Eder J. & Saringer M. (2003*). Personal Schedules for Workflow Systems*. Proceedings of the International Conference on Business Process Management (BPM 2003), June 26-27, Eindhoven, The Netherlands, LNCS 2678, pp. 216-231.

Personal Schedules for Workflow Systems

Johann Eder and Michael Ninaus and Horst Pichler

University of Klagenfurt
Institute for Informatics-Systems
{eder@isys.uni-klu.ac.at, mninaus@edu.uni-klu.ac.at, horst@isys.uni-klu.ac.at}

Abstract. Personal schedules allow workflow participants to improve their performance of activity executions. Participants are no longer surprised by the entries in their work-lists but receive advance information about (potential) future activity assignments, allowing better possibilities for work-planning. The personal schedule system is based on a probabilistic workflow time management system using duration histograms. A personal schedule collects future activity assignments together with their probability and their timing requirements and allows to analyze the workload of a participant and to support the scheduling of activities with the goal of reduced turn-around times and reduced number of violations of temporal constraints.

Keywords: workflow management system, time plans, temporal constraints, scheduling, personal scheduling

1 Introduction

In the execution of workflows, workflow participants are typically "surprised" by the activities they should perform, surprised in the sense that they find these activities in their to-do-lists when these activities are ready, i.e. all preceding activities are finished. Information about upcoming activities would be much earlier available in the workflow system. For an example, when the first activity of a sequence is ready, the succeeding activities will be ready soon. Current workflow systems do not make use of this information and do not forward this information to the participants depriving them of the possibility of planning their work ahead. For administrative processes, in particular in settings where workflow participants have dispositive competencies and have to manage their schedules this strategy leads to suboptimal results. The main shortcomings are the following:

- longer retention period of activities in work-lists before they are taken up
- longer turnaround time
- no workload balancing
- considerable number of deadline violations

Reasons for this situation might be that workflow systems typically do not compute schedules due to the partial knowledge they have about the execution of their processes, the impossibility to know the actual flow at decision points at process instantiation time and typical variance in the execution of individual tasks. Nevertheless, several research proposals and prototypical implementations (e.g. [5, 12, 2, 15]) propose improvements. The main idea of these efforts is to make best use of the available information. The approach presented in this paper follows these directions. We aim at improving the planning situation of workflow participants by making them information about their future workload available early and provide them with means for digesting and using this information.

For an improvement of the sketched planning situation we propose personal schedules. A personal schedules contains not only the "ready" activities but also those which might become ready in near future. When in a workflow instance the first activity is assigned to the work-list of a participant, the actors of all the succeeding activities also receive information about task will be assigned to them soon, together with the probability of the assignment and with temporal characteristics and constraints.

Such information about future activity assignments might arrive at a particular participant from several workflow systems. In practice we find that people are involved in workflows managed by different workflow systems and have additional responsibility not supported by any workflow system. So general central scheduling approaches like they have been developed an are used in production management and ERP systems cannot not be used in such environments. Personal schedules are primarily intended as a support structure for individuals organizing the execution of their workload in time striving for improved performance.

Our personal schedule approach is based on a probabilistic time management system [4] which uses duration histograms to express the uncertainty of workflow time plans stemming from variations in activity durations and from the unpredictability of control decisions.

The rest of the paper is organized as follows: In section 2 we introduce our workflow model and all basic definitions used in the following, and we summarize our probabilistic time management concept. In section 3 we show how to compute probabilistic time plans for workflows. In section 4 we present the concept of personal schedules, and in section 5 the algorithms for the computation of personal schedules and their manipulation is covered. In section 6 we discuss some applications and draw some conclusions.

2 Workflow Model and Time Histograms

2.1 Basic definitions and assumptions

We define a rather generic workflow model that we use in the rest of this paper and introduce the *Probabilistic Time Plan* on base of a *Probabilistic Timed Graph* as a structure for representing probabilistic information about the duration of activities and processes.

