Time Patterns in Workflow Management Systems

Cosmina Cristina Niculae
Department of Mathematics and Computer Science, Eindhoven University of Technology
P.O. Box 513, 5600 MB, Eindhoven, the Netherlands
c.c.niculae@student.tue.nl

Abstract

Business processes running in nowadays organizations become more and more complex and need to incorporate aspects like deadlines, service level agreements, schedules, consequently various time constraints. Several conceptual models aiming to formally define workflows are available in the Business Process Management field. Nevertheless, these conceptual models do not incorporate time constraints in their definition, not allowing the process designer to model time related aspects in the business process. Moreover, the available workflow management systems supporting the enactment of business processes in organizations do not properly deal with time. The aim of the current paper is to enrich an available conceptual workflow model from literature with a set of eleven time patterns. This collection of patterns is the result of an extensive literature review and summarizes in a compact representation the patterns present in the research papers in this domain in the last ten years.

Acknowledgements

The current report is the result of a Capita Selecta project which was part of my master studies curricula. The project was conducted in the Architecture of Information System Group, at the Mathematics and Computer Science Department of the Eindhoven University of Technology. I would like to thank professor Wil van der Aalst for his supervision, as well as for offering me support and wise pieces of advise while conducting my research.

1 Introduction

Time aspects like deadlines, service level agreements, schedules or activity durations are just a couple of examples that need to be supported in real-life business processes. Moreover, the availability of the resources that need to execute activities for a case and the necessity of providing them with personal schedules (in which to be stated the present and future activities that a resource must perform together with the associated times when these activities should be performed) needs to be considered. The different granularities used when expressing time in real-life situations (e.g. days, hours, business days, minutes, etc.) represent another issue in dealing with time constraints in business processes. Nevertheless, the workflow management systems supporting the enactment of business processes in nowadays organizations deal very little with time aspects, lacking the processes from expressiveness and flexibility. Temporal aspects of business processes have been studied in literature (e.g. [3, 4, 6, 8, 14, 21]) in the last ten years and several time constraints have been researched. Some prototypes of systems supporting time patterns have also been implemented (e.g. [4, 18]) and algorithms for consistency checking (checking if cases that satisfy all the time constraints present in the model can be executed) and generation of schedules for the implicated resources have been addressed (e.g. [5, 7, 16, 21]). Dealing with different time granularities in a business process has also been studied in the specialized literature (e.g. [2, 6, 11]).

Although intensive research has been done in this domain, currently there exists no work to integrate all the valid time constraints proposed in the literature, each author defining the time patterns independently and considering only a limited set of them.

The main objective of the current paper is to revise and collect all the time patterns addressed in the specialized literature in the last ten years. Furthermore, having as a starting point this collection of time constraints, a compact and comprehensive list of time patterns is provided. The main problems in dealing with time patterns in the current workflow management systems are also identified.

In this paper an available conceptual workflow model is enriched with a set of eleven time patterns identified in the literature. Moreover, a detailed review of the time patterns is provided, followed by a short description of the main problems arising when dealing with time aspects in workflow management.

We consider the need of providing a comprehensive and justified collection of time patterns requiring support in real-life business processes, the first step in the attempt of creating time-aware workflow management systems. Therefore, the time patterns presented in this study are subjected to a formal definition. Moreover, some practical aspects related to the implementation of the provided time constraints in a workflow management system are also addressed.

The research method used consists of a quantitative literature review aiming to consider the most important authors dealing with time aspects in workflow management in the last ten years.

The remaining of this paper is organized as follows: a conceptual workflow model available in the literature is introduced in section 2; further, temporal concepts needed in specifying the time constraints are defined; section 4 provides a collection of eleven time patterns identified in the reviewed literature; section 5 provides a description of the main aspects that need to be considered when dealing with time in workflow management; the report ends with conclusions and future directions.

2 Atemporal Workflow Management Systems

The increase in complexity and number of the business processes running inside nowadays organizations triggered the need to efficiently handle the logistics of such business processes by using workflow management systems. Such a system aims to automate the business processes by deciding which resource is needed for each step of the work and which is the correct order which needs to be followed when executing activities. A formal definition of a workflow management system is given by the Workflow Management Coalition [12] : "A system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic." The workflow management system is a practical implementation of the workflow management theory. In the remaining of this section we abstract from the workflow management systems and focus our attention to the theoretical concepts of workflow management in general.

This section is organized as follows: firstly, an introduction to the basic notions of workflow management is given; secondly we focus on the process dimension of a workflow and we introduce a representation for the main routing patterns used to define the workflow process; this section concludes with an example of a process for requesting loans in a bank which will further be referred when enriching the process definition with time support.

2.1 Short Introduction to Workflow Management

A workflow has three dimensions: the case dimension, the process dimension and the resource dimension [1]. Cases run in isolation, independent from one another. An example of a case is a request for a loan coming from a client, which represents an external customer (cases can also be generated by internal customers when other departments inside the same organization trigger a case execution). The execution of a case follows a set of rules defined within the workflow process. These rules are represented by means of tasks and conditions. Tasks constitute pieces of work that

need to be executed within the process while the flow from one task to another is defined through conditions (the pre-conditions of a task need to be true in order for the task to be executed, while the post-conditions of a task need to be true after the task has been executed). Tasks are executed by resources, either humans or machines. A proper resource classification helps abstracting from using the identity of an employee. The employee is therefore placed within a resource class, which gathers resources with similar characteristics. A resource class is typically based on roles (having similar skills, competencies, qualifications) and groups (inside a department, a team or an organizational unit). In this way, the process does not need to be changed in case changes in personnel occur, the resource classification closely follows the structure of the organization and the work distribution (which task is executed by which resource) is properly defined.

Aiming to enrich workflows with support for time constraints crosses all the three dimensions discussed above, as it will be shown later in this paper.

2.2 Workflow Process Definition

As conceptual workflow model used throughout the present paper, we consider the model proposed in [1], as this model is close to the recommendations made by the Workflow Management Coalition. The author of this paper uses high-level Petri Nets (Petri Nets extended with color to model data, with time and with hierarchy to structure large models) for modeling the process dimension of a workflow. The reason behind this decision is that Petri Nets are a simple and in the same time powerful modeling language able to represent the four main routing patterns applicable to a workflow model as identified by the Workflow Management Coalition [12] (sequential, parallel, conditional and iteration), patterns which need to be supported in all important modeling notations existent in the Business Process Management field. Consequently, Petri Nets are a general mean of representation for workflow models.

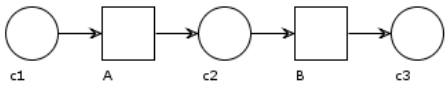

Figure 1: Sequential Pattern

When using Petri Nets concepts for modeling a workflow process, tasks are represented as transitions, conditions are represented as placed and cases are represented as tokens.

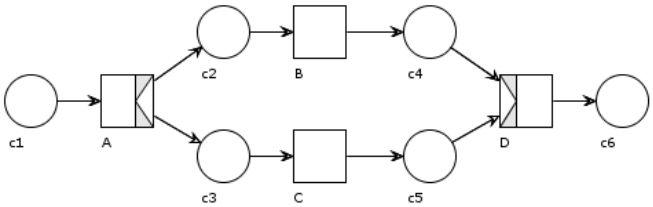

Figure 2: Parallel Pattern

The **sequential pattern** in presented in Figure 1. The two tasks, A and B, are executed sequentially, meaning that task B is executed after the completion of task A.

The **parallel routing**, depicted in Figure 2, presents two tasks, B and C which are executed simultaneously. The AND-split (task A) and AND-join (task D) special transitions are present in this graphical representation. These transitions are shortcuts ("syntactic sugar") of the equivalent, standard Petri Net representation. The AND-split A enables the execution of both B and C while the AND-join D synchronizes the two flows.

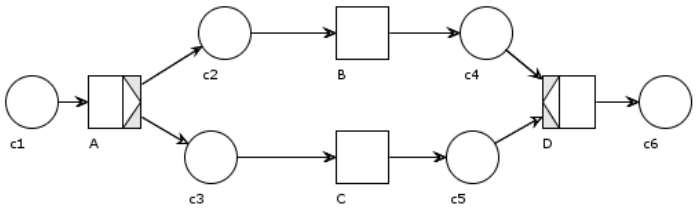

Figure 3: Explicit Choice Pattern

The **conditional pattern** is used to model a choice between two or more activities. The moment in time when the decision of executing one of the alternative activities is taken, makes the distinction between two types of conditional patterns, namely the explicit choice (Figure 3) and the implicit choice (Figure 4).

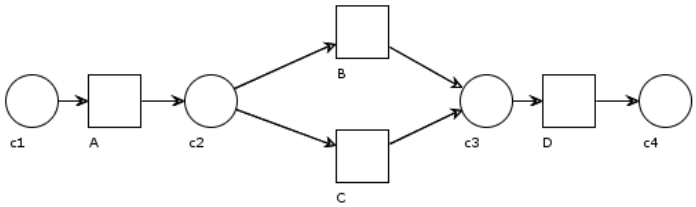

Figure 4: Implicit Choice Pattern

In the case of an explicit choice, the XOR-split (task A) and XOR-join (task D) special tasks are present and the decision is made after the completion of task A. In the case of Figure 4, the decision is made later, namely in the moment when either B or C are executed.

