Workflow Time Patterns for Process-Aware Information Systems

Andreas Lanz[1], Barbara Weber[2], and Manfred Reichert[1]

[1] Institute of Databases and Information Systems, Ulm University, Germany
{Andreas.Lanz,Manfred.Reichert}@uni-ulm.de
[2] Quality Engineering Research Group, University of Innsbruck, Austria
Barbara.Weber@uibk.ac.at

Abstract. Formal specification and operational support of time constraints constitute fundamental challenges for any process-aware information system. Although temporal constraints play an important role in the context of long-running business processes, time support is limited in existing process management systems. By contrast, different kinds of planning tools (e.g., calendar systems, project management tools) provide more sophisticated facilities for handling task-related time constraints, but lack operational support for business processes. This paper presents a set of time patterns to foster systematic design and comparison of the different technologies in respect to the time perspective. These time patterns are all based on empirical evidence from several large case studies. Their widespread use will contribute to further maturation of process-aware information systems and related evaluation schemes.

1 Introduction

Formal specification and operational support of the time perspective constitute fundamental challenges for any enterprise information system. Although temporal constraints play an important role in the context of long-running business processes (e.g., patient treatment, automotive engineering, flight planning) [1,2,3], support of the time perspective is rather limited in existing process management systems [1,4] (as opposed to other process perspectives like control flow and data flow). By contrast, different kinds of planning tools (e.g., calendar systems and project management tools) provide more sophisticated facilities for handling time constraints (e.g., periodic activities), but miss an operational support for business processes. So far, there is a lack of methods for systematically assessing and comparing the time capabilities provided by these different process support technologies (denoted as *Process-Aware Information Systems* (PAIS)).

To make PAIS better comparable and to facilitate selection of appropriate PAIS-enabling technologies, workflow patterns have been introduced [5,6,7]. Respective patterns provide means for analyzing the expressiveness of process modeling approaches in respect to different process perspectives. For example, proposed workflows patterns cover control flow [5], data flow [6], exceptions [8], and process change [7]. However, a framework for systematically evaluating PAIS in

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 94–107, 2010.

respect to their ability to deal with the time perspective is missing and is picked up by this paper. Our contribution is as follows: We suggest *time patterns* to foster comparison of existing PAIS with respect to their ability to cover the time perspective of processes. The proposed time patterns complement existing workflow patterns and have been systematically identified by analyzing a large collection of processes in healthcare, automotive engineering, aviation industry, and other domains. The presented work will not only facilitate PAIS comparison in respect to the support of time constraints, but also foster selection of appropriate time components when designing PAIS.

Section 2 summarizes basic notions. Section 3 presents the research method employed for identifying the time patterns. Section 4 describes the proposed time patterns sub-dividing them into 4 categories. We present related work in Section 5 and conclude with a summary and outlook in Section 6.

2 Basic Notions

This section describes basic concepts and notions used in this paper: A *process management system* is a specific type of information system which provides process support functions and separates process logic from application code. For each business process to be supported, a *process type* represented by a *process schema* has to be defined (cf. Fig. 1). In the following, a process schema corresponds to a directed graph, which comprises a set of *nodes* – representing *activities* and *control connectors* (e.g., XOR-Splits or AND-Joins) – and a set of *control edges* between them. The latter specify precedence relations. We further use the notion of *activity set* to refer to a subset of the activities of a process schema. Its elements are not required to be part of a sequence block, but may also belong, for example, to different parallel branches. During run-time *process instances* are created and executed according to a predefined process schema S. *Activity instances*, in turn, represent executions of single process steps of a particular process instance. Activities which shall be executed more than once (concurrently or sequentially) are referred to as *multi-instance activities.*

The patterns introduced in the following can be applied to these granularities, i.e., process schema, activity, activity set, activity instance, and process instance. We use the term *process element* as umbrella for all these concepts.