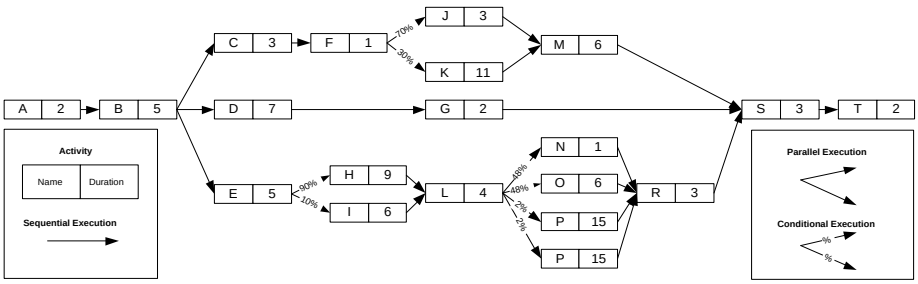

Fig. 1. Example workflow process schema

Essentially, a workflow is a collection of *activities,* and *dependencies* between activities. Activities correspond to individual steps in a business process. Dependencies determine the execution sequence of activities and the data flow between them. Activities can be executed sequentially, in parallel (and-splits) or conditional (or-splits). Consequently, a workflow can be represented by a directed acyclic graph, where nodes correspond to activities and edges correspond to dependencies between activities. Additionally the model contains the expected duration for each activity and statistically weighted values for each conditional branch, defined by administrator estimations or average values from past executions.

Figure 1 shows an example workflow schema. The 3 routes after B (*and-split*) will be processed concurrently. The workflow continues with S (*and-join*) not until M, G and R are finished. An example for a conditional execution is E. In this case only one specific path after E (*or-split*) will be chosen, which results in 2 different execution-routes from E to L (*or-join*) and 8 routes between E and T. H will be executed after E in 9 out of 10 cases and O will be executed after L in 4.8 out of 10 cases. Thus the probability that a workflow-instance will execute the path from E via H and O to T is $0.9 \cdot 0.48 = 0.432$.

2.2 Time Histograms

To represent probabilistic values of possible start-times and end-times of workflows and (complex) activities we enhance the idea of *Duration Histograms* [4] to *Start Time Histograms* and *End Time Histograms* which will subsequently be merged into the *Probabilistic Timed Graph*.

Activities can have multiple start-times due to conditional branches depending on the execution path. To represent these non-scalar values we use time-probability tuples. E.g., the path from A to E is sequential and unambiguously determined, therefore E holds only one start-time tuple $(1.0, 7)$ where the second value specifies the start-time and the first value specifies the according execution-probability. The start-time is calculated by adding the duration of all predecessor-activities. The activity L can be reached on two routes (via H

and via I) each with its own execution probability, thus L holds two different start-time tuples $\{(0.1, 18), (0.9, 21)\}$.

For end-time calculations we conform to the ePERT-approach [17]. At first we have to change our point of view on the workflow-process and start from the last activity T, initialized with a given deadline δ, which can be given or calculated from start-times. Furthermore we treat splits as joins and vice-versa. E.g., the reverse-path from T to R is sequential and unambiguously determined, therefore R holds only one end-time tuple $(1.0, 43)$. The second value specifies the end-time, which is calculated by subtracting the duration of all successor-activities from an assumed deadline $\delta = 48$. Activity L can be reached on four routes (via N, O, P and Q), thus it has four end-times tuples $\{(0.2, 25), (0.2, 25), (0.48, 34), (0.48, 39)\}$.

In our approach we do not need any knowledge about relations between tuples and paths, thus it is possible to aggregate tuples with equal time-information by adding their probability. For L that would be $\{(25, 0.4), (34, 0.48), (39, 0.48)\}$.

Since these distributions can be represented as histograms we call them *Time Histograms*, which are formally defined as follows:

Definition 1 (Time Histogram) *A time histogram H is a binary relation with n rows (p, t) with probability p and time-information t.*

A time histogram H is valid, if $\sum_{i=1}^{n} p_i = 1$ for $(p_i, t_i) \in H$.

An extended time histogram T is a relation of n rows (p_i, c_i, t_i), (probability p, cumulated probability c, and time-information t), with $\sum_{i=1}^{n} p_i = 1$, and $c_i = \sum_{t_j \leq t_i} p_j$ for $1 \leq i \leq n$.

A cumulated time histogram is the projection of an extended time histogram on the cumulated probabilities and the time information.

When an activity may start is represented in a *Start Time Histogram* or *S-Histogram*. The possible termination times in *End Time Histograms* or *E-Histograms* The *Probabilistic Timed Graph* (see Figure 2) is be generated by calculating the E-Histograms and L-Histograms of all activities.

2.3 Calculation of the Probabilistic Timed Graph

The calculation of the graph starts with the initialization of the start activities S-Histogram with $(1.0, 0)$. To determine the S-Histograms for all remaining activities the forward-calculations specified below have to be applied depending on the according control structures (sequence, conditional execution, parallel execution).

Definition 2 (Sequential Execution) *For two sequential activities B and C we calculate the S-Histogram of the successor activity C as*

$$S_C = S_B + d_B = \{(p, s + d_B) | (p, s) \in B\}$$

For the successors S-Histogram the predecessors duration d is added to each start-time s of the predecessor. Examples for the calculation of sequences are the activities B, D, E and T.

