published in: First East-European Symposium on Advances in Database and Information Systems ADBIS`97, pp. 217-224

ePERT: Extending PERT for Workflow Management Systems

Heinz Pozewaunig, Johann Eder, Walter Liebhart
Department of Informatics, University of Klagenfurt, Austria
email: {hepo, eder, walter}@ifi.uni-klu.ac.at

Abstract

Workflows management systems support the definition and execution of business processes. While business process reeingeneering tools use time information to simulate and optimize processes, the management of time is hardly supported in workflow systems. We introduce a concept for time management for workflow systems. It consists of calculating internal deadlines for all activities within a workflow, checking time constraints and monitoring time at run-time. For the calculation of internal deadlines we extend the net-diagram technique PERT to support the structures usually found in workflows. At run-time this time information is used to pro-acticely avoid time errors and reactively resolve time failures. The concept has been implemented in our prototype workflow management system Panta Rhei.

1 Introduction

The main purpose of workflow management system is to support the modeling and execution of business processes. Although most workflow management systems ([GHS95, Law97]) offer sophisticated modeling tools to specify and analyze workflows, they are insufficient in the handling of time issues [Poz96, JZ96]. But time is important for the management of processes and hence an integral part of business process modeling tools. Within these tools, time information mainly is used for simulation aims and to provide statistical data about durations, bottlenecks, etc. of business processes. Within workflow management time information primarily is important to manage the execution of workflows, to produce optimized execution plans and to avoid time failures.

Another aspect of explicit discussion of time is that the workflow designer has to deal not only with the structural aspects of the process design (for example, allowed sequences of execution chains, potential concurrent activities, normal and exceptional execution of activities, questions concerning the organization, ...), but also considers chronologically relevant elements.

Commercial workflow systems typically track whether deadlines or due dates are met ([Sch96]), but they are not trying to manage the situation before an error arises nor do they offer automated correction services. For process centered organizations the management of time is essential for the management of the processes themselves. Therefore, a workflow management system should be able to optimize the execution of business processes, to react on time failures adequately and to provide the process manager with the necessary information about processes, their time restrictions and their actual time requirements.

This paper is organized as follow: Section 2 gives a brief overview of related work within this area. In section 3 structural time aspects are defined and discussed. Section 4 presents a technique from operations research to integrate time into workflow management systems. This method is extended in order to meet typical workflow requirements. In section 5 the advantages of introducing time into workflow management systems is presented. Section 6 offers a short insight into the implementation of time concepts within a prototypical workflow management system and section 7 concludes this paper.

2 Related work

At present there are very few research activities concerning workflow and time. Of course, most commercial systems do have some limited abilities to handle time. For example, they allow the assignment of time duration or deadlines to activities and at runtime the system monitors these constraints. But as far as we know there is no system which supports optimization of processes as well as prediction and avoidance of failures based on time. In [JZ96] an ontology of time is developed for identifying time structures in workflow management systems. They propose the usage of an Event Condition Action (ECA-) model of an active database management system (DBMS) to represent time aspects within a workflow environment. Additionally, they discuss special scheduling aspects and basic failures with respect to time.

Temporal databases [Sno94] seem to be predestinated as a basic technology to manage the enactment of workflow instances. But they are not suitable for the explicit treatment of time at the level of workflows - in particular the modeling of time constraints and the time based scheduling. However, the underlying time calculus of temporal databases can be a solid basis for time based workflow management systems.

A promising formalism to model time information within the workflow context could be the area of temporal logics [Gal97]. But a first look onto this field led to our opinion

To appear in the First East-European Symposium on Advances in Databases and Information Systems - ADBIS'97, St. Petersburg, Russia.

that temporal logics are too restrictive - above all they do not offer adequate means to describe time quantities.

3 Structural time aspects

A *workflow* is the computational representation of business processes and can contain recursively other workflows. Workflows are defined by logical control structures, such as sequences, alternatives, loops, or choices. The basic element of every workflow is an *activity*, which represents an indivisible element of work.

Time issues can be expressed by time points, duration and intervals. In this context it is important to state, that the workflow designer must have the possibility to express user defined time metrics, i.e. for example to specify that a working day consists of 9 hours and must not be Saturday, Sunday or of course a holiday. Work days build working weeks and so on. This real calendar time metrics near to the universe of discourse must be considered during workflow scheduling and mapped to the internal time scale.