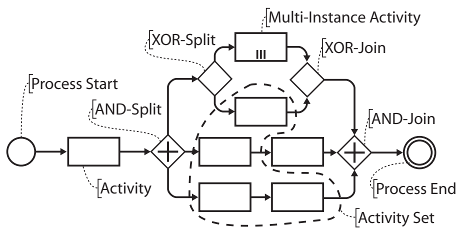


Fig. 1. Core Concepts of a Process Model

3 Research Method

The overall goal of this paper is to complement existing workflow patterns with a set of time patterns suitable to assess how effectively PAIS can deal with time. As motivated in the introduction, adequate modeling and management of temporal constrains will be a key feature of future PAIS, particularly regarding the support of long-running processes involving humans (e.g., patient treatment).

We describe the selection criteria for our time patterns, the data sources they are based on, and the procedure we have applied for pattern identification.

Selection Criteria. We consider patterns covering temporal aspects relevant for the modeling and control of processes and activities respectively. Our focus is on high coverage of real-world scenarios, and not on specific time features of a PAIS like verification of time constraints [4,1,2], escalation management [9], or scheduling support [10,11].

Sources of Data and Data Collection. As sources for our patterns we consider results of case studies we performed in different domains.

One of our data sources is a large *healthcare* project in which we designed core processes of a Women's Hospital [12]. Selected processes were implemented using existing workflow technology. As part of this project time aspects were elicited and documented. In total we consider 98 process models covering both administrative processes (e.g., order handling) and treatment processes (e.g., chemotherapies and ovarian cancer surgery).

As second data source we use process models from *automotive industry*. We consider a case study on electronic change management (ECM) [13]. Corresponding models have been published by the German Association of the Automotive Industry (VDA) [13]. In total this project provides 59 process models.

As third data source serves a case study we conducted with an *on-demand air service*. As part of this project we analyzed flight planning and handling post flight phases. As aviation industry is highly regulated, compliance with standards and regulations, in addition to company policies, is essential (e.g., minimum standards for flight time limitations, or rest time regulations). Many of these regulations contain time constraints to be obeyed.

Our fourth data source are *healthcare processes* from a large Medical University Hospital. It comprises 60 different processes, related to diagnostic and therapeutic procedures in the field of internal medicine (e.g., examinations in medical units like radiology, gastroenterology, and clinical chemistry). Finally, we have deep insight into patient scheduling systems.

Pattern Identification Procedure. To ground our patterns on a solid basis we first create a list of candidate patterns. For this purpose we conducted a detailed literature review and rely on our experience with PAIS. Next we thoroughly analyzed the above mentioned material to find empirical evidence for our time patterns and - if necessary - extend the pattern candidate list. As a pattern is defined as reusable solution to a commonly occurring problem we require each of our time patterns to be observed at least three times in different models of our samples. Therefore, only those patterns, for which enough empirical evidence exists, are included in the final list of patterns.

4 Time Patterns

As result of our analysis we have identified 10 different patterns which we divide into 4 distinct categories (cf. Fig. 2a). These time patterns constitute solutions for realizing commonly occurring time aspects in PAIS. Pattern Category I (*Durations and Time Lags*) provides support for expressing durations of process elements (e.g., activities) as well as time lags between events (e.g. milestones) or activities. Pattern Category II (*Restrictions of Process Execution Points*) allows specifying constraints regarding possible execution points of process elements (e.g., activity deadline). Category III (*Variability*) provides support for time based variability (e.g., control-flow varies depending on time context). Finally, Category IV (*Recurrent Process Elements*) comprises patterns for supporting recurrent process elements (e.g., periodicity and cyclic flows). Due to lack of space only 7 out of 10 patterns will be described in detail. For the remaining patterns we refer to our technical report [14].

Pattern Catalogue
Category I: Durations and Time Lags
TP1: Time Lags between Activities
TP2: Durations
TP3: Time Lags between Events
Category II: Restrictions of Process Execution Points
TP4: Fixed Date Elements
TP5: Schedule Restricted Elements
TP6: Time Based Restrictions
TP7: Validity Period
Category III: Variability
TP8: Time Dependent Variability
Category IV: Recurrent Process Elements
TP9: Cyclic Elements
TP10: Periodicity