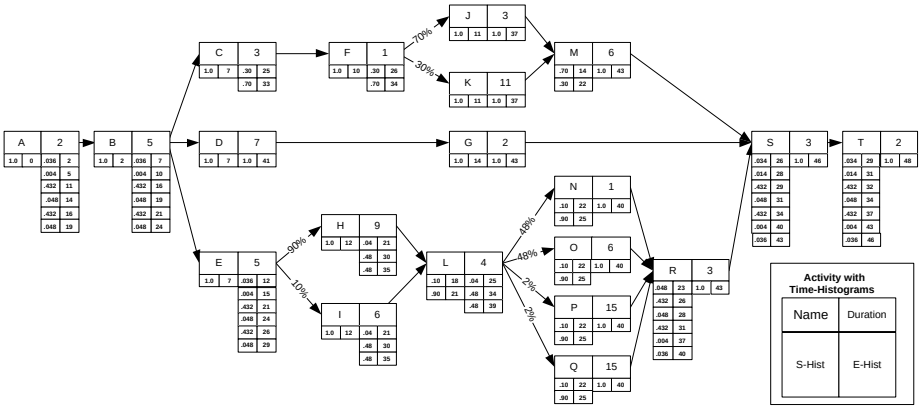

Fig. 2. Probabilistic Timed Graph with S-Histograms and E-Histograms

Definition 3 (Conditional Execution) *Let B_i be conditionally executed predecessors for the or-join activity C and q_i the according branching probabilities, then the S-Histogram S_C can be calculated as*

$$S_C = \{(p * q_i, s + d_i) | (p, s) \in \bigcup S_{B_i}\}$$

Each tuple (p, s) of every predecessors S-Histogram is aggregated into the successors S-histogram by adding the predecessors duration d to its start time s and weighting its starting-probability p with the given branching factor q_i. It is easy to see that the sum of all probabilities of the resulting histogram is 1, and thus it is a valid duration histogram. Examples are the activities L, M, and R in Figure 2.

After and-splits all succeeding routes will be processed concurrently. To calculate the S-Histogram for activities located after and-joins we define the following operation.

Definition 4 (Parallel Execution) *Let B_1 and B_2 be predecessors which are executed in parallel before the and-join activity C, then the S-Histogram S_C can be calculated as*

$$S_C = \{(p_{B_1} * p_{B_2}, max((s_{B_1} + d_{B_1}), (s_{B_2} + d_{B_2}))) | \\ \forall (p_{B_1}, s_{B_1}) \in S_{B_1} \wedge \forall (p_{B_2}, s_{B_2}) \in S_{B_2}\}$$

This definition can be extended to sets of time histograms because of the operations associativity. An example calculation can be found in figure 3 where 2 parallel activities X and Y with $S_X = \{(0.7, 6), (0.3, 9)\}$, $S_Y = \{(0.7, 6), (0.3, 9)\}$, $d_X = 6$ and $d_Y = 4$ join in activity Z. Note that we calculate the successor start-times from each predecessors start-time and duration $s + d$. Assuming that

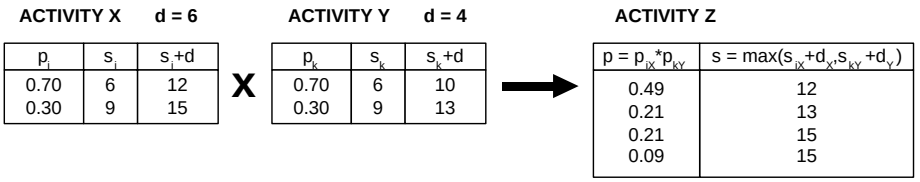

Fig. 3. Parallel execution of X and Y joining in Z

the concurrent activities are completely independent from each other, all possible end-time combinations have to be calculated whereas the greater end-time and the product of the probabilities are chosen for the resulting S-Histogram.

Before starting the reverse-calculation it is necessary to initialize the E-Histogram of the end-activity with $(1.0, \delta)$. δ is the workflows deadline, which can be chosen freely or calculated as structural deadline $\delta = max(se|(p, s) \in S_Z$ where Z denotes the workflows last activity. As stated above we have to change our point of view on the workflow-process, that means the calculation starts from the last activity, splits are treated as joins and vice-versa. The calculation-algorithms for S-Histograms (see definitions in section 2.3) must be modified as follows: Calculate the E-Histograms of activities on base of their successors (instead of predecessors). For sequences and conditionals subtract the successor durations d from their end-times e (instead of adding to s). For parallel execution calculate the predecessors end-times from the successor end-times and durations as $e - d$ (instead of their start-times $s + d$).