Time points and intervals can be assigned to activities and processes. After assigning a duration to every activity, we can determine the overall execution time of the whole process by finding the longest path through the workflow specification. This execution path is also called *critical path*, because it determines the amount of time, which is consumed by running the whole process. Every delay of any activity on the critical path causes also a delay of the overall workflow. All workflow structures do have a critical path. The trivial case is a sequential process, consisting *only* of one path - the critical path - whereas complex structures, containing concurrency or alternatives, do have other non-critical paths also.

Structural time dependencies are calculated from the structure of the workflow and the available time information. We call them structural because they only depend on the structure of a process and the duration times of each activity. In contrast to structural dependencies external ones are structural independent like defined deadlines (depending on a real calendar and not on the sequence of activities) of minimal or maximal duration times of activities. In this paper we only focus on structural time dependencies. Every start and end time of an activity is determined by the position of the activity within the workflow structure. A workflow element, residing at the critical path, has no slack time, but very narrowly defined start and end intervals. Every delay of such an activity is directly influencing the whole process duration. In contrast to this, an activity not residing at the critical path has more buffer time left, because its execution time does not immediately affect the execution time of the entire process. Only if the whole buffer time is consumed, this activity, with all its successors is getting critical.

Figure 1 shows a simple workflow specification. Activities are denoted by boxes. The name of each activity is a bold letter, figures in the lower right corner indicate their expected average execution time.

This example defines a workflow, beginning with an activity A with an estimated duration time of one time unit. The successor activities B and E are *and-connected* parallel activities which are executed concurrently. After completion of B, the activity C can be started, etc. But only if both activities D and F are finished, the last activity G is allowed to execute. The total execution time of the process is calculated by the maximum sum of all duration periods of all execution paths. The length of the critical path along

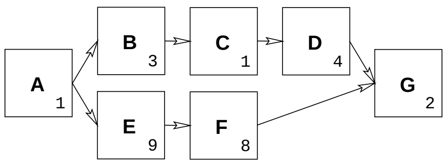

Figure 1: Simple workflow

the activities A, E F, and G is 20 time units. For all activities residing at the other paths, there is a common buffer of 9 time units, which can be consumed without delaying the minimum execution time of the workflow. Only if this amount is consumed, all other activities get critical.

In other words, there is a an structural time dependency determining the execution time of every activity. The upper bound for this time is fixed by the longest path through the process, which is also the shortest possible execution time for the process.

A specific problem appears if workflow activities can be executed alternatively. Every alternative in the workflow structure generates a new set of possible execution paths for an instanced process. But every possible execution path represents a certain specialized workflow definition with its own critical path, but different values. As, it is not efficient to keep track of all values, which are generated by computing all alternative branches of the workflow, we restrict our examinations to the best and worst case. Because we want to determine the shortest possible execution time of a workflow this is a legitimized limitation. Following values are calculated:

1. The *minimum duration* of the workflow, which is calculated as if for all alternatives the shortest possible path is chosen, and
2. the *maximum duration* of the workflow. This is the result of the assumption, that in every decision node the longest path is executed.

All this information can be generated automatically, if the workflow process designer provides the system with correct duration times for each activity.

4 Integration of structural time aspects into workflow management

4.1 An overview of PERT

From the area of operations research [GKP92, Phi86, EvF77] or production informatics [Gal93, DJOR90] several techniques are known to calculate and optimize parameters of process resources. By analyzing these techniques we have found several similarities between workflow modeling and execution on one side and project planning methods on the other side.

In our work we use a method of net optimization, PERT (*Program Evaluation and Review Technique*), to take the information, obtained by the workflow definition and calculate buffer times for every activity. There are many advantages in using this algorithm:

- As the PERT method is widely used in project management and, in a broader sense, you can look at a business processes also as a project, you can take PERT to model this kind of structure.

The main difference between project plans and workflow modeling is the number of instances, generated at execution time. In most cases projects do have only one object at run time, whereas workflows usually are instantiated many times. The modeling aspect is not concerned about that, but at run time it is much more difficult to determine differences between run time values and plan values.

- Time dependencies between parts of the workflow can be visualized immediately. You can identify some types of restriction just by looking at the diagram and important aspects are localized in a simple manner.
- Net diagrams are flexible and understandable. Adaption can be made easily and they are traceable for every one.
- The accordance between workflow description languages (WDLs) and PERT allows the transformation from one concept to another without great effort. It is also simple to interpret the values of the net diagram in terms of the workflow description [Poz96].