Fig. 2a. Pattern Catalogue

General Design Choices
A.) Parameters of a pattern may be set at different time points a.) At build-time (i.e., during process modeling) b.) At instantiation time (i.e., when a process instance is instantiated) c.) At run-time (i.e., during process execution)
B.) Time parameters can be specified in different time granularities a.) Basic (i.e., years, months, weeks, days, hours, minutes, seconds) b.) System-defined (e.g., business days) c.) User-defined (e.g., Wednesday afternoon)
C.) Patterns can be applied to different process elements a.) Single activity (including multi-instance activities) b.) Activity set c.) Process model d.) Set of process instances

Fig. 2b. General Design Choices

Fig. 2a gives an overview of the time patterns. For each discussed pattern we provide a name, synonyms, a brief description of the addressed problem, design choices, remarks regarding its implementation, examples from our case studies, a reference to related patterns, and known uses of the pattern summarized in a table (cf. Fig. 4 - Fig. 12).

In particular, *design choices* allow for parameterizing time patterns keeping the number of distinct patterns manageable. Design choices not only relevant for a particular pattern, but for several ones, are described only once. Typically, existing PAIS do not support all design choices regarding a specific pattern. We denote the combination of design choices supported by a particular approach as *pattern variant*.

Fig. 2b describes three design choices concerning the point in time when temporal constraints are set, the time granularities supported and the process elements to which the respective pattern can be applied. These design choices are valid for several or all of the patterns, and can be used for parameterizing them. If not all options of a design choice are valid for a time pattern this is described with the respective pattern. Additional design choices, only relevant for a specific pattern or pattern category, are provided with the respective description.

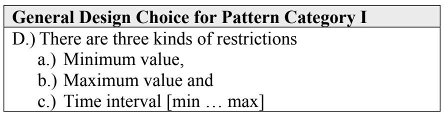
To formalize operational semantics of the time patterns we first introduce the notion of *temporal execution trace* (*trace* for short): We assume that all events related to the execution of a process instance[1] are recorded in a trace together with a timestamp designating their time of occurrence. Informally, temporal execution traces are defined as follows (for a formal definition see [14]):

Definition 1 (Temporal Execution Trace)

1. *An event occurrence is a tuple* $\varphi = (e, t)$ *consisting of event* e *and timestamp* t*, where* t *defines the exact point in time at which event* e *occurred.*
2. *A temporal execution trace* $\tau_S =< \varphi_1, \ldots, \varphi_n >$ *is an ordered set of event occurrences* φ_i*, where the order of* φ_i *in* τ_S *reflects the temporal order in which the events occurred during process execution*[2]*, i.e.*
$$\varphi_k, \varphi_j \textit{ with } k < j \Rightarrow t_k < t_j.$$
3. $occurrences(S, e, \tau_S) = \{\varphi \in \tau_S | \varphi = (e, \cdot)\}$ *corresponds to all occurrences* $\varphi = (e, \cdot)$ *of event* e *within trace* τ_S *on process schema* S*.*

For each time pattern we provide a description using the aforementioned schema (cf. Fig. 4 - Fig. 12). Additionally, we define pattern semantics by characterizing the traces τ_S that can be produced when executing any instance of process schema S while satisfying the time constraints expressed by the patterns.

Pattern Category I (Durations and Time Lags). Our first category comprises three time patterns expressing durations of process elements as well as time lags between them. Design Choice D constitutes a general design choice valid for all patterns from this category. It describes whether time lags are specified in terms of minimum/maximum values or time intervals (cf. Fig. 3).



General Design Choice for Pattern Category I
D.) There are three kinds of restrictions a.) Minimum value, b.) Maximum value and c.) Time interval [min … max]

Fig. 3. General Design Choices for Category I

[1] Including start and end events of the activities.

[2] We assume that events in τ_S do not occur at the exactly same point in time.