2.4 Additional issues on Time Histograms

For further issues in computations with time histograms we refer to [4], such issues are representation and calculation of iterations, compression of time histograms, checking the satisfiability of temporal constraints, and run-time assessments of the temporal situation of workflow execution.

3 Time Plans

The Timed Graph is the basis for the calculation of Time Plans which contain probabilistic information about execution intervals for activities. Time plans are not comparable with machine scheduling plans, because a participant still has the freedom of choice whether he/she executes a certain activity or not and when he/she executes it. Time plans are intended for the support of predictive personal scheduling issues by providing knowledge about upcoming activities and possible future bottle-necks.

3.1 X-Values

There is still one essential information, that the timed graph does not provide for personal scheduling: The probability that an activity will not be executed at

all. That knowledge may be of no interest in an overall process view, but for a participant who is supposed to execute an activity it is crucial. This information is stored in the *X-Value* of each activity in the workflow.

Definition 5 (X-Value) *Let x_A be the X-Value of an activity A, such that x_A specifies the probability of not executing A.*

The X-Value can be easily determined for every activity by linking its calculation to the forward-calculation of the E-Histograms (see definitions in section 2.3):

- Initialization of first activity with $X_{FirstActiviy} = 0$
- Sequences: $X_C = X_B$
- Conditional Execution: $X_C = 1 - \sum(X_{B_i})$
- Parallel Execution: $X_C = X_{B_1} = X_{B_2}$

3.2 Calculation of Time Plans

A time plan that holds all possible execution-intervals for an activity is defined as follows:

Definition 6 (Time Plan) *The Time Plan T_A on an activity A is a set of tuples $P_A = (p, s, e)$ where*

$$T_A = \{(c_S * c_E * (1 - x_A) + x_A, s, e) | (c_S, s) \in S_A \wedge (c_L, e) \in E_A\}$$

Note that we used cumulated S-Histograms and cumulated E-Histograms in this definition, which can be determined easily as stated in definition 1. Basically we create the cartesian product of the S-Histogram and the E-Histogram weighted by the activities X-Value. It provides information about the probability that no time-constrained is violated (deadline) when executing the activity in a certain time-interval. Figure 4 shows our example workflow with calculated time-plans for all activities. In this example each activity holds a tuple with probability $p = 1.0$, which means that an execution of the particular activity without constraint-violation is possible in the according time-interval.

Definition 7 (Safe Interval) *A Time Plan T_A of an activity A is safe if*

$$\exists (p, s, e) \in T_A | p = 1.0$$

*and (s,e) is called the **Safe Interval** of A.*

Taking a closer look on the time plan of activity H, the following conclusions can be drawn:

- H will not start before 12.
- In the Safe Interval between 12 and 21 execution without violating any time-constraints is possible.
- In time interval 12 to 30 there exists an 86.4%-chance that an execution without violating any time-constraints is possible.
- In time interval 12 to 35 there exists an 51.4%-chance that an execution without violating any time-constraints is possible.

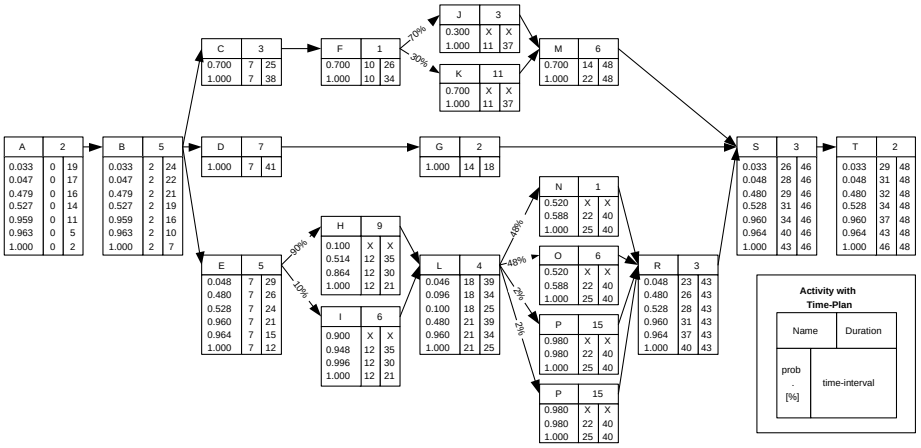

Fig. 4. Workflow with calculated Time Plans

4 Definition of Personal Schedules

Based on time plans one can calculate future workloads for participants in terms of personal schedules. We emphasize that these personal schedules are used for workload prediction only and not for providing directives to the participants. In particular future temporal bottlenecks and upcoming violations of time constraints should be detect as early as possible.