A PERT net diagram is a DAG (directed acyclic graph) with nodes i and directed edges (i, i'). Nodes represent end or begin states of activities. In the following, the notions *begin* and *end* are also called *states*. Every state s can be entered at E_s soonest and at L_s at the latest. Activities are represented by the edges between two states i and j with a duration time $d(i,j)$ assigned to them.

To ease the activity of correct estimating duration times, the designer must define three time values for every activity. The minimum period `a` defines the lower bound of the corresponding activity, although this execution duration is considers as very unlikely to happen. The maximum value `b` is the longest duration time the activity can consume and at last, the median `m` stands for most often needed duration time. These three values `a`, `b`, and `m` are input parameters for calculating the β-distribution by applying the equitation $\mu = \frac{a+4m+b}{6}$ and $\sigma^2 = \left(\frac{b-a}{6}\right)^2$. This type of propabilitity distribution is assigned only for one reason: to simplify the calculation of the expectation μ and variance σ^2. These parameters are used from now on under the assumption of a normal distribution of the duration time of activities. But, for the sake of clearness, in this paper we only investigate net diagrams with one value estimated. The variance is used in the implementation but not important for presenting the concepts.

Every net diagram has only *one* begin state and only *one* end state, but this means no restriction to the general case, because every process has a defined start and end to which a state can be associated.

In addition, dotted arcs in PERT diagrams represent dummy activities, used to synchronize different states.

Calculation of all values of the net is done in two phases. First, you have to forward calculate the values E of every state s. Starting with the value 0 of the first state of the diagram, every value E_j of a state j is determined by adding E_i of predecessor i to the duration $d(i,j)$ of activity (i,j). If there are two or more predecessors, the *maximum* value is chosen. After applying this calculation to every state, the duration of the whole workflow is determined by the value of the last state.

The next phase is the backward calculation of the latest allowed values L_i for every state. Starting at the last state every value L_i of a state i is the difference of $L_j - d(i,j)$, if j is the predecessor state of i. If there are more than one predecessor states, the *minimum* value is selected for L_i.

For detailed information how to generate and calculate net diagrams, refer to e. g. [Phi86, EvF77].

4.2 An extension to PERT

Several concepts needed to represent workflows are not covered by the "classical" PERT-diagram. One of the most important is the notion of *alternative* execution paths a process can take. There is no way to model this by the means of a net diagram, but to design a separate plan for each possible execution path. Because of the complexity of an average workflow structure and, as a consequence, the exploding number of separate net diagrams, this is in general not acceptable. Therefore, and, in addition, to allow the transformation of further control structures, like *iteration* or *choices*, a new syntactical element is introduced directly to the syntax of PERT, the *alternative* [Poz96]. From now on, we call net diagrams with alternatives *extended* PERT, for short ePERT.

Alternative paths in ePERT are indicated by an arc, spanning the arrows which represents the successing activities. As it is not affordable to remember all paths through the net, four values are stored for earliest and latest occurrence in the best and worst case of the execution. You can see the graphical representation of the new state type in figure 2.

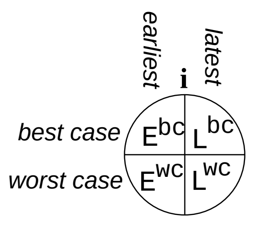

Figure 2: Extended ePERT node type

1. E_s^{bc} (*earliest best case of state s*) is the earliest point in time of the occurrence of state s, if s is part of the `shortest` execution path of the process.
2. L_s^{bc} (*latest best case of state s*) is the latest possible occurrence of state s to ensure minimal execution time.
3. E_s^{wc} (*earliest worst case of state s*) denotes the earliest moment in time of a state s which resides on the longest critical path.
4. L_s^{wc} (*latest worst case of state s*) is the `latest allowed occurrence` of state s. If a state enters at this specified time, the defined execution time of the whole workflow can be guaranteed only, if from now on the shortest path is chosen. If it is not possible to enter this state on time the whole process is late.

The state invariant

$$(E_s^{bc} \le L_s^{bc}) \wedge (E_s^{bc} \le E_s^{wc}) \wedge (E_s^{bc} \le L_s^{wc}) \wedge (E_s^{wc} \le L_s^{wc})$$

must hold for all values of an state S, to reflect the semantics of the best and worst case path. First of all, the condition of the formula has to be considered in the phase of backward calculation when synchronization is performed to consider the effects of different paths.