Pattern TP1 (*Time Lags between two Activities*). This pattern is described in Fig. 4. It enables definition of different kinds of time lags between two activities. Informally, semantics of pattern TP1 is as follows: A time lag between activities A and B can be mapped onto a time lag between their start/end events e_A and e_B. This way, for example, start/start as well as end/start time lags can be described (see Design Choice E in Fig. 4). Compliance of a given trace with such time lag now means that all occurrences $\varphi = (e_A, t_A)$ and $\psi = (e_B, t_B)$ of the two events fulfill the given time lag. Note that this also needs to be valid in connection with loops. As illustrated in Fig. 5, considering time lags, for which one of the activities resides inside a loop and the other one outside that loop (e.g. A1 and A3 or A3 and A6 in Fig. 5) is problematic due to unclear semantics (for details see [14]). This becomes even more complicated when considering nested loops. For example a time lag between activities A_1 and A_6 in Fig. 5 could relate to the first iteration, the last iteration, every iteration or any special iteration

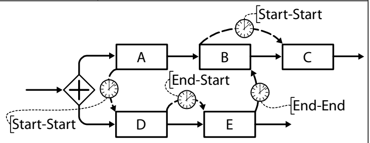
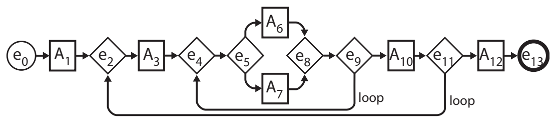
Time Pattern TP1: Time Lags between two Activities	
Also known as	Upper and Lower Bound Constraints, Inter-Task Constraints, Temporal Relations
Problem	There is a given time lag between two activities which needs to be respected. Time Lags may not only exist between succeeding activities, but also between arbitrary ones. Time lags are often required to comply with existing rules and regulations. The time lag may or may not have binding character.
Design Choices	D.) Time Lags may represent all three kinds of restrictions (cf. Fig. 3) E.) Time Lags can be realized based on four different time relations a.) Between start of two activities (i.e., Start-Start relation) b.) Between start of the first and completion of the second activity (i.e., Start-End) c.) Between completion of the first and start of the second activity (i.e., End-Start) d.) Between completion of two activities (i.e., End-End)
Solution	A time constraint is introduced between the start and / or end event of the two activities. Timers may be used to realize this pattern at runtime. For example, to realize an end-start relation, the timer starts after completing A. If the time lag between A and B is a  Start-Start A B C End-Start End-End Start-Start D E minimum time lag, B may only be started after the timer has expired. Depending on whether a time lag has binding character the activation of the activity may be delayed until the time lag is satisfied. If the time lag is a maximum time lag B may be started as soon as the timer is started until its expiry. In case the timer expires an exception is raised. For time intervals both of the above cases apply.
Context	The mechanism evaluating the constraint (i.e., starting the timer) needs to be able to access the value of the time lag when it determines the impact of the constraint.
Examples	• The *maximum time lag* between discharge of a patient from a hospital and sending out the discharge letter to the general practitioner of the patient should be 2 weeks (Design Choices D[b] E[d]) • Patients must not eat *at least 12 hours* before a surgery takes place. The latest point in time where the patient can have a meal is determined by the date of the surgery (Design Choices D[a] E[c]) • A contrast medium has to be administered *2 to 3 hours before* a radiological examination. The interval in which the contrast medium should be administered depends on the examination date (Design Choices D[c] E[a])
Related Patterns	TP2 – Durations TP3 – Time Lags between Events; TP1 can be implemented based on pattern TP3
Known uses	MS Project, BPMN, Eder et al. [2], Bettini et al. [4], Combi et al. [1]

Fig. 4. TP1 - Time Lags between Activities

of the outer loop and the inner one respectively. Hence this case needs to be excluded when defining semantics of pattern TP1.

To formally express this we define function $iteration(S, \varphi, \tau)$. It returns ordered set $(e_0 : n_0, e_{L_1} : n_{L_1}, \ldots, e_{L_k} : n_{L_k})$ which uniquely identifies each loop and its current iteration count with respect to a possibly surrounding loop (cf. Fig. 5). Thereby, e_0 is the start event of the respective process instance of schema S and e_{L_i}, $(1 \leq i \leq k)$ is the first event of a loop containing event e of event occurrence $\varphi = (e, \cdot)$[3] (e.g., e_2 and e_4 for start event e_{A_6S} of activity A_6 in Fig. 5). n_{L_i} $(1 \leq i \leq k)$, in turn, designates the iteration count of an inner loop e_{L_i} with respect to an outer loop $e_{L_{i-1}}$ (cf. Fig. 5), with n_0 always having value 1. A minimum time lag t_{min} (Design Choice D[a] in Fig. 3), for example, can now be formalized as follows:

$$\forall \varphi \in occurrences(S, e_A, \tau) \forall \psi \in occurrences(S, e_B, \tau) :$$
$$iteration(S, \varphi, \tau) = iteration(S, \psi, \tau) \Rightarrow \varphi^t + t_{min} \leq \psi^t.$$