We assume that each workflow participant is responsible for executing activities from different workflows and different workflow instances. Therefore a number of *instance activities* including time plans is given for each participant, enabling future workload calculations for this participant. Such workload calculations are represented as *personal schedules* formally defined as follows:

Definition 8 (Personal Schedule) *A personal schedule PS_ρ of participant ρ is defined as a relation (v, p, s, e) with $(p, s, e) \in T_v$ and:*

v *... activity that participant* ρ *must work off*
T_v *... time plan of activity* v
p *... probability to meet execution of* v *in* (s, e)
s *... planned start time of activity* v
e *... planned end time of activity* v
$(e = s + d,\ d$ *... execution time of activity* $v)$

A personal schedule represents a possible execution plan (schedule) for a participant containing all his known future tasks. For each activity the personal schedule contains a planned execution interval and the corresponding probability to meet execution within this interval from the time plan. It is obvious that a personal schedule is not an optimized plan but only one possible execution order

used for workload calculations. Since we are working on administrative workflows, the real execution order is always up to the participant himself. Nonetheless we envision a situation where personal schedules support participants in choosing a most efficient execution order in the context of the whole workflow.

As mentioned above we calculate execution plans for participants and assume that activities are not executed in parallel. Therefore the planned execution intervals of a personal schedule must not overlap:

Definition 9 (Non-overlapping) *A personal schedule PS_ρ is non-overlapping if $\forall(v, p, s, e) \in PS_\rho$ the interval (s, e) is not overlapping with other intervals from PS_ρ.*

With the help of personal schedules, the most probable as well as the minimum and maximum future workload can be determined in advance.

As personal schedules predict future workload using probabilities, we are faced with uncertainties and estimations. In contrast to production planning systems, the purpose here, is not to find the optimal plan. Moreover in this context this would lead to a schedule algorithm which is known to be np-hard [1]. We therefore reduce complexity by using *earliest deadline first strategy*. That means we are planning activities gradually starting with the activity having the earliest deadline and so on.

5 Calculation of personal schedules for workflows

Having time plans for all future activities of a participant enables one to make some predictions about the worklist behavior of this participant in near future. We calculate personal schedules as a simulation of the activities that will be added to the worklist of a participant. This way potential overload can be detected and the probability for this overload occurring can be determined as well. In this section we show how to calculate personal schedules and how these calculations can be interpreted. To illustrate the steps of the algorithm a running example is used. Figure 5 shows some exemplary activities that participant 'M' must execute and for which the personal schedule calculation will be shown.

For each activity the time plan contains a *safe* interval, in which the execution is aimed at. The safe interval of an activity is that period of the time plan of the activity, for which the probability is $p = 1.0$ (see definition 7 above). Therefore the safe intervals for the activities of the example are:

- safe interval of $A = (4, 10)$,
- safe interval of $B = (8, 20)$,
- safe interval of $C = (18, 30)$.

On the basis of this example we show how a personal schedule is calculated activity by activity. The goal is, to find a possible execution sequence of these instance activities without overlaps. If no such execution sequence can be found, a temporal restriction violation is expected due to the overloading of participant 'M'.

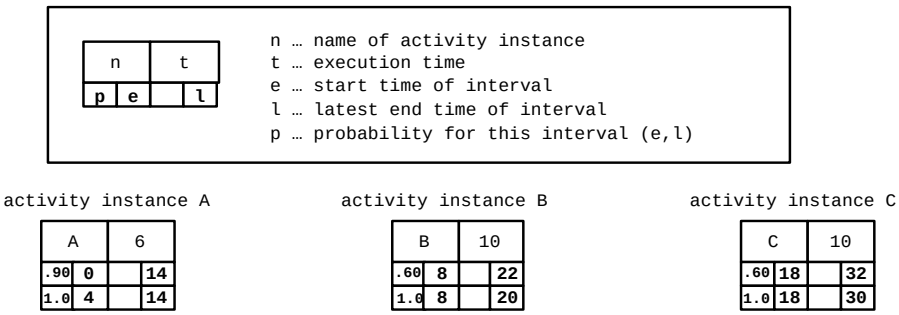

Fig. 5. Example: future activities including time plans of participant 'M'

At the beginning of the computation an empty personal schedule is initialized. The instance activities are inserted ordered ascending by their L-values (earliest deadline first). The individual activities are now inserted into the personal schedule. It is tried to find a execution period in the time plan of the current activity so that there is no overlap with the past activities in the personal schedule. The execution periods for all activities should lie within their safe intervals, since otherwise an execution of the workflow instance without temporal constraint violation cannot be guaranteed. If no non-overlapping execution period in the safe interval can be determined for an activity, the probability, with which the entire personal schedule can be held nevertheless will be computed. This value is important for decisions about the continuation of the workflows.