The interpretation of this kind of net diagram must be distinguished from the meaning of a standard PERT-diagram. ePERT values are the accumulation of all possible execution paths. Only the shortest and longest execution paths are tracked, all others are included in the interval between E_s and L_s. Also the rules for computing the E- and L-values change. According to the classical PERT method the calculation of the values of every state happens in two steps, *forward* and *backward* computation. See table 1 for the adapted calculation instructions for E and L values in cases of begin/end of sequences, alternative and concurrent ramifications. The following example is explaining this in more detail.

Example 1 *Alternative process execution*
In this example a workflow with seven activities is structured as shown in the ePERT-diagram of figure 3. Note that after the end of activity B the scheduler or the user has the opportunity to choose between successor D *or* E*, which depends on values obtained by the activity B. Also in the state after this ramification there exists the choice to select between activities* F *and* G.

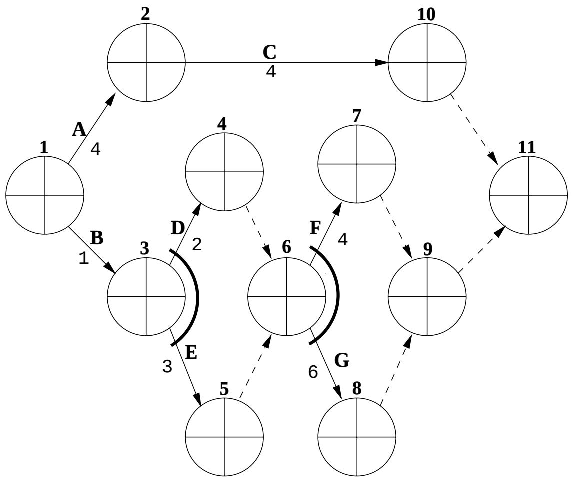

Figure 3: An ePERT of a workflow with alternatives

Ω

After applying the calculation instructions of table 1 the shortest and longest possible execution time for this workflow is determined (figure 4). The instructions are applied as follows. During the forward calculation the values E of the *end* of control structures are the point we want to regard. If the current state j is end of e. g. an alternative, the best case value E_j^{bc} is the result of the duration of the shortest of all paths from begin to the current node j, $E_j = \min(\{E_\varsigma + d(\varsigma, j)\})$, where ς is one of the direct predecessors of j. In contrast to the best case, the worst case must be the longest path from start to the current state, hence, E_j^{wc} is determined by the maximum of all incoming paths, $E_j = \max(\{E_\varsigma + d(\varsigma, j)\})$. To calculate all other E values you have to determine the correct end type of the structure and find the dedicated instruction.

To determine the latest possible values L during the phase of backward calculation, the second part of table 1 has to be considered. Now you have to pay attention to every *start* of a control structure. The instructions must be followed in the same way, e. g. if the current state i is start of a concurrent ramifications, in the best case you determine the value L_i^{bc} as minimum of all outgoing paths from the current state to the end of the process, $L_i = \min(\{L_\tau - d(i, \tau)\})$. The variable τ stands for a successor state of i which all must be investigated. The other cases can be found in the same way by determining the type of the start of the structure and applying the instruction for the best and the worst case.

The shortest critical path of the diagram of length 8 is the lower bound for the execution time of this workflow, if at all decision points the shortest process path is chosen. There are two interesting aspects, we want to stress.

1. The process is also allowed to e. g. start at a (relative) date of 2, without delaying the whole process. But in this case not all alternatives are still valid with respect to the deadline. This is the direct consequence of the fact that all the available slack time for the workflow is consumed already at start up time. Therefore, to meet the given deadline, from now on only the shortest path must be executed. Generally, the time value L_s^{wc} of every state s determines the point in time in which the whole estimated execution time can hold, if from now on the shortest path is chosen. The nearer the current time, the more restricted is the choice of possible paths. A following activity (i, j) with the assumed duration d is allowed to start at time t only, if $L_j^{wc} \geq t + d(i, j)$ holds. If this condition does not hold, a different activity as successor of (i, j) must be selected, if possible. This is allowed, if the following activity is the beginning of a choice control structure or an alternative control structure.

2. Another important fact is the need for *best case synchronization* in the phase of reverse calculating. If the current state i is start of a sequence immediately before the end of an alternative control structure (look at states 5 and 8 in figure 4), it must be checked if a synchronization should be performed. This is the case if the state invariant (refer to page 4) do not hold for this state i. Such states are not on the minimal critical path. If it is so, L_i must be set to E_i, because the latest allowed entry time of i can not be earlier than the earliest allowed entry time. If the invariant holds, L is simply the difference between the entry value of the successor state j and the duration between i and j.