Possible trace (without timestamps) and respective iterations:

$$\underbrace{\text{Trace (Events)}}_{\text{Iteration}} \Rightarrow \underbrace{e_0, e_{A_1S}, e_{A_1E}}_{(e_0:1)}, \underbrace{e_2, e_{A_3S}, e_{A_3E}}_{(e_0:1, e_2:1)}, \underbrace{e_4, e_5, e_{A_6S}, e_{A_6E}, e_8, e_9}_{(e_0:1, e_2:1, e_4:1)}, \ldots$$

$$\ldots, \underbrace{e_4, e_5, e_{A_7S}, e_{A_7E}, e_8, e_9}_{(e_0:1, e_2:1, e_4:2)}, \underbrace{e_{A_{10}S}, e_{A_{10}E}, e_{11}}_{(e_0:1, e_2:1)}, \underbrace{e_2, e_{A_3S}, e_{A_3E}}_{(e_0:1, e_2:2)}, \ldots$$

(e_{A_iS}: start event, and e_{A_iE}: end event of activity A_i.)

$$\ldots, \underbrace{e_4, e_5, e_{A_6S}, e_{A_6E}, e_8, e_9}_{(e_0:1, e_2:2, e_4:1)}, \underbrace{e_{A_{10}S}, e_{A_{10}E}, e_{11}}_{(e_0:1, e_2:2)}, \underbrace{e_{A_{12}S}, e_{A_{12}E}, e_{13}}_{(e_0:1)}$$

Fig. 5. Nested Loops and Iterations

Pattern TP2 (*Durations*) is described in Fig. 6. It allows specifying duration of process elements. If pattern TP2 is applied to an activity or process, same compliance rules as for pattern TP1 must hold. Thereby, e_A corresponds to the start event of the respective activity (process), while e_B corresponds to its end event. For a set of activities (process instances) (see Design Choices C[b] and C[d] in Fig. 2b), in turn, these rules need to be applied to the first start event and the last end event of all activity (process) instances of this set.

Pattern TP3 (*Time Lags between arbitrary Events*). TP3 is described in Fig. 7. It enables specification of time lags between two discrete events. Semantics of this pattern is similar to the one of TP1. However, no restrictions regarding events e_A and e_B apply (i.e., respective events do not need to be start/end events of activities). Thus, opposed to TP1, TP3 provides more generic support

[3] Here we only consider well-nested loops.

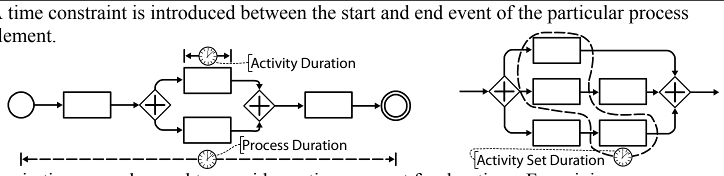
Time Pattern TP2: Durations	
Also known as	-
Problem	A particular process element has a certain duration restriction. Durations result from both waiting and processing times. Durations are often determined by external benchmarks (e.g., regulations, policies, QoS agreements). The duration may or may not have binding character.
Design Choices	C.) Durations can be applied to all four kinds of process elements (cf. Fig. 2b) D.) Durations may represent all three kinds of restrictions (cf. Fig. 3)
Solution	A time constraint is introduced between the start and end event of the particular process element.  Activity Duration Process Duration Activity Set Duration Again timers can be used to provide runtime support for durations. For minimum (maximum) durations the respective element must not complete before (after) the timer has expired, otherwise appropriate exception handling is initiated. For intervals, the completion event has to occur within the interval boundaries.
Context	The mechanism evaluating the constraint (i.e., starting the timer) needs to be able to access the value of the duration before the particular element is executed.
Examples	• The assembly of a new engine must not take longer than 30 minutes (task work) (Design Choices C[a], D[b]) • Depending on its severity, ovarian cancer surgeries take 1 to 10 hours (Design Choices C[a], D[c]). • Maintenance issues need to be resolved within 1hr (Design Choices C[c], D[b]) • Processing 100 requests must not take longer than 1 second (Design Choices C[d], D[b])
Related Patterns	TP1 – Time Lags between Activities TP3 – Time Lags between Events – TP2 can be implemented based on TP3
Known uses	MS Project, BPMN, MQ Workflow, Eder et al. [2], Bettini et al. [4], Combi et al. [1]

