external, absolute deadlines are assigned to activities. However, the re-computation only affects the L-values and, hence, we only have to repeat the backward traversal of the workflow schema.

Figure 4 shows the workflow graph after deadlines for the activities *L*, *H*, and *G* have been given externally. When we look at the values for activity *A*, we can conclude that it is possible to meet all deadlines. However, if at conditional branches longer paths are followed, then it is necessary to skip optional activities or select faster alternatives.

If all L-values are negative, we cannot make it and we should raise a time exception. At this point, E-values are not really needed since the L-values are affected by the external deadlines. However, E-values could be used for performing agent load analysis. This can be done by checking the activities that are/will be assigned to an agent and the E-values for these activities. This topic, not discussed further in this paper, is subject of ongoing research to improve the forecast of delays in workflow executions.

4.2 Internal Deadline Calculation

During the execution of a workflow instance, the execution durations of activities may vary considerably from their average values. When the execution takes less than the average execution time, slack time becomes available. On the other hand, when the execution takes longer that the average execution time, slack time for future activities may be reduced. In addition to the slack time that is generated when activities take less time to finish, slack time may be available due to the following.

1. The deadline assigned to the entire workflow process is greater than the L-values of all activities that signal the end of the process, i.e., they have no successors;

2. Activities belonging to parallel branches may have different execution characteristics. Since the longest branch determines the execution of all parallel branches belonging to the same unconditional split point, shorter branches have slack available to them;

3. In conditional and alternative structures, slack is generated from the difference in the duration of different paths;

4. When an optional activity is not executed, its average execution time becomes the available slack for its successor activities.

Given the current absolute time point, *now*, the average duration of an activity A, and the L-values for this activity, we can assess the state of the workflow instance containing A with respect to its execution progress as follows.

- If *now* + *duration*(A) ≤ L^A_{WS}, the process is running smoothly and all deadlines will be met, given that the remaining activities take the expected execution time;

- If L^A_{WS} ≤ *now* + *duration*(A) ≤ L^A_{WF}, the process can still meet all deadlines. However, it might be necessary to drop some optional activities or chose faster alternatives;

- If *now* + *duration*(A) ≤ L^A_{BF}, there is still a chance that the workflow finishes in time. However, this depends on which conditional branches are followed;

- If L^A_{BF} ≤ *now* + *duration*(A), then it is only possible to meet the deadlines if the remaining activities finish faster than expected.

We should note that since externally defined deadlines are already taken into account during the construction of the timed graph, we do not have to consider these deadlines as long as L_{BF} can be met. If L_{BF} is missed, escalation is invoked. Based on the above observations, we can summarize the status of a workflow using the following states.

Green: We expect to finish the workflow in time without dropping any of the optional activities or changing the alternative selection policies;

Yellow: Although we may still be able to finish in time, we may have to eliminate some of the optional activities or select specific alternatives. In particular, before launching an optional activity, a decision has to be made whether the activity should be executed. Similarly, a decision needs to be made regarding the selection of the alternative activity to execute next. The rest of the activities are executed normally in this state;

Red: The threat of missing a deadline is great and a time error should be raised to trigger escalation actions.

To monitor the state at which a process instance is currently operating, we use two threshold values for each activity, L_{GY} and L_{YR} . L_{GY} signals the change from a green state to a yellow state. L_{YR} signals the change from a yellow state to a red state. Default values for these thresholds are set as follows: $L^A_{GY} = L^A_{WS}$

and $L^{A}_{YR} = L^{A}_{WF}$. These values are conservative choices, where no risk concerning alternative paths is taken. However, these threshold values should take into account the variance in activity durations, the proportion of best and worst cases, and the willingness to accept risks and, thus, are influenced by more information than is usually available in workflow systems. It is an important tuning knob for the time management system, and we believe that it should be the responsibility of a process manager to set these values and adjust them accordingly.

The above thresholds could be treated as internal activity deadlines (i.e., deadlines not assigned at build or instantiation times). In particular, the deadlines of all non-optional activities could be set to L_{YR}, while the deadlines of optional activities could be set to L_{GY}. Note that for optional activities we use L_{GY} because a decision has to be made before launching such activities, according to the discussion presented in the description of the yellow state.

However, it may be beneficial to assign different internal activity deadlines than the above threshold values when these deadlines can influence the sequence in which activities are selected from worklists and, therefore, influence when activities are executed. For instance, in workflow systems where the shortest deadline first scheduling policy is used by the engine or the workflow participants can choose the next activity to execute from their worklists, strict internal deadlines can be used to accelerate process execution and create slack that can be used to address unexpected delays and exceptions in the future. If these deadlines cannot be met, deadline extension can be granted based on the current state of the process and the available slack.