This new planning technique allows a more general treatment of workflows. Control structures like *if*s, *loop*s and *choice*s of an workflow description language can be correctly mapped into a net diagram for the purpose of integrating structural time dependencies.

After building the ePERT-net from the time and structure data of the workflow description the scheduler can use this information at run time to generate correct schedules with respect to specified duration times. In comparing the actual time to the prescribed values E or L, the system can now identify delayed instances of a workflow timely and at every state. The enactment service of the workflow engine is in a position to trigger reactions before incorrect situations arise.

`Forward`	`bc`	`wc`
sequence	$E_j = E_i + d(i,j)$	$E_j = E_i + d(i,j)$
concurrency	$E_j = \max(\{E_\varsigma + d(\varsigma, j)\})$	$E_j = \max(\{E_\varsigma + d(\varsigma, j)\})$
alternative	$E_j = \min(\{E_\varsigma + d(\varsigma, j)\})$	$E_j = \max(\{E_\varsigma + d(\varsigma, j)\})$
	Annotation	
concurrency	$\forall$ predecessors ς of j	
alternative	$\forall$ predecessors ς of j	
`Reverse`	`bc`	`wc`
sequence	$L_i = \begin{cases} L_j - d(i,j), & \text{if } L_j - d(i,j) - E_i \geq 0 \\ E_i, & \text{else} \end{cases}$	$L_i = L_j - d(i,j)$
concurrency	$L_i = \min(\{L_\tau - d(i,\tau)\})$	$L_i = \min(\{L_\tau - d(i,\tau)\})$
alternative	$L_i = \min(\{L_\tau - d(i,\tau)\})$	$L_i = \max(\{L_\tau - d(i,\tau)\})$
	Annotation	
sequence	$L_i^{\sigma^2} = L_j^{\sigma^2} + d(i,j)^{\sigma^2}$	
concurrency	$\forall$ successors τ of i	
alternative	$\forall$ successors τ of i	

Table 1: Calculation instructions for ePERT

5 Usage of time information

The advantages obtained by introducing time into workflow management systems comprise two levels: the workflow modeling level and the workflow execution level.

5.1 Usage of time at modeling time

An important purpose to introduce time aspects into workflow managing systems is to ease the mapping of business process models into workflow management systems. As already mentioned, business process models deal with time information. If the workflow management system is capable to handle time information, the effort to rework previous made results is minimized. A further reason is to invite the workflow designer to use the potential to optimize the process structure with respect to time as early as possible. If he has to deal with explicit duration times of activities he has available very soon the parameters for optimization. Another important aspect is the early prevention of time inconsistencies. At compile time the system is able to identify static time failures and to inform the workflow designer of this fact. E.g. the minimum execution duration of all activities on the critical path is longer than the specified duration of the whole process.

5.2 Usage of time at runtime

The main advantage of introducing time into workflow modeling is the possibility to avoid time failures at run time and to use time information for optimizing workflow executions. The primary goal is to perform workflows effectively and efficiently under the restriction of scarce resources and the avoidance of time failures. *Structural time failures* lead to a violation of the overall process duration. According to our model, this happens, if the latest worst case of state s within any activity is violated. A violation (a delay) can be identified if the current time is greater than $L_s^{wc} + \sigma$ (under the assumption, that L_s^{wc} is already transformed to calendar based values). Some reasons for the violation of structural dependencies are: the agent needs more time to execute an activity as normal, an activity is not executed because the responsible agent is absent, a system failure occurred and hence the activity cannot be performed because the machine is down and so on.

To achieve the previous mentioned goals, the workflow scheduler must be extended by pro-active and re-active features.

5.2.1 Pro-active scheduling

Pro-active scheduling means to avoid time failures by selecting the optimal path through a workflow represented by an ePERT net. The time managing component of the workflow scheduler is in charge of examining all time values and time related rules specified by the workflow designer, linking them to run time instances of activities and calculating their absolute real time values. As soon as a time problem is foreseeable, the scheduler can react in several ways:

- *Routing:* At any decision point within the workflow the scheduler computes which of the possible paths lead to a time violation. Based on this information only such alternative paths are executed or offered to the workflow agent which are still on time.
- *Reassignment of activities:* If a system failure occurs then in some cases (e.g. a disconnected client node) the scheduler can bypass the failure by assigning the activity to an other user who is not concerned by the failure.
- *Non-assignment of uncritical activities:* If there is much buffer time left, the scheduler has the opportunity to defer the ongoing of a workflow in order to free resources (humans, information systems) which are needed for time critical activities.
- *Skipping non vital activities:* If there are specified non vital activities [EL96] within a workflow then these activities can be skipped in order to make up for lost time.
- *Changing the priority of activities:* The priority level of a delayed workflow can be increased in order to enable faster execution. Following this idea, the scheduler can be extended to generate dynamic priorities for activity execution depending on time restrictions. This approach is similar to aging algorithms known in the field of operating systems.