The pseudocode for personal schedule calculation is now shown followed by a description of the calculation for our example:

```
Input V                 set of activities including time plans
      type              normal, min or max personal schedule
      period            for min or max personal schedule
Out   personal schedule

FUNCTION calculatePersonalSchedule()
BEGIN
   V.sort() //sort activities by their deadlines
   PS := ∅
   noPlanFound = FALSE
   FOR every (v ∈ V) AND WHILE (NOT noPlanFound)
      bestPlan := ∅
      bestPeriodFound = FALSE
      v.timePlan.sort() //sort time plan entries descending by probabilities
      FOR every (period ∈ v.timePlan) AND WHILE (NOT bestPeriodFound)
         IF (period.startTime = 'x') THEN
```

```
            //X-Value, activity is planned to be not executed
            bestPlan.weigth(period.getProbability())
            bestPeriodFound = TRUE
         ELSE
            //find non-overlapping time interval in period
            newPeriod := getFreePeriod(PS, period, v.executionTime)
            newPeriod.probability = period.getProbability()
            //if period is safe
            IF (newPeriod.getProbability() = 1.0) THEN
               check constraints for minimized or maximized personal schedule
               bestPlan.add(a, newPeriod)
               bestPeriodFound = TRUE
            //if period is overlapping with other intervals
            ELSE IF (newPeriod.getProbability() = 0.0) THEN
               newPeriod.endTime := period.endTime
               newPeriod.startTime := newPeriod.endTime - v.executionTime
               //try to shift conflicting activities
               newPlan := freePeriod(bestPlan, newPeriod)
               IF (newPlan ≠ ∅) THEN
                  bestPlan := newPlan
                  bestPlan.add(a, freePeriod)
               END-IF
            //if period is not safe but non-overlapping
            ELSE
               newPlan = PS
               newPlan.add(a, newPeriod)
               IF (newPlan.getProbability() > bestPlan.getProbability()) THEN
                  bestPlan := newPlan
               ELSE
                  bestPeriodFound = TRUE
               END-IF
            END-IF
         END-FOR
         IF (bestPeriodFound = FALSE) THEN
            noPlanFound = TRUE
         ELSE
            PS := bestPlan
         END-IF
      END-FOR
      RETURN PS
   END

   function getFreePeriod()
      Returns a time interval of the specified length that lies within the
      specified period, if such an interval exists without overlapping with
      other intervals of the specified personal schedule.

   function freePeriod()
      Returns a new personal schedule containing all activities of the specified
```

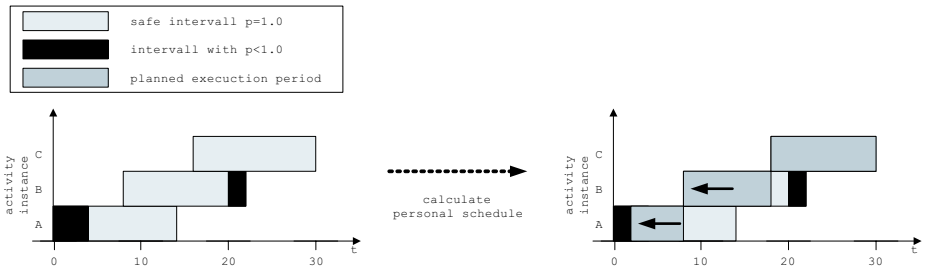

Fig. 6. Example: calculation of a personal schedule

`personal schedule. In the new personal schedule the execution times of all are shifted, so that their is no overlapping with the specified period.`

In detail the determination of a valid start time is processed as follows. For each activity the earliest possible time in the safe interval is selected as start time first. Since the first activity A from the example is the first one inserted to the personal schedule there is obviously no overlapping. The execution of A is thus planned at the beginning of the earliest interval: $(4, 10)$. However if there are any overlaps with other periods in the personal schedule, the earliest start time in the safe interval without overlaps is determined. For activity B this case occurred and so its execution is planned at $(10, 20)$.