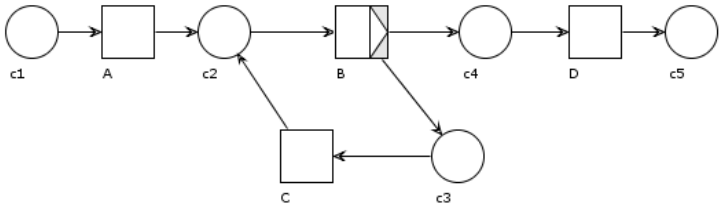

Figure 5: Iteration Pattern - While Variant

The **iteration pattern** has several variants (while variant, repeat variant and combination variant) and models the execution of a task for multiple times. We have chosen to present only the while variant in this section (Figure 5) as it is outside of the scope of the current paper to discuss in detail all the iteration patterns.

As a workflow system is reactive to the environment, four types of triggers are defined to express the interaction between the system and the surrounding environment.

- The *resource triggered task* defines a task which needs the availability of a resource in order to be executed.
- The *message triggered task* defines a task which needs the arrival of an external message in order to be executed (e.g. incoming e-mail).

- The *time triggered task* defines a task whose execution is triggered by a clock (e.g. the "archive case" task is triggered if the necessary documents are not received in 2 days). It needs to be mentioned that time constraints needed in real-life processes are complex and numerous, imply not only individual tasks but also relations between different tasks in the process, either consecutive or disconnected, imply dealing with multiple granularities etc., and therefore cannot be captured by the time trigger symbol, which just enables the execution of a task at a predefined time - a correspondent for deadline. Therefore, we consider the current process definition to be atemporal, not providing the time support necessary for real-life business processes.
- The *automatic task* for which no triggers are needed.

For further details concerning the Workflow Nets representation, we refer the user to [1].

2.3 Atemporal Example: Loan Application

As an example, the process of applying for a loan is considered. In the present subsection the process is described and modeled using the conceptual model described in this chapter. Later on, we will revise the process, considering the temporal constraints that the process incorporates and a new process model for the same situation, enriched with time patterns, will be provided.

The process of applying for a loan is modeled for a bank situated in Europe. The process starts when a client comes to one of the bank offices and requests a loan. The bank office is opened every day from Monday to Friday from 9:00 A.M. until 18:00 P.M. except the legal holidays. The client is received by a front office desk employee and is given a form to fill in. This operation takes between 5 and 10 minutes. After this, the client is requested to wait no more than 20 minutes until a credit officer becomes available and, as soon as this happens, the client is assigned to the available credit officer. Immediately, a meeting of maximum one hour starts where the credit officer explains to the client the procedure that needs to be followed and the documents that need to be provided to the bank. After the meeting, the bank waits for the needed documents. If the documents are not received in 10 business days, then the loan request is rejected. If the documents are received in time, a report is made by the credit officer. The report is sent for approval to the central bank office situated in America. The time zone difference between the two locations is of 10 hours. As soon as an answer is received from the central office, the request is further processed: either it is rejected or it is accepted. If the request is accepted, a positive answer is sent to the customer together with the payment conditions. This activity is not performed during the legal holidays. After the loan is accepted, the customer, who has received the loan, starts paying regular interest rates to the bank to gradually return the money. This is done according to a pre-established timetable between the customer and the bank. When the payment is completed, or in the eventuality that the request is rejected, the case is archived and the process ends. The Service Level Agreement between the customer and the bank states that the maximum allowed distance in time between the meeting with the credit officer and the positive or negative answer sent to the customer should be between 20 and 30 business days.

The model of this process description, abstracting from the time constraints and from the resource classification, is presented in Figure 6.

3 Temporal Concepts

Before introducing the list of time patterns identified in literature as needing to be supported by current workflow management systems, we must address some general time aspects applicable in workflows enriched with time support. Time aspects are important when formalizing the time constraints corresponding to each of the patterns identified and will be addressed as design decisions.

In their work [13], time in a workflow application is defined as being linear and having a determined starting time (e.g. the time when the system is installed) while it has an undetermined

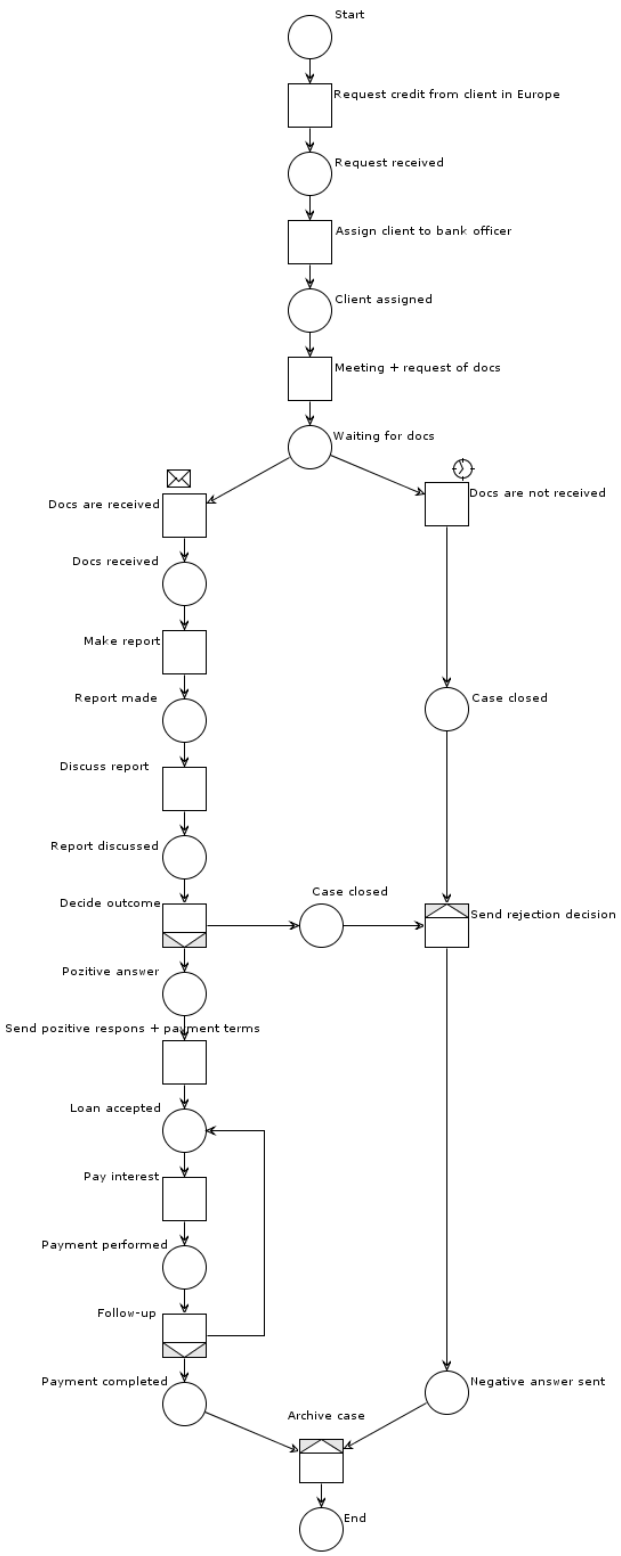

Figure 6: Atemporal Loan Application Process

upper bound (time is infinite in the future). Moreover, in a workflow management system time is discrete and measured in a certain time granularity. Dealing with instants, intervals and durations must also be supported by a workflow application.

Several notions concerning time have been introduced while defining time in workflow management systems. Other similar notions need to be introduced in order to facilitate the time patterns definition.

Time granularity [17] is defined as a unit of measure for time, such as second, minute, hour, day, week, month, year and so on. Between different time granularities there might exist certain constraint relations (e.g. 1 minute = 60 seconds) or uncertain ones (e.g. one month might have 28, 29, 30 or 31 days). Lanz et al. [15] state that time parameters can be specified in basic granularities (i.e. years, months, weeks, days, hours, minutes, seconds), in system defined granularities (e.g. business days) or in user defined granularities (e.g. Monday morning). In [4], a similar division of time granularities is given, namely standard granularities (e.g. days, weeks, years, etc.), bounded granularities (e.g. Years since 2008) and specialized granularities (e.g business days, business months). The two approaches are approximately equivalent, the system defined granularities corresponding to the specialized granularities and the basic granularities corresponding to the standard granularities. Nevertheless, based on our understanding, the user defined granularities represent a broader class than the bounded ones, as the granularities included in this class do not necessarily need to be bounded to a standard granularity value (e.g. Monday morning). Therefore, we adopt in this paper the more general classification provided in [15], categorizing granularities in *basic*, *system defined* and *user defined*.

In [21], Sadiq et al. state that the specification of time can be done using both relative and absolute time values and considering a certain time granularity. Relative time values for a certain granularity do not have a fixed position on the time axis (e.g. 2 hours, 10 days, 7 weeks) while absolute time values of a certain granularity have a fixed location on the time axis (e.g. 24 December 2010). The same time specification is provided in [11], the two categories being named anchored and unanchored time primitives. In this paper, the notions of *absolute and relative time values* will be further used.

Instants are defined as specific points on the time axis. *Intervals* are derived types from instants and represent either the slot on the time axis starting from a certain time instant, measured before a certain time instant or being bounded by two time instants [4]. *Durations or time distances* represent a bounded interval specified in a certain granularity [15]. *Events* are occurrences of facts taking place during a case execution at particular time instants [4, 15]. Some examples of events are the start of a task (the time when the agent picks the activity from the working list and starts executing it) or the completion of a task (the time instant when the activity is finalized). We will further use the notion of events in the succeeding of this paper.