Fig. 6. TP2 - Durations

for expressing arbitrary time lags. For example, respective events can be triggers from an external source (e.g., receiving a message, occurrence of a heart stroke) not controllable by the PAIS. In addition, they may refer to events not bound to a specific activity (e.g., event "delivery of all parts" requires several activities/processes to complete) or to events triggered inside an activity (e.g., milestone of an activity or subprocess, occurrence of exceptions).

Pattern Category II (Restrictions of Process Execution Points). This category comprises four patterns for restricting execution points (e.g., earliest start or latest end time) of process elements. Regarding this category design choice F describes what kind of execution point is specified by the respective constraint (e.g., earliest start or latest end date) (cf. Fig. 8).

Pattern TP4 (*Fixed Date Element*). TP4 is described in Fig. 9. It provides support for specifying a deadline. In many cases, fixed date elements implicitly determine latest (earliest) start (end) time of preceding (succeeding) activities as well. In most cases, the value of such fixed date element may not only depend on process schema S, but also on the current process instance (i.e., trace τ) and current iteration I of respective process element A. Therefore, we define function

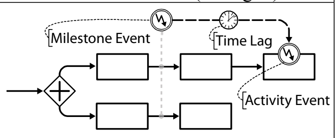
Time Pattern TP3: Time Lags between Arbitrary Events	
Also known as	-
Problem	There is a given time lag between two discrete events which needs to be respected. Events occur, for example, when instantiating or completing a process instance, when reaching a milestone in a process instance, or when triggering specific events inside an activity. Time lags are often required to comply with existing rules and regulations. The time lag may or may not have binding character.
Design Choices	D.) Time Lags between Events may represent all three kinds of restrictions (cf. Fig. 3)
Solution	A time constraint is introduced between the respective events. Again timers can be used to realize this pattern at runtime (cf. Fig. 4). Additionally an observer monitoring external events and notifying the mechanism evaluating the constraint is necessary. 
Context	The mechanism evaluating the constraint (i.e., starting the timer) needs to be able to access the value of the time lag in order to determine the impact of the constraint.
Examples	• Maximum time lags in an electronic change management process between sending a request for comments (by the partners affected by a change) and getting a response (Event) (Design Choices D[b]). • The time lag between delivery of all parts (milestone) and the assembly of the car's chassis (milestone) should be no more than 2 hours (e.g. just-in-time production) (Design Choices D[c]).
Related Patterns	TP1 – Time Lags between Activities TP2 – Durations
Known uses	Bettini et al. [4], Combi et al. [1]

Fig. 7. TP3 - Time Lags between Events

General Design Choice for Pattern Category II
F.) Patterns can restrict three dates of a process element a.) Earliest start date, b.) Latest start date, c.) Latest completion date

Fig. 8. General Design Choices for Category II

$fde(S, A, I, \tau)$, which returns for each process element A with fixed date element and each iteration I the current value of the fixed date element. Therefore fde effectively represents the Fixed Date attached to each Fixed Date Element (cf. Fig. 9). Compliance of a trace then means that each occurrence of the respective event e of element A in τ complies with the associated value of the fixed date constraint.