Possible alternatives for computing these internal deadlines are the no slack and proportional slack policies described in [PR97b]. In the former case, the internal deadline is set to the duration of the activity. In the later case, the duration is extended by a fraction of the available slack according to the proportion of the duration of the actual activity to the duration of the rest of the workflow. If the internal deadline does not influence the order in which activities are selected from worklists, which is the case when FIFO is used, the internal deadlines are not necessary and the threshold values introduced above can be used. This policy corresponds to the total slack policy presented in [PR97b]. For these worklist selection strategies, the deadline is only necessary to determine when an escalation has to be raised.

4.3 Handling Missed Deadlines

When a deadline is missed, a time failure is generated and escalation actions are taken. These escalation actions depend on the state of the workflow process (green, yellow or red), and some of the possible alternatives are the following.

- **Deadline Extension:** When an internal deadline is missed while the process is in either the green or the yellow state, the deadline may be extended. For non-optional activities, the upper bound for the new internal deadline is L_{YR} For optional activities, the upper bound for the new internal deadline is L_{YR}, according to the discussion presented in the previous section. Extending internal deadlines is helpful when the proportional slack or the no slack strategies are followed during deadline assignment.

- **Alternative Selection:** When the process is in the yellow state and its internal deadline is missed, besides extending its deadline (as in the previous case), the selection policy for future alternative activities may be changed to favor alternatives with faster execution times. Of course, the above is beneficial only when the process deadline can be met with these changes. The pre-emptive escalation work of [PR97a, PR97b] can be used for determining this.

- **Option Removal:** If no deadline extension can be granted and no alternative selection policy can be altered to preserve the process deadline, future optional activities can be eliminated. Actually, these optional activities are marked as dropped, and the decision to drop them is made when they are about to be scheduled for execution, as we discussed in the previous section.

- **Time Error:** If the process is in the red state, a timing exception has to be raised to escalate the problem. Here, recovery may be automatically invoked (ala [EL96]) or human interaction may be required to proceed. In the latter case, there are several options available to process managers in order to regain a valid workflow state. The workflow schema can be dynamically changed (e.g., by parallelizing sequential activities), activity priorities can be raised to speed up execution, or deadlines can be renegotiated.

The escalation strategy tries to avoid higher escalations as long as possible. The threshold values between the timing states defined above are again used for determining the escalation level. Pro-active actions like avoiding alternative branches or skipping optional activities are delayed as long as possible. When such pro-active means are taken, the timed graph has to be recomputed to reflect the changed workflow.

5 Related Work

Assignment of internal deadlines differs from dynamic workflow modification that is supported by some existing workflow products and research prototypes.

The latter is done to reflect changes in the model of the business process or a particular instance of the process. In contrast, our goal is to capture time information at build time, monitor process execution at run-time, and react to time failures without modifying the business process model. In this, it is somewhat similar to scheduling in real-time systems [LL73, AGM88, HSTR89]. However, real-time systems use deadlines for scheduling system components such as CPU and I/O. We view scheduling and internal deadline assignment and adjustment as complimentary mechanisms.

Our work is related to the work described in [MN95]. In [MN95], the authors presented priority-driven CPU scheduling algorithms for transactional workflows. Each workflow process consists of several sequential tasks. Each task is an ACID transaction having an average response time goal. The assignment of priorities is based on the performance of the tasks relative to their original response time goals. In contrast to these algorithms, our work does not concentrate on CPU scheduling. In addition, our algorithms are not restricted to transactional workflows, and they allow both sequential, conditional, parallel, alternative, and optional execution of tasks.

In [KGM93a, KGM93b], the authors studied the problem of how the deadline of a real-time activity is automatically translated to deadlines for all sequential and parallel sub-tasks constituting the activity. Each sub-task deadline is assigned just before the sub-task is submitted for execution, and the algorithms for deadline assignment assume that the earliest deadline first scheduling policy is used. While our work has similarities with the above work, there are several important differences. In particular, we treat alternative, conditional, and optional activities. Also, we offer techniques for building the timed graph at process build time and using the graph for arriving at a process deadline. Finally, our work supports the assignment of external deadlines to individual activities as well as to the entire process.

In [PR96, PR97a, PR97b], the authors proposed the use of static data (e.g., escalation costs), statistical data (e.g., average activity execution time and probability of executing a conditional activity), and run-time information (e.g., agent work-list length) for adjusting activity deadlines and estimate the remaining execution time for workflow instances. However, this work does not address time management at process build time, nor does it consider alternative activity executions and optional activities.

In [PEL97], the authors presented an extension to the net-diagram technique PERT to compute internal activity deadlines in the presence of sequential, alternative, and concurrent executions of activities. Under this technique, business analysts provide estimates of the best, worst, and median execution times for activities, and the β-distribution is used to compute activity execution times as well as shortest and longest process execution times. Having done that, time constraints are checked at build time and escalations are monitored at run-time.

Our work extends this work by providing a technique for handling optional activity executions, and addressing the computation of internal deadlines under various circumstances.