The notion of a *calendar* has been defined by several authors [11, 15, 17]. A *calendar* represents an infinite set of time units of varying granularities.

A *schedule* is defined as a (possibly) infinite set of subsets of the time points of a calendar. A schedule can contain either a set of discrete points from a calendar (e.g. every day at 10:00, 12:00, 14:00 and 16:00) or a set of intervals from a calendar (e.g. Monday-Friday 9:00 - 18:00) [15].

In [8, 15, 21] three important milestones for the specification of time patterns during the lifetime of a workflow management system are identified: at *build/design time* (when the designer models the workflow process), at *instantiation time* (when a new case is instantiated) and at *run-time* (when a case is executed). We will refer to these three time moments later in this paper.

4 Time Patterns in Workflow Management Systems

Workflow management systems running in today's organizations need to manage time constraints in order to be able to instantiate real-life processes and to offer flexibility for corporations. Research has been done in academia in trying to define the time patterns that need to be supported by workflow applications. Nevertheless, due to our knowledge, a complete collection of time patterns that should be implemented in a workflow management system has not been gathered so far.

The majority of the authors reviewed in this paper which have tried to define time constraints have handled only subsets and concentrated their efforts in offering a broader perspective over this issue, including algorithms for consistency checking and schedule generation for the resources involved (e.g. [4, 8, 10]). Lanz et al. [15] came up recently with a paper similar with the present one, by defining a set of ten time patterns and specifying them formally. We aim to further extend their work by defining a comprehensive collection of time patterns resulting from a quantitative literature review.

The identified time patterns will be defined considering the events taking placing during the life-cycle execution of work items in a case. Symbols are associated with the time patterns in order to visually identify these constraints on a process schema. These symbols represent no standard notation and have a purely illustrative purpose.

The remaining of this section is structured as follows: first, the execution life-cycle of a work item is reviewed, next the eleven time patterns are formally introduced together with their correlation with the papers included in this study and finally the loan application example is reexamined, identifying the time patterns present in it and remodeling the process using the newly introduced symbols.

4.1 Work Item Life-cycle

During the execution life-cycle, a work item normally progresses through a set of states, from creation until fail or completion, as it can be seen in Figure 7. The life-cycle of a work item has been addressed in studies like [19, 20].

A work item is initially **created** when the preconditions required for its enablement are accomplished and the item is ready to be executed. From this point on, several paths to completion are possible and both the workflow system and the resources involved in the execution of the work item are triggering actions that carry it from a state to the other. The actiones triggered by the information system are symbolized in the diagram as black arrows with an S label, while the actions triggered by resources are represented as green, dashed arrows with an R label.

After creation, a work item is **offered** by the information system to one or several resources or it may be directly **allocated** to one resource. The transition from offered to allocated can be also made once a resource picks the work item from the offered items list by issuing the allocate command. When a work item is allocated to a particular resource it means that the resource has committed to execute it in the near future. A work item is in **started** state when a resource starts to execute it. The resource can also decide to **suspend** the execution of a task for a certain period and to resume it on a later time. Besides a resource being able to suspend a task in execution, we also consider in this research the possibility of the system to suspend a work item in execution, saving its current state and resuming it later for execution to the same resource. These newly introduced transitions (symbolized with red arrows in the diagram) will be justified later in this chapter when formally introducing the time patterns.

Finally, a work item is **completed** once a resource has successfully finished executing it or might **fail** when its completion is not possible.

It needs to be mentioned that a larger number of transitions are possible between the states in the work item life-cycle as presented in [20]. In the diagram above we have chosen to represent only the transitions that are meaningful in order to define when a work item complies with specific time constraints.

4.2 Time Patterns

In this section we discuss the range of time patterns that have been identified in the reviewed literature. The patterns will be defined based on the different states that a work item can take during its life cycle, as discussed in Section 4.1. First the patterns are introduced, while the chapter ends with summarizing the presence of the patterns in the academic studies considered.

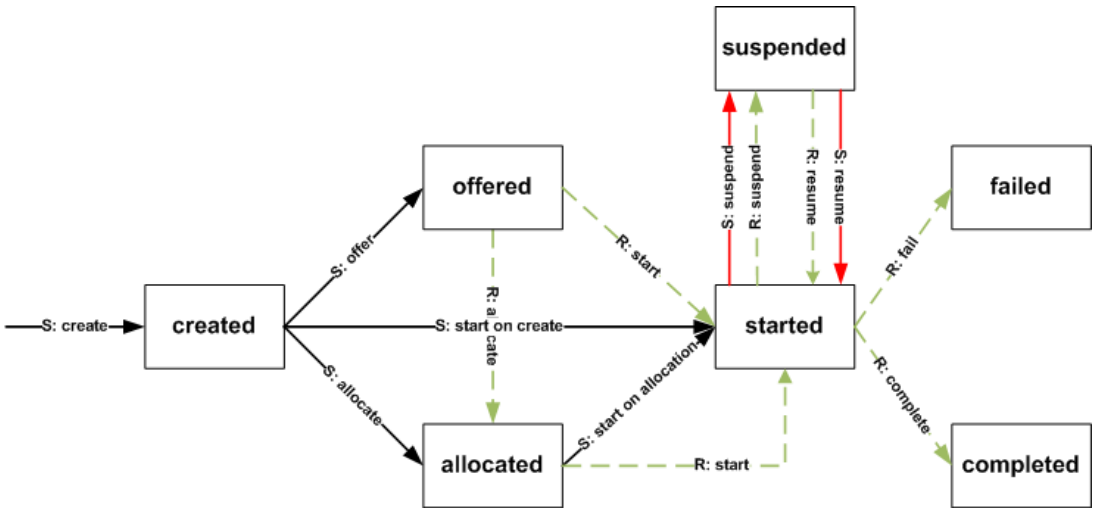

Figure 7: Work Item Life Cycle

4.2.1 Time Patterns Definition

1. Pattern T-TD (Task duration)

Description The ability to specify at design time the time length in which an activity needs to be completed.

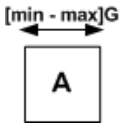

Figure 8: Task Duration Pattern

Related Patterns

- **T-TC2A** - *Time constraints between 2 activities*
- **T-TC2E** - *Time constraints between 2 events*

Design Decisions The time duration is specified either as a minimum duration, a maximum duration or an interval (i.e. minimum and maximum duration). The duration needs to be specified in a certain granularity and the time is expressed in relative values.

Examples

- The registration procedure for a new customer takes between 5 and 10 minutes (task duration expressed as an interval).
- Depending on the gravity of the case, a surgical intervention for a patient with lung problems might take a maximum of 10 hours (task duration expressed as a maximum duration).

Motivation Task duration offers the ability to specify at design time the amount of time in which an activity needs to be completed during runtime. This type of constraint is usually imposed by external factors (e.g. rules, regulations, Service Level Agreements). By supporting this pattern, the system is able to notify the resources involved in the execution of this activity about the constraint, keeping them informed about the time aspects concerning the specific work item and helping in this way the process to comply with the external rules or regulations.

Implementation suggestions The task duration pattern refers to the time interval passing since the work item is **created** and until its **completion**. Therefore, from the moment the work item becomes enabled and until its execution is successfully completed, a certain time length needs to be complied, this length including the intermediary states through which the work item passes during its life cycle as **allocated** or **offered** for example. In case the duration is expressed through the mean of a minimum duration, then the time exceeded between the creation and completion of the respective work item should not be lower that the specified duration. In case

of a maximum duration, the above time should not be higher than the imposed duration while in case of an interval constraint specification, the work item needs to be completed within the imposed boundaries.

One possible way to implement this in a workflow system is by starting a timer once the work item enters the **creation** state during run time and constantly monitoring the time elapsed since this moment and until the final **completion** of the activity. In case the time constraint imposed by this pattern is not met, an exception needs to be raised.

Evaluation in Literature Defining a task duration constraint is one of the most common time patterns identified in literature, as it can be seen in Table 1. Eder et al. [8, 9, 10] define durations as being deterministic and make no distinction based on different time granularities. In [21] the duration constraint for a task is defined either by using a single value specified in relative time (e.g. 3 hours) or as an interval, containing the minimum and the maximum possible values (e.g between 10 and 30 minutes). Granularity issue is considered. Combi et al. [4] extend this last definition by discussing the impossibility of the designer to specify a fixed time duration for an activity at the process definition. Moreover, the interval for defining a task duration might not be completely determined (i.e. only the minimum or the maximum durations might be known). In [18], an activity duration is more complexly defined by specifying a minimum duration, a maximum duration, a mean and a variance, in order for a distribution function to be fitted using these data. Lanz et al. [15] define durations by specifying a minimum duration, a maximum duration or an interval. Moreover, the definition is extended by allowing durations to be applicable not only to activities, but also to sets of activities, to the process model and to sets of process instances.

In the current paper durations can be applied only to activities, as applying durations to groups of activities or process instances can be further derived from this pattern. Moreover, durations are specified in terms of minimum value, maximum value or interval, considering a certain time granularity and considering relative time.

2. Pattern T-D (Deadline)

Description The ability to specify at design time that a certain event needs to be executed at a specified date.