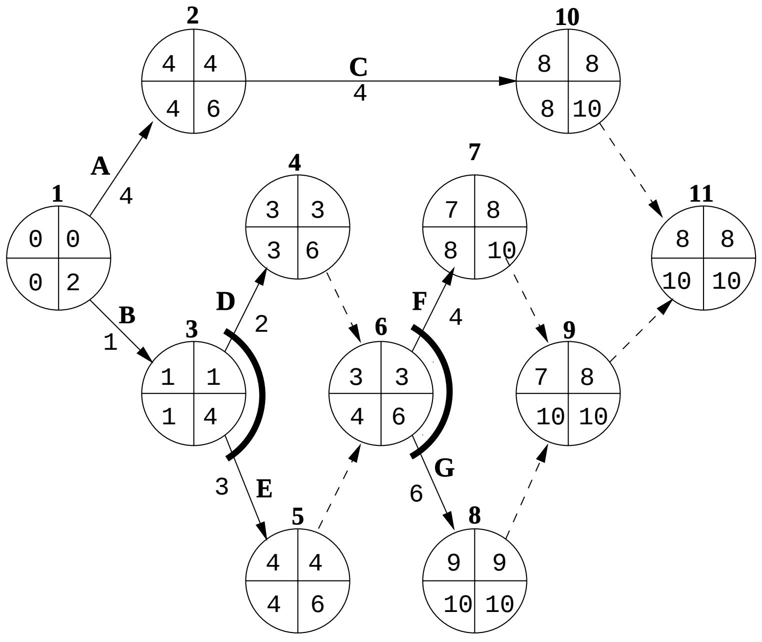

Figure 4: Complete ePERT-Diagram

Priority information can be passed on to the user through certain attributes of worklist items. So the user is able to arrange his work in accordance with overall requirements.

5.2.2 Re-active scheduling

After a time failure is detected, similar to other failures ([EL96]) some kind of reaction is necessary. We propose several reactions which are defined by the workflow designer during modeling time and used by a re-active scheduler at run time. Additionally, scheduling decisions may be based on human information at run time.

- *ignore:* The failure is ignored and workflow execution is continued. Of course, the failure is recorded and can later be used for analyzing processes.
- *warning:* The process manager and/or the responsible workflow agent is informed of the time failure. With an intelligent warning mechanism several warning levels can be realized (e.g. if a special activity is not executed immediately or within the next n time units then a time failure will occur).
- *recovery:* Particularly for highly automated processes (production workflows [GHS95]) the system should be able to react on time failures without any intervention of humans. To enable a failure-tolerant workflow execution the integration of transactional concepts into workflow modeling and execution is necessary (refer to [EL96, EL97, AAA$^+$96, KS95]. Necessary extensions are compensation activities (activities which "undo" the effects of already executed activities), alternative execution paths and so on. Based on these extensions possible reactions on a time failure are to ignore the failure, to skip future activities which are less important for the overall workflow in order to make-up-for time, or to go some steps back in the workflow execution and to continue forward execution on an alternative, "shorter" path.
- *time_activity:* A special activity is assigned to a workflow agent in order to inform him of the time failure and to support him with additional information concerning the time failure and necessary reactions.
- *cancel:* The workflow is cancelled because the intended goal of the business process cannot be reached any longer.

5.2.3 Workflow controlling

One other important advantage of the usage of time aware workflow management system is the instant access to all state information of running processes. This opens many interesting aspects.

Activities and workflows which might run late can be identified immediately by observing the expectancy and standard derivation of every activity and compare it to the current run time value. Every late activity can be easily identified. Hot spots and bottle necks can be located without additional expense and in doing so, a first step towards process controlling is made.

Not only the current state of the workflow management system should be considered, but also past values must be studied to get full advantage of the technology. Further process optimization can be achieved easily by investigating the statistical material computed from the audit data (refer to the interface 5 description of the Workflow Management Coalition ([Hol95]). As a further advantage, probabilistic statements of the behavior of workflows can be issued. So questions like the following can be answered:

- What is the probability of the delay of a certain workflow?