6 Conclusion

In this paper, we presented a method for incorporating time aspects into workflow management. The idea is to enrich a workflow specification by time information for activities, and to translate such a workflow description into a PERT-diagram that shows for each activity the time when the activity has to be at a specific state (e.g., started or finished) to satisfy the overall time constraints of the workflow. For these calculations, we extended the net diagram technique PERT to handle alternatives and optional activities in the process definition. At run-time, the PERT-diagram supports the workflow scheduler in finding optimized workflow executions.

An important advantage of our work in the explicit treatment of time during workflow definition and execution. In particular, our work enables process managers to plan workflows along the time dimension and to be alerted about potential time error, e.g., missed deadlines, early on so that they can take steps to avoid the conflicts or to escalate in order to minimize operational costs. The time information is also used to inform workflow users about time constraints about the activities in their worklists. This information allows users to make priority decisions between activities based on the urgency of activities in the global process and their deadlines to avoid time errors.

References

[AGM88] R. Abbott and H. Garcia-Molina. Scheduling real-time transactions: a performance evaluation. In *Proceedings of the 14th International Conference on Very Large Data Bases*, pages 1-12, Los Angeles, CA, 1988.

[CS96] CSE Systems. *Benutzerhandbuch V 4.1 Workflow*. CSE Systems, Computer & Software Engineering GmbH, Klagenfurt, Austria, 1996.

[CSE98] CSE Systems Homepage. http://www.csesys.co.at/, February 1998.

[EL96] Johann Eder and Walter Liebhart. Workflow recovery. *In First IFCIS International Conference on Cooperative Information Sy-*

stems (CoopIS 96), Brussels, Belgium, Jun 1996. IEEE Computer Society Press.

[Flo] TeamWare Flow. Collaborative workflow system for the way people work. P.O. Box 780, FIN-00101, Helsinki, Finland.

[HSTR89] J. Huang, J.A. Stankovic, D. Towsley, and K. Ramamritham. Experimental evaluation of real-time transaction processing. In *Proceeding of the 10th Real-Time Systems Symposium*, December 1989.

[InC] InConcert. Technical product overview. XSoft, a division of xerox. 3400 Hillview Avenue, Palo Alto, CA 94304. http://www.xsoft.com.

[JZ96] Heinrich Jasper and Olaf Zukunft. Zeitaspekte bei der Modellierung und Ausführung von Workflows. In S. Jablonski, H. Groiss, R.Kaschek, and W. Liebhart, editors, *Geschäftsprozeßmodellierung und Workflowsysteme,* volume 2 of *Proceedings Reihe der Informatik '96*, pages 109-119, Escherweg 2, 26121 Oldenburg, 1996.

[KGM93a] B. Kao and H. Garcia-Molina. Deadline assignment in a distributed soft real-time system. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 428-437, 1993.

[KGM93b] B. Kao and H. Garcia-Molina. Subtask deadline assignment for complex distributed soft real-time tasks. Technical Report 93-1491, Stanford University, 1993.

[LL73] C.L. Lin and J. Layland. Scheduling algorithms for multiprogramming in hard real-time environments. *Journal of the Association of Computing Machinery*, 20(1):46-61, January 1973.

[LR94] F. Leymann and D. Roller. Business process management with flowmark. In *Proceedings of the 39th IEEE Computer Society International Conference*, pages 230-233, San Francisco, California, February 1994. http://www.software.ibm.com/workgroup.

[MN95] M. Marazakis and C. Nikolaou. Towards adaptive scheduling of tasks in transactional workflows. In *Winter Simulation Conference*, Washington D.C., 1995.

[PEL97] H. Pozewaunig, J. Eder, and W. Liebhart. ePERT: Extending PERT for Workflow Management Systems. In *First East-European Symposium on Advances in Database and Information Systems ADBIS '97*, St. Petersburg, Russia, Sept. 1997.

[PR96] E. Panagos and M. Rabinovich. Escalations in workflow management systems. In *DART Workshop*, Rockville, Maryland, November 1996.

[PR97a] E. Panagos and M. Rabinovich. Predictive workflow management. In *Proceedings of the 3rd International Workshop on Next Generation Information Technologies and Systems*, Neve Ilan, ISRAEL, June 1997.

[PR97b] E. Panagos and M. Rabinovich. Reducing escalation-related costs in WFMSs. In A. Dogac et al., editor, *NATO Advanced Study Institue on Workflow Management Systems and Interoperability*. Springer, Istanbul, Turkey, August 1997.

[SAP97] SAP Walldorf, Germany. SAP *Business Workflow c Online-Help*, 1997. Part of the SAP System.

[Ult] Ultimus. Workflow suite. Business workflow automation. 4915 Waters Edge Dr., Suite 135, Raleigh, NC 27606. http://www.ultimus1.com.