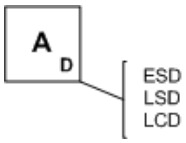

Figure 9: Deadline Pattern

Related Patterns

- **T-RTCS** - *Repetitive time constraint with calendar support*

Design Decisions The deadline constraint might be associated with two states of the life cycle of a work item (i.e. events), namely the **start** of an activity (specified by the latest start date - LSD and/or the earliest start date - ESD) and the **completion** of a task (specified by the latest completion date - LCD). Time might be relative or absolute and the granularity issue needs to be considered.

Examples

- The report has to be handed in before 25 December 2010 (deadline expressed as latest completion date - absolute time).
- The update of the report needs to be started at latest 2 months before the conference (deadline expressed as latest starting date - relative time).
- The medicines need to start being administered to the patient not earlier than 11:00 o'clock in the morning (deadline expressed as earliest starting date - absolute time).

Motivation The deadline pattern offers the ability to specify at design time that the start and completion states of an activity need to comply to certain fixed dates. This type of constraint might be imposed by external regulations or by internal requirements of the organization running

the process. The workflow system implemeting this pattern needs to notify during run time the resources involved about the constraint.

Implementation suggestions When considering the deadline pattern and assuming that the system automatically allocates work items to resources after creating them, 3 states in the life cycle of a work item become of interest, namely the **allocated** state, the **start** state and the **completion** state. In case of a deadline specified as a latest start date, since the time instant when the work item is allocated to a single resource, the resource has at most the time elapsing between the allocation date and until the deadline date to start the activity. In case of a deadline specified as an earliest start date, then once a resource has an allocated work item with such a constraint, that he/she needs to wait since the moment of allocation at least until the deadline date in order to start the execution of that particular work item. In case of a latest completion date, since the resource has an allocated work item with such a constraint, he/she has time to complete the execution of that particular work item (including start and possible suspend states) until the deadline date.

In case the resources also execute the allocation step, the work items being offered to a common work list and the resources needing to execute the "allocate" command to pick the work items and carry them in the **allocated** state (see Figure 7), then the above explanations need to be reconsidered, considering time elapsed since the work items enter the **offered** state and not the **allocated** one.

A possible way to implement such policies in a workflow system is by starting a timer once the work item enters the state **allocated** (if the system automatically allocates work items to resources) or the state **offered** respectively. The timer should expire as soon as the deadline date passes. If the activity has been started/completed when the timer was still active, then the deadline was met, otherwise the resources failed to comply with the deadline.

In case resources fail to comply with these constraints or the system fails to allocate/offer the work items in time in such a way that the resources can meet the deadline, an exception needs to be raised.

Evaluation in Literature Regarding the deadline pattern, Combi et al. [6] consider the specification of a time when a certain activity must finish, expressed in absolute timestamp, being equivalent to a deadline constraint. The same definition can be found in [16]. Sadiq et al. [21] extend this pattern, by considering as deadline a constraint for the starting of an activity also. The same definition is used in [15], a deadline constraint being specified either by the earliest start date, latest start date or latest completion date.

As the definition presented by [15] is the most comprehensive, we adopt the same specification of a deadline in this paper; therefore, the deadline constraint is associated to an activity and can be expressed by the earliest start date, latest start date or the latest completion date. Moreover, we consider both relative and absolute time when specifying the deadline pattern.

3. Pattern T-RTCS (Repetitive time constraint with calendar support)

Description The ability to specify at design time that a certain activity needs to be executed following a certain periodicity rule over time.

Figure 10: Repetitive Time Constraint With Calendar Support Pattern

Related Patterns

- **T-D** - *Deadline*
- **T-SR** - *Schedule restricted*
- **T-TBR** - *Time based restriction*

Design Decisions The pattern is specified by providing a periodicity rule and by either providing a fixed interval in relative or absolute time in which the periodicity rule needs to be applied, by providing only the beginning/ending date starting

when/until when the periodicity rule is applied, by providing a fixed number of iterations for the periodicity rule or without providing any time bounds. Different granularities are considered.

Examples

- The treatment with antibiotics must be administered in 5 treatment cycles which are performed every 10 days; at the end of one treatment cycle the next one is scheduled (a fixed number of iteration and a periodicity rule are provided in this case).
- The Business Process Management Systems course is taking place every Thursday and Friday from 15:45 until 17:30 starting on 15 February 2011 and ending on 20 June 2011 (a time interval specified in absolute values and a periodicity rule have been provided in this case).

Motivation The repetitive time constraint with calendar support pattern associates periodicity rules to activities in a process, supporting in this way business processes which contain activities that need to be executed periodically and not necessarily at an equal distance in time. A workflow management system supporting this pattern should be able to notify resources involved in the execution of such an activity at the necessary points in time when the activity needs to be executed.

Implementation suggestions The repetitive time constraint with calendar support pattern implies that a certain work item is executed (namely it needs to enter the **started** state from the life cycle diagram, meaning that its execution has been actually started) following a specified periodicity rule.

In the case of this pattern, a calendar needs to be associated with the activities that are restricted by this constraint. The time instants when the respective activities need to be executed can be automatically, when the relevant points in the calendar can be completely determined at instantiation time (e.g. when a periodicity rule and a fully determined interval in absolute time are provided) or manually, with the support from users - when the next time instant when an activity needs to be executed is determined only at run time - (e.g. when a new date needs to be booked in the calendar after a particular activity has been completed) maintained in the calendar. The system needs to notify the resources involved at the correct times that the activity should be executed. In order to do this, the time elapsing between succeeding iterations can be monitored using timers. Moreover, the system can enforce resources to perform the work item at a specified time by automatically starting, at the right time instants, the work item for a specific resource, immediately after creation (see work item life cycle diagram, Figure 7).

If resources fail to comply with the constraint (do not start the activity during the times specified by the calendar or start it outside the calendar boundaries), exceptions are raised.

Evaluation in Literature The idea of expressing time in a periodic manner has been discussed in several papers [6, 8, 15, 16]. In [6, 8, 16], a repetitive time constraint is understood as a task that needs to be executed on a periodic timestamp, meaning on certain fixed dates. Therefore, the authors restrict the execution time of a task to certain fixed points (instants) (e.g. every Monday morning, every 5th workday of the month). Lanz et al. [15] extend the concept in their pattern named *Periodicity*, defining that an activity should follow a particular periodicity rule. Nevertheless, they make the distinction between the periodicity process elements (Time Pattern 10: Periodicity) and the cyclic elements (Time Pattern 9: Cyclic Elements), specifying that in the latter case the time lags between the cycles need to be considered. Based on the provided information, we see no significant difference between these two patterns, considering that they belong to the same category. Therefore, our aim is that the *Repetitive time constraint with calendar support* pattern should include both these situations.

Consequently, this pattern should be specified by either providing a time interval (e.g. from next Monday until the end of the year, between 24 December 2010 and 15 January 2011) using relative or absolute time, by providing a fixed number of iterations (e.g. for 5 consecutive weeks, for 10 treatment cycles), or without providing any time bound (e.g. every Monday at 11:00). Moreover, a periodicity rule needs to be present (e.g. every 12 hours for 5 subsequent days, every

3 business months, every Monday at 11:00). In most of the cases, the exact date of the next periodic execution of an activity is determined only during execution time, therefore we consider the support of a calendar for this pattern with the help of which the next time when an activity needs to be executed can be scheduled.

4. Pattern T-NTC (Negative time constraint)

Description The ability to specify at design time that a certain activity cannot be executed at a certain time.

Figure 11: Negative Time Constraint Pattern

Related Patterns

- n/a

Design Decisions This constraint is specified by providing a time instant or a time interval - expressed both in relative and absolute time and in a particular granularity - in which the execution of a certain activity is not possible.

Examples

- The registration of a customer can not be performed on Christmas Day (negative time constraint expressed as time instant - relative time).
- The Business Process Management Systems course is not held on 14 May 2011 (negative time constraint expressed as time instant - absolute time).

Motivation The negative time constraint pattern offers the ability to specify during the design process that a particular activity cannot be executed at a certain instant/interval in time. This type of pattern is useful for business processes running inside organizations where for e.g. free days are not considered in the life cycle of a case and activities cannot be executed in these days. A workflow management system supporting this pattern might enforce its application, avoiding to pass this decision to the resources involved or, to the contrary, it might leave the implicated resources to comply with this constraint and just notify them about it.

Implementation suggestions The negative time constraint pattern implies that a certain work item cannot be in the **started** state at the times specified by the constraint.

When considering the negative time constraint pattern and assuming that the system enforces it, then two additional paths need to be considered in the work item life cycle diagram (see Figure 7, red arrows symbols), namely the ones in which the system is able to suspend and resume a started work item automatically. These transitions are not supported in [20] and have been considered in this paper especially for the mentioned purpose. Therefore, in case the work item is created and has been already **started**, then during the time limits stated in the constraint the system shall freeze the current state of the work item, saving the performed work, and automatically suspending it, remaining that after the constraint expires, the execution of the work item to be automatically resumed to the same resource that was initially working on it. The states in which the work item might be found before reaching the **started** state, namely **created, offered** or **allocated** are not considered since we only take care that the work item is not in execution, namely in the **started** state, during the time specified by the constraint.

Another option is to leave the compliance of the constraint to the implicated resources which should not start or should suspend an already started work item during the time constraints specified in the pattern. In case the resources do not comply with the constraints, exceptions need to be raised.