- How likely is the change of an uncritical activity to a critical one?

In particular the log of workflow execution can be used to improve the time estimates of the workflow designer. By analyzing execution paths, execution times and time failures the original workflow specifications can be adapted to new requirements.

6 Implementation of time concepts in *Panta Rhei*

Panta Rhei is a prototypical workflow management system, implemented at the Department of Informatics, University of Klagenfurt [EGL97]. The basic concepts of the system are:

- All process information is mapped to the data space. Process schema as well as process instance data are stored in the database. Thus processes can be manipulated as data which offers enormous flexibility.
- Case data and process data are represented in a uniform way. The form metaphor is used for representing data. From a programming point of view a form is a data type. From an implementation perspective, a form is a view on the database.
- Processes are defined in a highly generic *W*orkflow *D*efinition *L*anguage (WDL) with the usual control structures. The only variables in this language are of type form, such that all data used in processes is stored in the database.
- All concepts are first class citizens in the workflow language. Agents, forms, activities and even process definitions and control structures can be taken from form items.

Besides these basic concepts there are several other important features:
The system offers several possibilities in order to guarantee a consistent and reliable execution of business processes, i.e. the handling of failures and exceptions. The *advanced transaction* system of Panta Rhei tries to automate the recovery from failures and exceptions as much as possible. The transaction system is both, a runtime feature and a build time feature. As build-time feature it offers additional possibilities for a workflow designer to specify valid processes. The workflow schema can be enriched by additional information which is used in case of an exceptional situation. As runtime feature it (automatically) recovers from (semantical) failures, compensates activities and triggers alternative executions.
Another important feature of Panta Rhei is its open architecture.The interface of a user to the system is completely integrated in a web browser. This makes the system open such that anybody with access to the WWW can participate in a workflow. This feature is important, if one realizes that workflows are frequently started as a reaction to some request from outside of an organization (e.g. an order is received). Second, the communication between the workflow system and other systems is realized through the exchange of forms. Panta Rhei does not make any assumptions on client applications and remote systems but that they are able to receive forms and return or send forms. All process specific information is represented in forms. So there is no need for complex suite of APIs to make workflow systems collaborate.

The concepts of time are included into Panta Rhei in the following manner:

- By compiling the workflow description language, a meta structure is generated automatically, which contains a consistent ePERT diagram. This is checked against defined external time constraints and is used by the scheduler.
- For each activity type the designer must specify a possible reaction in case of a time failure and three values:

  a_t is the shortest allowed duration time of activity t,

  b_t is the longest estimated duration time for activity t,

  m_t is the most often occurring value for the duration of t.

  The following syntax diagram is part of the WDL of Panta Rhei and describes the structure of an activity (task):

```
<TaskDefinition>::= <TaskHead><TaskDecl>
<TaskHead>      ::= task <Name>(<ArgList>)
<TaskDecl>      ::= [desc <String>]
                    [postcondition <String>]
                    [procudure <String>]
                    <TimeEstimation> ...
<TimeEstimation> ::= minduration <DurationTime>
                     avgduration <DurationTime>
                     maxduration <DurationTime>
<DurationTime> ::= <Number> (minutes|hours|days)
```

- Within every process definition a *reaction* is defined, to specify the behavior of the process in case of time errors.
- For every iteration, there are three specification possibilities.
  1. The average iteration number has to be defined.
  2. Estimates for the minimum, maximum and average number of iterations are specified. With this three values the average duration times of every activity within the iteration control structure can be calculated.
  3. The average duration for the loop is defined. Here, the workflow designer has also to estimate three values, `min`, `max`, and `average time` for best calculation results.

A process described in Panta Rhei looks like the following example.

Example 2 *Workflow specified in Panta Rhei*
An insurance offer must be signed by the customer. If there is no signature on the offer, the custumer is asked repeatedly to do so. Experiences show, that there are no more requests for this activity after two trials, either because the customer signed the contract or he never wanted so. This fact is reflected by the iteration specification within the loop. All these specifications are compiled directly to an ePERT diagramm and are used to determine the overall process duration. Hence, a precondition is the duration specification for the task requestSignature.

```
TASK requestSig(of INOUT offerType)
  MINDURATION 1 HOURS
  AVGDURATION 2 DAYS
  MAXDURATION 7 DAYS
  ...

PROCESS insurance(offer INOUT offerType)
  DESCR "A clipped process to administrate
         an insurance offering";
  SUBJ IS offer.subject;

  ...
  REACTION WARNING;
BEGIN
  WHILE offer.signature == "" DO
    [minit 0, avgit 0, maxit 2]
    officialInCharge requestSig (offer);
  END;
  ...
END;
```

Ω

All this information entered at the design phase with the help of WDL is used to compile an ePERT-definition from the workflow specification. The resulting diagram is a pattern for each instance of this workflow structure. The workflow enactment service can easily check the status of every running process and issues error indicators if a condition is not met. The scheduler uses this data to dynamically generate priorities for each workflow instance.