If no non-overlapping execution period was found in the safe interval by shifting, the resulting personal schedule cannot be guaranteed any more. That means violation of time restriction may occur. In that case the probability with which the personal schedule can be met nevertheless, is determined. As previously mentioned it is sufficient for the computation of a personal schedule to pursue an earliest deadline first strategy. No non-overlapping execution period was found for activity C in our example. So we try to determine an optimal schedule by shifting the current activity or by shifting the overlapping predecessor activities.

The current activity can be shifted thereby only toward the deadline, since bringing the start of this activity forward is not possible because of the overlapping activity. On the other hand the temporal conflict can be dissolved by starting the conflicting activity earlier. This is shown in our example where the start time of activity C cannot be protracted to resolve the conflict and so the predecessor activity must be moved to an earlier starting times. This is recursively accomplished, until an execution plan without overlaps will be found. If recursive shifting of the predecessor activities does not result in a non-overlapping execution plan, then the entire personal schedule computation is broken off. In this case a successful treatment of the activities of this participant is not possible and escalation decisions should be made urgently.

5.1 Admissibility of a personal schedule

As we have shown how to calculate a personal schedule, we are now interested in how to interpret the personal schedule and how to provide helpful information for participants as well as for workflow designers.

It is of interest whether a personal schedule is safe, that is, if all its activities can be executed in the planned interval without restrictions. Restrictions arise because the activity is not yet available at the planned start time or because the planned end of the activity leads to deadline violations. The admissibility of a personal schedule is defined as the probability, with which no time constraint violations occur if all activities are executed in their planned periods.

Definition 10 (Admissibility of a personal schedule) *Let PS be a personal schedule and let p_i be the probabilities of the activities v_i of PS. The admissibility of PS is defined as $\zeta = \prod p_i | (v_i, p_i, e_i, s_i) \in PS$ and $0 \leq \zeta \leq 1$. A personal schedule is safe, iff $\zeta = 1.0$.*

For the exemplary personal schedule from the previous section the admissibility can be calculated as: $\zeta = 0.9 * 1.0 * 1.0 = 0.9$. That means with an admissibility of 90% this personal schedule can be met or in other words that with an admissibility of 10% some time constraint violations will occur making escalation decisions necessary.

5.2 Workload of a personal schedule

Another statement which can be made from the computed personal schedules refers to the workload of participants in certain time intervals. Having a personal schedule the question about the expected workload in one period, *i.e.,* on one day, in one week or in any time interval, can be answered. The workload of a participant within any period can be determined from the overlap of the desired period with all activity entries of the personal schedule. Formally the workload of a personal schedule is defined as follows:

Definition 11 (Workload of a personal schedule) *Let PS_ρ be the personal schedule of participant ρ. The workload of the personal schedule within a period $\gamma = (s, e)$ is defined as:*

$$\eta = \sum overlap(\gamma, (s_i, e_i)) | (v_i, p_i, s_i, e_i) \in PS_\rho$$

$overlap((s_1, e_1), (s_2, s_2)) = max(min(e_1, e_2) - max(s_1, s_2), 0)$ returns the overlap between two time intervals.

For our exemplary personal schedule the workload for *i.e.,* period $(0, 7)$ (which may stand for next week) may be determined as:

$$\eta_{(0,7)} = overlap((0,7),(2,8)) + overlap((0,7),(8,18)) + overlap((0,7),(18,30)) = 6 + 0 + 0 = 6.$$

That means that participant ρ, for whom the personal schedule was provided is busy in the next week at 6, out of 7, days with activities from the workflows. If the granulation for the workload calculation is reduced, it can be stated that the free period falls on the first day (between TU 0 and 1).

5.3 Minimized personal schedule

A minimized personal schedule represents a variant of the original personal schedule, in which one tries to shift as much activities as possible out of a certain period without violating time restrictions. This computation is obviously only practically based on safe personal schedules, since otherwise the minimum workload is anyhow larger than the available time. One should already have interfered correctively.

By shifting activities out of a period, one receives the minimum load for a participant from the workflow in this period and thus a good basis for various planning decisions (*i.e.,* accept additional orders, vacation planning, ...). The computation of a minimized personal schedule is made similar to the original personal schedule calculation. An additional constraint is added: Try to move all activities, whose planned execution time overlaps with the given minimized period, out of this period. Shifting the activities may only take place within their safe interval.

5.4 Maximized personal schedule

A maximized personal schedule represents another variant of the original personal schedule, in which one tries to shift as much activities as possible in a certain period without violating time restrictions. The determination of maximized personal schedules makes sense in order to avoid *i.e.,* unbalanced workloads. It is to be considered however that by shifting the activities the execution of these activities is unnecessarily delayed. Buffer time is lost and cannot be used by following activities if time exceeding occurs.