Evaluation in Literature The need to specify time in a negative manner has been addressed in papers like [4, 13, 15]. Nevertheless, in [13] the need for specifying a negative constraint for a particular interval is only mentioned, without being captured in a specific pattern, while in [15], expressing time in a negative manner is addressed as an exceptional case for Time Pattern 10:

Periodicity and Time Pattern 5: Schedule Restricted Elements. Combi et al. [4] is the only author that introduces this constraint formally, giving it the name *Absolute Constraint* pattern.

Considering this pattern important on its own and not an exceptional case occurring when specifying schedules or repetitive constraints, this paper considers it independently.

5. Pattern T-SR (Schedule restricted)

Description The ability to specify at design time that a certain activity might be executed only in the bounds imposed by a particular schedule.

Figure 12: Schedule Restricted Pattern

Related Patterns

- **T-RTCS** - *Repetitive time constraint with calendar support*
- **T-TBR** - *Time based restriction*
- **T-VP** - *Validity period*

Design Decisions A schedule can be specified in terms of several discrete time points (execution is possible only at 11:00, 13:00 and 17:00 each day) or in terms of several time intervals (execution is possible every day between 11:00 and 17:00). Time is neither lower or upper bounded and takes relative values. The structure of the schedule is known at design time and the concrete times in the schedule are determined at instantiation time.

Examples

- The bank office is opened from Monday until Friday between 8:00 - 17:00 (schedule restricted expressed as several time intervals).
- Sending the receipts to the farmacie is possible every day at 11:00 or 15:00 o'clock (schedule restricted expressed as several discrete time points).

Motivation The schedule restricted pattern restricts the execution of activities in a process (work items being in the **started** state) to a particular schedule, supporting in this way business processes which contain activities that need to be executed following a fixed schedule. A workflow management system supporting this pattern should be able to notify resources involved in the execution of such an activity at the necessary points in time when the activity might be executed.

Implementation suggestions The schedule restricted pattern implies that a certain work item might be executed (namely it needs to enter the **started** state from the life cycle diagram) following a specified schedule. The main difference between this pattern and the *repetitive time constraint with calendar support* pattern is that in the case of the former, during the times imposed by the schedule, the work items might be executed but it is not necessarily that they must be executed on each time instant/frame specified in the schedule. Moreover, the concrete points in time of the schedule are always known at instantiation time which is not always the case of a calendar where dates can be booked in it at run time also.

A schedule needs to be associated with the activities that are restricted by this constraint. The system must notify the resources involved at the correct times when the activity might be executed. In order to do this, the time elapsing between succeeding time instants/frames can be monitored using timers. The system might also enforce this pattern in the sense that, in case work items are executed outside the schedule boundaries, the system can automatically suspend them until the time gets within the schedule boundaries (see work item life cycle diagram, Figure 7).

If resources start the execution of activities outside the schedule times, exceptions are raised.

Evaluation in Literature The Schedule restricted pattern has been considered in only two of the reviewed papers, namely [4, 15]. This constraint restricts the execution of an activity to a particular schedule, meaning that only in this time interval the activity can be executed; nevertheless, this

does not imply that the activity is necessarily executed in this period: for example, a clinic is opened from Monday to Friday from 9:00 to 18:00 each day; the registration of a new patient can only be performed in this period; however, there might be days in which no new patient comes to the clinic, therefore the registration task is not executed in that particular day.

For the *Schedule restricted* pattern, the schedule structure is fully determined at the instantiation time while the exact execution of a task is known only at run-time. No future execution times for an activity can be determined in advanced. The only restriction imposed by the workflow management system is that an activity restricted by a schedule is executed within the limits stated by that particular schedule.

6. Pattern T-TBR (Time based restriction)

Description The ability to specify at design time that a certain activity might be executed only a limited number of times during a determined period.

Figure 13: Time Based Restrictions Pattern

Related Patterns

- **T-RTCS** - *Repetitive time constraint with calendar support*
- **T-SR** - *Schedule restricted*

Design Decisions The period of time can be expressed in relative or absolute time intervals (e.g. between 5 December 2010 and 24 December 2010, per day, in 5 hours). The number of executions per time period also needs to be specified at design time. The symbol associated with this pattern can be seen in Figure 13 (n/TU = number of executions per Time Unit).

Examples

- Each customer can receive one free product from our shop per year (time based restriction expressed as a limited number of executions in a relative time interval).
- 10 CDs can be borrowed free of charge between 1 March 2011 and 1 April 2011, after this number of CDs borrowed, an amount of 10/CD will be charged (time based restriction expressed as a limited number of executions in an absolute time interval).

Motivation This pattern supports business processes which contain activities for which the number of executions needs to be limited within a given timeframe. A workflow management system supporting this pattern should be able to notify resources involved in the execution of such an activity when the number of executions equals the specified limit.

Implementation suggestions In order to implement this pattern, the system should monitor during run time the number of executions for the activities restricted by this constraint within the specified time interval. Therefore, each activity of this kind should have a counter and a calendar associated. Once a work item enters the **completed** state (its execution has been successfully completed), the counter is incremented. Moreover, if the timeframe is repetitive (e.g. per year), then once the first time interval expires, the system should reset the counter associated with the respective activity and start monitoring the number of executions for the new interval. In case resources execute the activity outside the specified timeframe or the number of executions within the timeframe exceeds the specified number, exceptions need also to be raised.

Evaluation in Literature This pattern has only been addressed by Lanz et al. [15] and implies that a particular process element can be executed for a limited number of times within a given timeframe.

7. Pattern T-VP (Validity period)

Description The ability to specify at design time that the entire life cycle of a work item is restricted by a validity period.

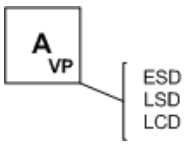

Figure 14: Validity Period Pattern

Related Patterns

- **T-SR** - *Schedule restricted*

Design Decisions The life time of a work item is restricted by a given validity period, which can be specified by the earliest start date (ESD), latest start date (LSD) or latest completion date (LCD). Time can be absolute or relative and the granularity issue needs to be considered.

Examples

- Starting from 10 January 2011, the new registration procedure applicable to foreign citizens needs to be followed (validity period expressed using the earliest start date).
- 25 December 2011 is the latest time when the current application form is used, from this day on the new application form will be considered (validity period expressed using the latest completion date).

Motivation This pattern is especially useful in the context of process evolution, when the process changes over time and activities must comply with these changes. A workflow management system supporting this pattern should help placing the life cycle of the concerning work items within the validity period. Moreover, it should notify resources about the constraint.

Implementation suggestions This constraint specifies that the life cycle of a work item (including all the intermediary states, from **creation** until **failure** or **completion**) must reside within the validity period. In case the validity period is expressed as an earliest start date, then the workflow system might enforce that the concerned work items can only enter the **created** state after this specified date. In case of a latest start date constraint, then the system might enforce that all the work items restricted by this pattern can be allowed to start their life cycles (be put in **created** state) at latest at this date, and after this date none of the concerned work items can be placed in the **created** state anymore. This means that the activities restricted by this constraint will stop being executed after this date. In case the pattern is specified through the latest completion date restriction, then the life cycle of the activities restricted by this pattern should end before this date (the corresponding work items should enter the **completed**/**failed** state before the specified date). No other instances of the same activities can be **completed** after this date. In case resources do not comply with this last constraint and continue executing work items after this date, exceptions are raised. In the case of the first two variants of the constraint, if we assume that the system enforces the constraints as described above, then resources cannot fail in complying with the restrictions. Otherwise, if the system does not enforce the constraints, then in case the life cycle of the concerned work items relies outside of the validity period, exceptions need to be raised.

Evaluation in Literature Workflow processes are dynamic entities, continuously subjected to changes having multiple causes (e.g. governmental rules and regulations applying to the process have changed, new tasks need to be introduced in the process, process redesign has been performed etc.). Sadiq et al. [21] have defined 5 types of process changes, namely flush, abort, migrate, adapt

and build. For more information concerning these 5 milestones which might occur to a workflow process we refer the reader to [21]. In order to gradually adapt to these type of changes - allow the cases already in execution to finish following the old process model and migrate the new cases to the new process model for example - a workflow management system should be able to cope with these situations.

Lanz et al. [15] have therefore understood the importance of a validity pattern which limits the life time of a process element to a given validity period.

8. Pattern T-TDV (Time dependent variability)

Description The ability to specify at design time that the control flow of the process will vary during execution due to time aspects (a specific path in the case of an explicit/implicit choice pattern (XOR-split/deferred choice) might be taken in certain time conditions while the other is taken for different time specifications; different instances of a subprocess might be activated depending on time conditions).

Related Patterns

- n/a

Design Decisions Time might be specified in absolute or relative manner and granularities should be considered.

Examples

- In the case of a bank transaction, if the money are deposited before 12:00 o'clock, then the transfer is finalized in the same day, otherwise the transfer is finalized in the next working day (deferred choice situation depending on time conditions).

Figure 15: Time Dependent Variability Pattern

Motivation There exist situations in real life processes in which the control flow of the process varies depending on certain time conditions. A workflow system supporting this pattern should be able to decide during run time which path in the process needs to be activated based on time aspects.

Implementation suggestions In order to implement this pattern, all the variation that the process might take based on time aspects need to be specified at design time. During run time, the system should be able to check its internal clock and based on the current time it must activate one path or another. The deferred choice situation is already implemented in some workflow systems with the help of time triggers.