7 Conclusion

We presented a concept for the treatment of some time issues in workflow management systems. A technique from operations research was extended to accommodate the structures typically found in workflow systems. This concept was included into our prototype workflow management system *Panta Rhei*. Our approach of integrating time into workflow management systems offers several advantages: First of all, the mapping of business process models into workflow management systems is simplified since we are now able to handle time information. Second, already at modeling time inconsistent time specifications can be identified and parameters for process optimization are available. At run time, a pro-active and re-active scheduler is able to find optimal workflow executions in terms of time and resources and, in particular to avoid time failures. The concept so contributes to the management of processes and resources in workflow management systems.

References

[AAA$^+$96] G. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Günthör, and C. Mohan. Advanced transaction models in workflow contexts. In *Proc. of 12th IEEE Intl. Conference on Data Engineering*, New Orleans, LA, 1996.

[DJOR90] C. Dorninger, O. Janschek, E. Olearczick, and H. Röhrenbacher. *PPS Produktionsplanung und -steuerung - Konzepte, Methoden und Kritik.* Ueberreuter, Wien, 1990.

[EGL97] J. Eder, H. Groiss, and W. Liebhart. The workflow management system Panta Rhei. In *NATO Advanced Study Institute (to appear)*, Istanbul, Turkey, 1997. Springer.

[EL96] J. Eder and W. Liebhart. Workflow recovery. In *1st IFCIS Int. Conference on Cooperative Information Systems*, Brussels, Belgium, June 1996. IEEE Computer Society Press.

[EL97] J. Eder and W. Liebhart. Workflow transactions. In P. Lawrence, editor, *Workflow Handbook 1997.* John Wiley, 1997.

[EvF77] Horst A. Eiselt and Helmut von Frajer. *Operations Research Handbook - Standard Algorithms and Methods with Examples.* Walter de Gruyter, Berlin, 1977.

[Gal93] R. Les Galloway. *Principles of operations management.* Routledge, London, New York, 1993.

[Gal97] A. Galton. *Temporal Logics and their applications.* Academic Press, London, 1997.

[GHS95] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation. In A. Elmagarmid, editor, *Distributed and Parallel Databases*, volume 3. Kluwer Academic Pub., Boston, 1995.

[GKP92] F. Glover, D. Klingman, and N. V. Phillips. *Network Models in Optimization and Their Applications in Practice.* John Wiley & Sons, Inc, New York, Chister, et. al., 1992.

[Hol95] D. Hollingsworth. The workflow reference model. Draft 1.1 TC00-1003, Workflow Management Coalition, July 1995.

[JZ96] H. Jasper and O. Zukunft. Zeitaspekte bei der Modellierung und Ausfuehrung von Workflows. In S. Jablonski, H. Groiss, R. Kaschek, and W. Liebhart, editors, *Geschaeftsprozeßmodellierung und Workflowsysteme*, volume 2 of *Proc. Reihe der Informatik '96*, pages 109 – 119, Oldenburg, 1996.

[KS95] N. Krishnakumar and A. Sheth. Managing heterogeneous multi-system tasks to support enterprise-wide operations. In A. Elmagarmid, editor, *Distributed and Parallel Databases*, volume 3, 1995.

[Law97] P. Lawrence, editor. *Workflow Handbook 1997.* John Wiley, 1997.

[Phi86] S. Philipose. *Operations Research - A Practical Approach.* Tata McGraw-Hill, New Delhi, New York, 1986.

[Poz96] H. Pozewaunig. Behandlung von Zeit in Workflow-Managementsystemen - Modellierung und Integration. Master's thesis, University of Klagenfurt, 1996.

[Sch96] G. Schmidt. Scheduling models for workflow management. In B. Scholz-Reiter and E. Stickel, editors, *Business Process Modelling.* Springer, 1996.

[Sno94] R.T. Snodgrass et al. Tsql2 language specification. *SIGMOD Record*, 23(1):65 – 86, 1994.