6 Applications and Conclusions

In the previous sections we introduced time plans and personal schedules. Finally we want to describe some application to demonstrate how these information can be used in workflow systems.

- Provide early information about future activities for participants: At the start of an instance the participants (persons) can already be informed about their future tasks since this information is contained in the computed time plans.
- Recognize delays because of overload: The admissibility of a personal schedule is limited due to the overloading of the participants. If this admissibility of the personal schedule is observed, then delays in the instances can be

promptly recognized. Moreover by watching the individual personal schedules of the participants over a certain period also bottlenecks can be identified.

- The admissibility of a personal schedule gives information about the probability with which a successful execution of the instances the participant is involved in is possible. This value can be linked with threshold values, in order to be able to accomplish automatic control of the current instances. It could be specified that *i.e.,* if the admissibility of a workflow instance falls below 95%, a warning is triggered. For critical workflows the value could also be set conservatively to 100%. Further a second value (*i.e.,* 80%) could be specified causing an error alarm to occur if the admissibility of the personal schedule falls below this threshold. Such a model is called *traffic light model* (see [5]) since different states are assigned to each workflow instance according to its admissibility: green, yellow, red.
- Determine overloaded and idle participants: If future workload of participants is computed and evaluated for certain periods (*i.e.,* workload in the next week, workload in the next months, ...), then it is easy to recognize whether certain participants will be overloaded or will have idle time left. This way the basis for controlling interferences is given. It shouldn't be ignored that these mechanisms could also be used to determine and control efficiency of employees.
- Before the start of a new workflow instance a personal schedules can be calculated to check if the new instance will lead to some constraint violations due to capacity bottlenecks. If the personal schedule is safe, then the instance can be started without any problems. Otherwise it should be considered whether measures have to be taken, in order to ensure a successful execution of all instances, or if a delay of the new instance is necessary. It may be useful to embed the computation and analysis of personal schedules into a system for scenario planning.
- Early warning systems and scenario planning: Future bottlenecks and time exceeding should be recognized as early as possible. An early warning system for workflow systems can be established with the help of personal schedules. Therefore, before the start of each new instance, personal schedules for all participants of the workflows are computed. With the help of the computation of admissibility the probability for successful execution can be already given to the new instance before the start. Due to this value one can decide, whether the instance is started at all or whether it proves as favorable to delay the execution of the instance or use perhaps additional resources.

The introduction of personal schedules has the aim to make information about workflow execution available for the participants as early as possible. The integration of personal schedules into personal digital assistants, and worklist managers, as well as feedback from personal schedulers to workflow time managers are subject of ongoing research.

References

[1] J. Brucker. Scheduling Algorithms. Springer Verlag, 1998.

[2] C. Bussler. Workflow Instance Scheduling with Project Management Tools. In *9th Workshop DEXA'98*, 1998. IEEE Computer Society Press.

[3] P. Dadam, M. Reichert, and K. Kuhn. Clinical workflows - the killer application for process-oriented information systems?

[4] J. Eder and H. Pichler Duration Histograms for Workflow Systems In *Proceedings of the Working Conference on Engineering Information Systems in the Internet Context*, pages 239–253, Kanazawa, Japan, 2002.

[5] J. Eder and E. Panagos. Managing Time in Workflow Systems. In *Workflow Handbook 2001*. Future Strategies INC. in association with Workflow Management Coalition, 2000.

[6] J. Eder, E. Panagos, and M. Rabinovich. Time constraints in workflow systems. In *Proc. International Conference CAiSE'99*. Springer Verlag, 1999.

[7] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.

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

[9] 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.

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

[11] P. Lawrence. *Workflow Handbook 1997*. John Wiley & Sons, 1997.

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

[13] M. Ninaus Auslastungsberechnungen in probabilistischen Workflow Systemen *Masterthesis*. ISYS Department, University of Klagenfurt, Austria, 2002.

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

[15] 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.

[16] E. Panagos and M. Rabinovich. Reducing escalation-related costs in WFMSs. In *NATO Advanced Study Institue on Workflow Management Systems and Interoperability*, Istanbul, Turkey, August 1997.

[17] H. Pozewaunig, J. Eder, and W. Liebhart. ePERT: Extending PERT for workflow management systems. In *First European Symposium in Advances in Databases and Information Systems (ADBIS)*, St. Petersburg, Russis, 1997.

[18] Workflow Management Coalition, Brussels, Belgium. *Glossary: A Workflow Management Coalition Specification*, November 1994.