Evaluation in Literature The time dependent variability pattern is used when the control flow of the process varies during execution due to time aspects. This pattern appears in [15].

9. Pattern T-TC2A (Time constraints between 2 activities)

Description The ability to specify at design time that a certain time lag between two activities (either sequential or arbitrary ones) needs to be respected; this also implies that the two activities must be executed in a certain order which is determined both by the control flow patterns and by the time pattern.

Related Patterns

- **T-TD** - *Task duration*
- **T-TC2E** - *Time constraints between 2 events*

Design Decisions The time lag is defined as a minimum value, a maximum value or an interval. The 7 possible ordering relations are: "before", "meets", "overlaps", "starts", "covers", "finishes" and "equals". These 7 ordering relations (resulting through the possible combinations between the begin/start of the two activities), produce 13 different cases for positioning the two tasks on the time axis, as 6 of the ordering relations (except "equals") create different situations when interchanging the two activities. In Figure 16, an exemplification of this pattern for two tasks A and B (the events considered are: start of activity A: As, completion of activity A: Ae, start of activity B: Bs, completion of activity B: Be) has been detailed. In the left part of the picture, the 4 events considered are positioned on the time axis. On the right part, the symbols associated with each of these situations are introduced. Attached to the symbol that marks the ordering relation between the two tasks, the time lag specified in a certain granularity might be given. Only 7 situations are presented as interchanging A and B has not been considered, therefore 6 cases have not been included.

Examples

- The maximum time lag between receiving the necessary documents are issuing a response to the client is 30 days ("before" ordering relation, time lag specified through a maximum value).
- The time between finishing to administrate the treatment to the patient and starting the surgery is between 2 and 3 weeks ("before" ordering relation, time lag specified through a time interval).

Motivation In real life business processes there are often cases in which time lags between activities have to be complied as they are imposed by existing rules or regulations. A system implementing this pattern should be able to support its users in respecting these constraints.

Implementation suggestions The 13 possible relations between 2 activities resume to the relations existing between the states **started** and **completed** of the 2 corresponding work items. Therefore, a workflow system implementing this pattern should be able to monitor the time elapsing between these events and check its compliance with the specified constraints. For example, in the case of a "before" ordering relation between 2 activities A and B, the system might start a timer as soon as the work item corresponding to activity A enters the **completed** state and monitor time until the work item corresponding to activity B enters the **started** state. In case the elapsing time between these 2 events complies with the time lag specified in the pattern, then the pattern is respected. Otherwise, an exception needs to be raised.

Evaluation in Literature Specifying a time constraint between two activities is together with the duration pattern, one of the most common patterns studied by academia (see Table 1).

In [21], this pattern is defined as being equivalent to the distance between the end/start of two activities, specified in relative time. In papers like [4, 8, 15] the definition is further extended by introducing upper bound constraints, lower bound constraints and in the case of [15] even intervals.

When specifying this time pattern, we decided to follow the classification provided in [14], where 7 ordering relations between two activities, resulting from the possible combinations between the start/end of the two activities, are described. This is the most comprehensive classification we found in literature. Beside specifying an ordering relation for the two activities, a time lag is also considered.

10. Pattern T-TC2E (Time constraints between 2 events)

Description The ability to specify time lags between arbitrary events taking place during workflow execution. As in a process model, a time constraint between two arbitrary events (other than

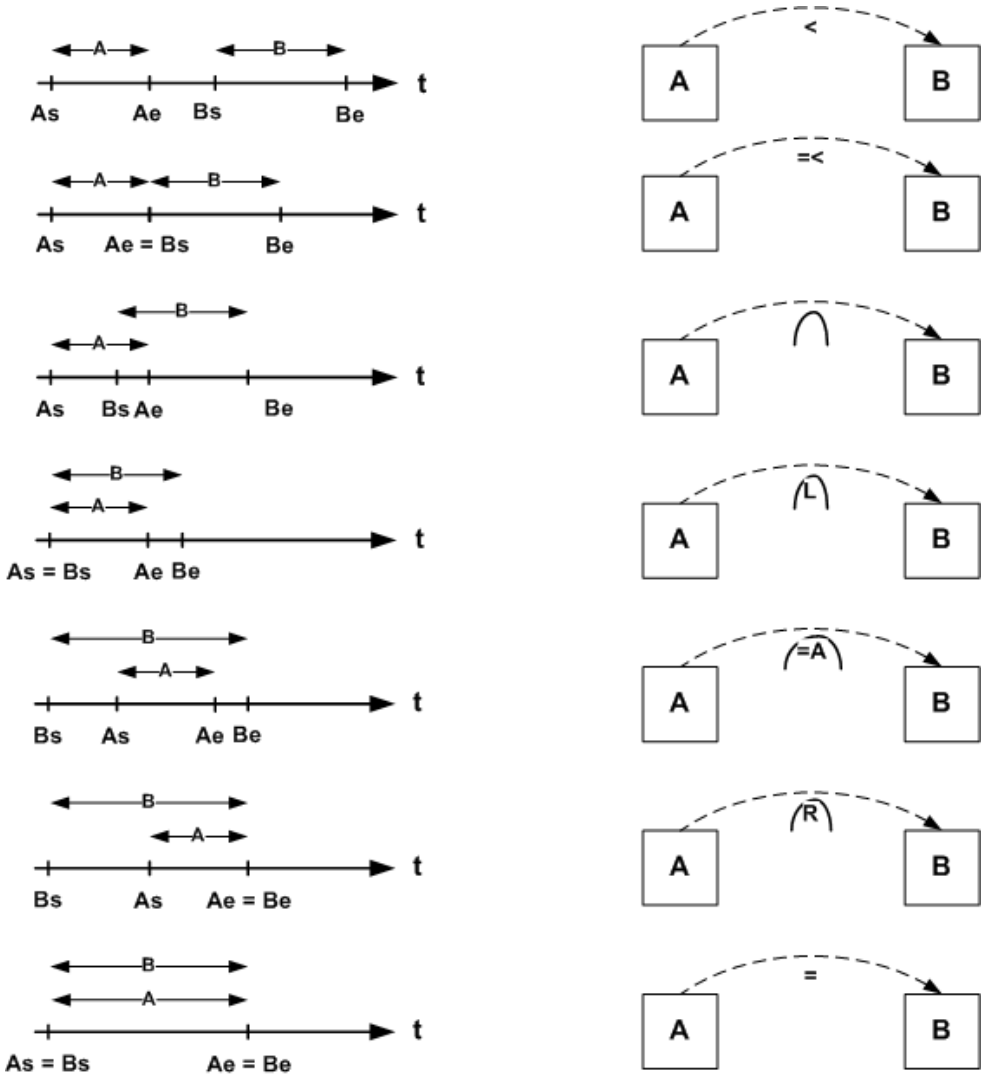

Figure 16: Time Constraints between Two Activities Pattern

the start/end of a task) cannot be captured, no special symbol is defined for this situation.

Related Patterns

- **T-TD** - *Task duration*
- **T-TC2A** - *Time constraints between 2 activities*
- **T-RD** - *Required delay*

Design Decisions Specifying time constraints between two arbitrary events is made through time lags, defined in terms of a minimum value, a maximum value or a time interval. Time might be relative or absolute and a particular granularity is used.

Examples

- The time between an application letter is received and the candidate is contacted is at maximum 2 weeks (time between an external message trigger and the completion of an activity, specified as a maximum time lag).

Motivation The need of having time lags between arbitrary events taking place during run time in workflow management systems arises as a consequence of the requirements imposed by existing rules and regulations. A system implementing this pattern should be able to support its users in respecting these constraints.

Implementation suggestions As the **starting** and **completion** of a task are events taking place during execution time, we can generalize the *Time constraints between 2 activities* pattern definition by considering two arbitrary events happening during execution in a workflow management system (e.g. external triggers, allocation of a work item to a resource, instantiation or completion of a case, etc.). Monitoring time between such events can be done with the help of timers, in a similar way as described for the *Time constraints between 2 activities* pattern. In case the pattern is not complied with, exceptions are raised.

Evaluation in Literature Lanz et al. [15] is the only author to define a special time pattern dealing with two arbitrary events (Time Pattern 3: Time Lags between Arbitrary Events). Nevertheless, in [6], a particular case of this pattern is discussed when the action of allocating an activity to a human resource, performed by the scheduler component of the system, is taken into consideration. In this situation, the events: the scheduler starts the operation of selecting the appropriate agent to perform a certain task, the scheduler has completed the operation and has placed the work item in the work list of the selected agent, are considered. By not imposing a certain ordering relation between the start/end of activities (allowing for negative time specifications), Bettini et al. [3] use the same notion of setting time constraints between arbitrary events.

Events might be internal to the process (e.g. the start of a task and the completion of a task) or external (e.g. the arrival of an external message). The possible relations between the events start/completion of tasks have been used in specifying the *Time constraints between two activities* pattern, therefore, as in the current case, all possible events are considered, we regard the *Time constraints between 2 events* pattern as a generalization of the previously discussed *Time constraints between two activities* pattern.

11. Pattern T-RD (Required delay)

Description The ability to specify at design time that two adjacent activities need to be executed at a fixed distance in time.