Pattern TP5 (*Schedule Restricted Element*). TP5 is described in Fig. 10. It enables us to restrict the execution of a particular element by a schedule; i.e., a timetable (e.g., a bus schedule). A particular schedule s_A attached to an activity (process) A can be instantiated as a (possibly infinite) set of subsets of the time points of a calendar C, i.e., $s_A \subseteq 2^C$. Depending on Design Choice G (cf. Fig. 10) the schedule is either instantiated as set of discrete time points $s_A = \{t | t \in C\}$ or as set of intervals $s_A = \{[t_{min}, t_{max}] | [t_{min}, t_{max}] \subseteq C\}$. An exception in respect to this schedule (cf. Fig. 10) is then expressed by removing and/or adding the respective time points or time intervals from/to the

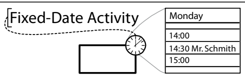
Time Pattern TP4: Fixed Date Elements	
Also known as	Deadline
Problem	A particular element has to be executed at a particular date. Fixed Date Elements often determine the latest or earliest start / completion time of preceding / succeeding activities as well. If the deadline is missed the activity or process may even become obsolete.
Design Choices	C.) A fixed date can be applied to an activity (a.) or process instance (c.) (cf. Fig. 2b) F.) A fixed date can restrict all three types of dates (cf. Fig. 8)
Solution	A Fixed Date is attached to the respective element. A Fixed Date can be realized using a timer which is started, as soon as the value of the fixed date is known and expires at the respective date. If, for example, for a latest start date the respective element has not been started before the timer has expired appropriate exception handling is initiated. This could, for example, lead to the cancelation of the respective activity. Other restriction can be handled analogously (cf. Fig. 4 for an example).  Fixed-Date Activity Monday 14:00 14:30 Mr. Schmith 15:00
Context	The value of the fixed date needs to be available prior to the respective activity becoming available for execution.
Examples	• Assume that software is released every two weeks on Friday evening. Thus, the deadline for changes (except bug fixes) is the day before the release date (time error might lead to delays or have no effect) (Design Choices C[a] F[c]). • To perform chemotherapy the physician has to inform the pharmacy about the dosage of the cytostatic drug until 11:00. If the deadline is missed the pharmacy checks back by phone for the exact dosage (escalation mechanism) (Design Choices C[a] F[c]). • A patient has an appointment for an examination Monday at 10:00, but due to a full schedule of the physician it may well be that the patient has to wait until the examination starts (i.e., earliest possible execution point is given) (Design Choices C[a] F[b]).
Related Patterns	TP5 – Schedule Restricted Elements; Fixed Date Elements are often schedule restricted elements as well.
Known uses	MS Project, BPMN, Eder et al. [2], Bettini et al. [4], Combi et al. [1]

Fig. 9. TP4 - Fixed Date Elements

schedule. To verify that a particular activity/process instance complies with the schedule it needs to be checked whether or not timestamp t of the respective event constitutes an element of the schedule (i.e. $t \in s_A$).

Pattern TP6 (*Time Based Restrictions*) enables us to restrict the number of times a particular process element can be executed within a predefined time frame (cf. Fig. 11). The particular time frame(s) are either defined by the time points of two events (Design Choice I[a]) or by a schedule (Design Choice I[b]). Based on these time frames it becomes possible to determine the number of executions within a particular time frame.

Pattern TP7 (*Validity Period*) enables us to restrict the lifetime of a process element to a given validity period (cf. Fig. 12). Semantics can be expressed by checking whether the timestamps of respective events lie within the particular validity period attached to the process element.

Pattern Category III (*Variability*) and **Pattern Category IV (*Recurrent Process Elements*).** Our catalogue comprises three additional patterns TP8, TP9 and TP10 covering time dependent variability, cyclic elements and periodicity. Due to lack of space we omit them here and refer to our technical report [14] instead.