Related Patterns

- **T-TC2E** - *Time constraints between 2 events*

Design Decisions Time should be specified as a duration in a specific granularity. The symbol associated with this pattern can be seen in Figure 17.

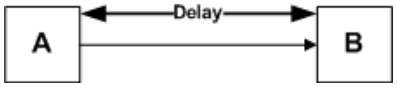

Figure 17: Required Delay Pattern

Examples

- An employee from the European office of a company needs to send a report for further processing to one of his colleagues from the office of the same company situated in America; in this case the system needs to wait for a time equivalent with the time zone difference between these two locations before allowing the case execution to continue.

Motivation The need for two succeeding activities in a business process to be separated by a fixed difference in time might be caused by time zone differences in the case of cross-organizational processes in which the organizations involved in the process are located in different countries. A system supporting this pattern should be able to enforce its behavior.

Implementation suggestions The required delay pattern specifies that a fixed distance in time should exist between two adjacent activities. This means that, from the moment the work item corresponding to the first activity enters the **completed** state and until the second work item,

corresponding to the second activity, is allowed to enter the **created** state, a fixed amount of time must pass. The workflow management system must enforce this constraint by not placing the second work item in the **created** state before the specified time elapses. This can be achieved by starting a timer as soon as the first work item is **completed**. No exceptions are applicable in this case as the workflow system enforces the pattern.

Evaluation in Literature The need for the workflow management system to wait for a certain time period before allowing the enablement of the succeeding activity has been discussed in [4, 16]. Combi et al. [4] calls this type of contraint *Required Delays* and considers it not to be temporal constraint, due to the fact that if such a constraint is specified, the system automatically waits for the indicated period before allowing the case to continue its execution. This constraint can be specified between any two activities in the process, and in this paper we extend this specification to two arbitrary events taking place during workflow execution.

Li et al. [16] mention a similar constraint taking place during execution time, when a system needs to wait for a certain amount of time between allowing for the case execution to continue. The authors identify this constraint for the *Time Difference* pattern, where two activities are placed in different time zones and therefore the system needs to wait in order for the time zone of the successor activity to become working time.

As the second example is a particular case for the more general *Required Delays* pattern, we choose to introduce this particular constraint in our patterns collection.

4.2.2 Evaluation Results

The discussed patterns are the result of a literature review comprising the most important authors researching the time patterns subject. Table 1 presents the correlations between the identified patterns and the reviewed literature.

A two scale assessment is used with "x" indicating that the corresponding authors have discussed the respective pattern and "-" indicating no recognition of the pattern in the reviewed article. As it can be seen in the table below, the majority of the patterns included receive simultaneous support from several authors. Moreover, the current work does not consider patterns that did not receive previous support in literature.

Table 1: Summarization of Time Patterns

Names of the Patterns	[3]	[4]	[6]	[8]	[9]	[10]	[13]	[14]	[15]	[16]	[18]	[21]
T-TD	x	x	x	x	x	x	x	-	x	x	x	x
T-D	-	-	x	-	-	-	-	-	x	x	-	x
T-RTCS	-	-	x	x	-	-	-	-	x	x	-	-
T-NTC	-	x	-	-	-	-	x	-	x	-	-	-
T-SR	-	x	-	-	-	-	-	-	x	-	-	-
T-TBR	-	-	-	-	-	-	-	-	x	-	-	-
T-VP	-	-	-	-	-	-	-	-	x	-	-	-
T-TDV	-	-	-	-	-	-	-	-	x	-	-	-
T-TC2A	x	x	x	-	x	-	x	x	x	x	-	x
T-TC2E	x	x	x	-	-	-	-	-	x	-	-	-
T-RD	-	x	-	-	-	-	-	-	-	x	-	-

4.3 Temporal Example: Loan Application

The Loan Application process introduced in chapter 2 is revisited in this section. The text of the process description is presented again, this time the time patterns existent in the text (7 out of 11 patterns) being emphasized.

The process of applying for a loan is modeled for a bank situated in Europe. The process starts when a client comes to one of the bank offices and requests a loan. *The bank office is opened every day from Monday to Friday from 9:00 A.M. until 18:00 P.M.* (**Schedule restricted pattern**) *except the legal holidays* (**Negative time constraint pattern**). The client is received by a front office desk employee and is given a form to fill in. *This operation takes between 5 and 10 minutes* (**Task duration pattern** specified through a time interval). After this, *the client is requested to wait no more than 20 minutes until a credit officer becomes available and as soon as this happens the client is assigned to the available credit officer* (**Time constraints between 2 activities pattern**). *Immediately* (**Time constraints between 2 activities pattern**) *a meeting of maximum one hour* (**Task duration pattern** specified through a maximum value) starts, where the credit officer explains to the client the procedure that needs to be followed and the documents that need to be provided to the bank. *After the meeting, the bank waits for the needed documents. If the documents are not received in 10 business days, then the loan request is rejected. If the documents are received in time, a report is made by the credit officer* (**Time dependent variability pattern** applied for a deferred choice situation). The report is sent for approval to the central bank office situated in America. *The time zone difference between the two locations is of 10 hours* (**Required delay pattern** applied for a time zone difference). As soon as an answer is received from the central office, the request is further processed: either it is rejected or it is accepted. If the request is accepted, a positive answer is sent to the customer together with the payment conditions. *This activity is not performed during the legal holidays* (**Negative time constraint pattern**). After the loan is accepted, the customer, who has received the loan, *starts paying regular interest rates to the bank to gradually return the money. This is done according to a pre-established timetable between the customer and the bank* (**Repetitive time constraint with calendar support pattern**). When the payment is completed, or in the eventuality that the request is rejected, the case is archived and the process ends. *The Service Level Agreement between the customer and the bank states that the maximum allowed distance between the meeting with the credit officer and the positive or negative answer sent to the customer should be between 20 and 30 business days* (**Time constraints between 2 activities pattern**).

The model of this process description, incorporating the time constraints, is presented in Figure 18.

5 Important Aspects Related to Time Patterns Support in Workflow Management Systems

Considering time in workflow management systems implies, apart from defining a relevant set of time patterns that need to be supported by the system, dealing with some important aspects that need to be considered as: **multi-granularity of time**, **consistency checking** and **scheduling algorithms**.

Time is specified in natural language using **multiple time measures or granularities** (e.g. days, hours, business weeks etc.). Some of the reviewed authors consider dealing with time constraints in workflow management using a single basic granularity (e.g. [6, 21]), having the assumption that the conversion from one granularity to another is equivalent. Nevertheless, conversion between granularities is not always determined, problems arising for the granularities with time ticks of different length (e.g. months) or for the granularities with gaps among ticks (e.g. business days) [2]. Bettini et al. [2] considered working with multiple granularities when specifying time constraints for workflows. In [11], an algorithm for transforming from one granularity to another is formally described. The authors argue that converting from a coarser to a finer granularity cannot always be realized without loss of information (e.g. converting from months into days) and suggest to include the coarser granularity to an interval of the type [lower bound, upper bound] specified in the finer granularity. In [17], another model for converting from one granularity to another is considered. Defining a general algorithm to allow time constraints in workflow processes to be specified in multiple granularities needs to be further researched.

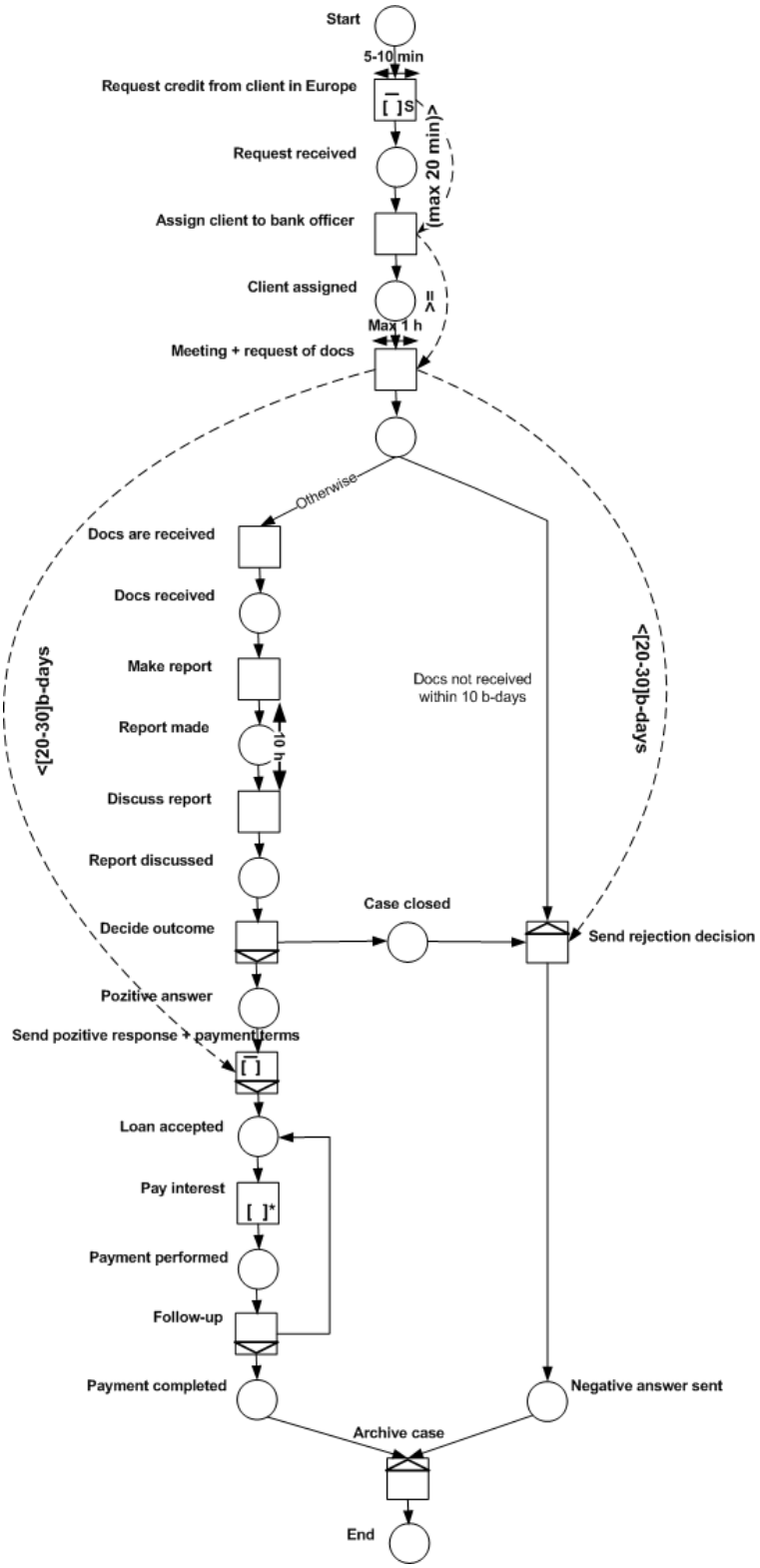

Figure 18: Temporal Loan Application Process

Another important aspect that needs to be considered when dealing with time constraints in workflow management is **consistency checking**. A given workflow model is consistent if the temporal constraints set can be realized based on the syntax of the model and considering all the time patterns used [21]. In [3], the consistency of a workflow graph is checked at the definition stage of the process and an algorithm is proposed for this. Sadiq et al. [21] further extend the concept of consistency checking by realizing that the temporal constraints specified for a workflow model need to be verified at different points in time, namely during workflow modeling (at build time), during instantiation time and during execution. The same approach is introduced in [8]. This happens due to the fact that time constraints defined in relative time can be fully converted in absolute time either at instantiation time or during execution time, problem which is named in literature as the *temporal uncertainty*. This clearly shows that checking consistency affects two of the three dimensions of a workflow, namely the process and the case dimension.

The next step after consistency checking is to provide the resources involved in the process with a **schedule** in which the temporal bounds when tasks involved in the process can be executed, respecting the temporal constraints, are specified. Schedules show that the resource dimension of a process is also affected when dealing with time in workflow management. Scheduling algorithms have been proposed in papers like [3, 7, 10, 16]. Moreover, a classification of scheduling algorithms applicable in dynamic environments (workflow management systems are highly dynamic environments), can be found in [22]. Providing employees with schedules has several advantages: employees are offered the possibility to plan their work ahead as information about the future activities allocated to participants is provided early, the overload of certain agents might be discovered and a better allocation procedure might be considered, idle participants can easily be detected, possible constraints violations due to capacity bottlenecks might be observed when computing the schedules, and so on. Eder et al. [10] propose an algorithm for personal schedules based on a probabilistic time graph. In [6], schedules in a workflow system are classified in free schedules (agents can use any amount of time for completing an activity, as long as they start the activity in the time interval specified in the schedule), restricted due-time schedules (an upper bound for the time that agents have to execute activities is set in the schedule) or bounded schedules (agents might be forced to use more time than the declared minimum duration for executing an activity and less than the declared maximum duration in order to satisfy the time constraints). Nevertheless, each author discussing about this problem has considered only a limited number of patterns in his work and has built a scheduling algorithm working only for that limited amount of patterns. Therefore, an algorithm valid for an extended collection of time patterns needs to be defined.

All in all, dealing with time in workflow management has several and complex implications and future efforts need to be done in order to clarify all these aspects and to fully define a time aware workflow management system.

6 Conclusions

The need for workflow processes extended with time support is becoming more and more acute in today's organizations. Several authors have considered time patterns in their studies, nevertheless the majority of them addressed only a limited set of them. In this paper, we provided a comprehensive list of time patterns that have been considered in the specialized literature in the last approximately 10 years. Furthermore, we provided symbols for each of these patterns and extended an already existing workflow conceptual model with time support. A detailed review of the literature dealing with time constraints has been given, together with a short description for the most important problems that need to be faced when willing to create time aware workflow management systems.

As future work, algorithms for consistency checking that consider the entire collection of time patterns need to be addressed.

References

[1] Van Der Aalst. The application of petri nets to workflow management, 1998.

[2] Claudio Bettini, X. Sean Wang, and Sushil Jajodia. A general framework for time granularity and its application to temporal reasoning. *Annals of Mathematics and Artificial Intelligence*, 22:29–58, 1998.

[3] Claudio Bettini, X. Sean Wang, and Sushil Jajodia. Temporal reasoning in workflow systems. *Distributed and Parallel Databases*, 11:269–306, 2002.

[4] C. Combi, M. Gozzi, J.M. Juarez, B. Oliboni, and G. Pozzi. Conceptual modeling of temporal clinical workflows. In *Temporal Representation and Reasoning, 14th International Symposium on*, pages 70 –81, 2007.

[5] C. Combi and G. Pozzi. Task scheduling for a temporalworkflow management system. In *Temporal Representation and Reasoning, 2006. TIME 2006. Thirteenth International Symposium on*, pages 61–68, 2006.

[6] Carlo Combi and Giuseppe Pozzi. Temporal conceptual modelling of workflows. In *Conceptual Modeling - ER 2003*, volume 2813 of *Lecture Notes in Computer Science*, pages 59–76. Springer Berlin / Heidelberg, 2003.

[7] Carlo Combi and Giuseppe Pozzi. Architectures for a temporal workflow management system. In *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pages 659–666, New York, NY, USA, 2004. ACM.

[8] Johann Eder, Euthimios Panagos, and Michael Rabinovich. Time constraints in workflow systems. In *Advanced Information Systems Engineering*, volume 1626 of *Lecture Notes in Computer Science*, pages 286–300. Springer Berlin / Heidelberg, 1999.

[9] Johann Eder and Horst Pichler. Duration histograms for workflow systems. In *Proceedings of the IFIP TC8 / WG8.1 Working Conference on Engineering Information Systems in the Internet Context*, pages 239–253, Deventer, The Netherlands, The Netherlands, 2002. Kluwer, B.V.

[10] Johann Eder, Horst Pichler, Wolfgang Gruber, and Michael Ninaus. Personal schedules for workflow systems. In *Proceedings of the 2003 international conference on Business process management*, BPM'03, pages 216–231, Berlin, Heidelberg, 2003. Springer-Verlag.

[11] Iqbal Goralwalla, Yuri Leontiev, M.Tamer zsu, Duane Szafron, and Carlo Combi. Temporal granularity: Completing the puzzle. *Journal of Intelligent Information Systems*, 16:41–63, 2001.

[12] D. Hollingsworth. Workflow management coalition - the workflow reference model. Technical report, Workflow Management Coalition, 1995.

[13] Heinrich Jasper, Olaf Zukunft, and Helge Behrends. Time issues in advanced workflow management applications of active databases, 1995.

[14] Eleanna Kafeza and Kamalakar Karlapalem. Gaining control over time in workflow management applications. In *Proceedings of the 11th International Conference on Database and Expert Systems Applications*, DEXA '00, pages 232–242, London, UK, 2000. Springer-Verlag.

[15] Andreas Lanz, Barbara Weber, and Manfred Reichert. Workflow time patterns for process-aware information systems. In *Enterprise, Business-Process and Information Systems Modeling*, volume 50 of *Lecture Notes in Business Information Processing*, pages 94–107. Springer Berlin Heidelberg, 2010.

[16] Weiping Li and Yushun Fan. A time management method in workflow management system. In *Grid and Pervasive Computing Conference, 2009. GPC '09. Workshops at the*, pages 3 –10, 2009.

[17] Jianxun Liu, Chunjie Zhou, and Jian Cao. An integrated time management model for distributed workflow management systems in grid environments. *Concurr. Comput. : Pract. Exper.*, 21:2084–2098, 2009.

[18] W.M.P.van der Aalst A.J. Moleman R.S. Mans, N.C. Russell and P.J.M. Bakker. Schedule-aware workflow management systems, 2009.

[19] Nick Russell and Arthur H. M. Ter Hofstede. Exception handling patterns in process-aware information systems.

[20] Nick Russell, Wil M.P. van der Aalst, Arthur H.M. ter Hofstede, and David Edmond. Workflow resource patterns: Identification, representation and tool support. In Oscar Pastor and Joo Falco e Cunha, editors, *Advanced Information Systems Engineering*, volume 3520 of *Lecture Notes in Computer Science*, pages 11–42. Springer Berlin / Heidelberg, 2005.

[21] S.W. Sadiq, O. Marjanovic, and M.E. Orlowska. Managing change and time in dynamic workflow processes. *International Journal of Cooperative Information Systems*, 9(1-2):93–116, 2000.

[22] V. Suresh and Dipak Chaudhuri. Dynamic scheduling–a survey of research. *International Journal of Production Economics*, 32(1):53–63, 1993.