5 Related Work

Patterns were first used by Alexander [15] to describe solutions to recurring problems and best practices in architectural design. Patterns also have a long tradition in computer science. Gamma et al. [16] applied same concepts to software engineering and described 23 design patterns. In the workflow area, patterns were introduced for analyzing expressiveness of process meta models [5,17]. In this context, control flow patterns describe constructs to specify activities and their ordering. In addition, workflow data patterns [6] provide ways for modeling the data aspect in PAIS. Furthermore, patterns for describing control-flow changes [18,7] and service interactions were introduced [19]. The introduction of workflow patterns has had significant impact on PAIS design and on the evaluation of PAIS and process languages. To evaluate powerfulness of a PAIS

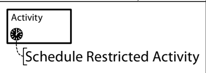
Time Pattern TP5: Schedule Restricted Elements	
Also known as	-
Problem	The execution of a particular element (i.e., activity or process) is restricted by a schedule. The structure of this schedule is known at process type level, while the concrete date is determined at instance level. The schedule provides restrictions on when the respective element can be executed. In particular, for rather restricted schedules even small delays in process execution can become critical (if schedule restricted elements being on a critical path are affected by the delay or the path becomes critical due to the delays). Schedules may contain exceptions (e.g., every year except leap years).
Design Choices	C.) A fixed date can be applied to an activity (a.) or process instance (c.) (cf. Fig. 2b) F.) A fixed date can restrict all three types of dates (cf. Fig. 8) G.) Execution of the element can be bound to a.) several discrete points in time (execution is only possible every full hour) or b.) one or more time frames (e.g. execution is only possible from 09:00 to 12:00)
Solution	A schedule is attached to the respective element. A schedule restriction can be realized using a timer which is started when the process is started and expires when the first time frame of the schedule is reached (a discrete point in time (Design Choice G[a]) can be seen as a time frame with only one time point). The timer is then reset and its expiration date is set to the end of the next time frame of the schedule. This is repeated until no more time frames are in the schedule or the process element has been started / completed (cf. Design Choice F). If the start / end of the respective element does not occur within a valid time frame or there is no longer a time frame available in the schedule, appropriate exception handling is initiated. 
Context	The schedule needs to be known at process type level or at least at process instantiation.
Examples	• Between Munich and Amsterdam there are flights at 6:05, 10:30, 12:25, 17:35 and 20:40 (Design Choice C[a] G[a]). • Opening hours of the dermatological clinic are MO – FR 8:00 – 17:00 except for public holidays. Dermatological examinations can only be scheduled within this time frame (Design Choices C[a] G[b]). • An information letter is sent by the leasing company to each customer within the first two weeks of each year (Design Choices C[a] G[b]) • Comprehensive lab tests in a hospital can only be done from MO – FR 8:00 – 17:00 (Design Choices C[a] G[b])
Related Patterns	TP4 – Fixed Date Elements (often schedule restricted elements) TP6 – Time Based Restrictions (like schedule based restrictions constrain possible execution points for an element) TP7 – Validity Period
Known uses	MS Project, Eder et al. [2], Combi et al. [1]

Fig. 10. TP5 - Schedule Restricted Element

regarding its ability to cope with time aspects, existing workflow patterns are important, but not sufficient. In addition, patterns addressing time constraints are needed.

Most academic approaches on time support for PAIS focus on time features like verification of time constraints [4,1,2], escalation management [9], and scheduling support [10,11]. The effect of ad-hoc changes on temporal constraints is investigated in [20]. A systematic investigation of requirements for time support from different heterogeneous application domains is missing so far.

6 Summary and Outlook

We have proposed time patterns to foster selection of appropriate PAIS-enabling technologies and to facilitate comparison of process management systems, calendar systems and project planning tools regarding their coverage of the time perspective in PAIS. In [14] we provide additional time patterns as well as a pattern-based evaluation of existing systems based on the time patterns. We

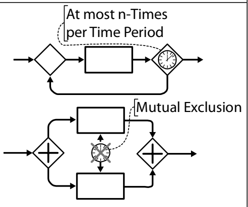
Time Pattern TP6: Time Based Restrictions	
Also known as	Some occurrences of this pattern are often referred to as "Mutual Exclusion"
Problem	Particular process elements may only be executed a limited number of times within a given timeframe. Time Based Restrictions are often needed to express the influence of resource restrictions (resource shortage) onto process execution.
Design Choices	H.) Time Based Restrictions can be applied to different types of process elements a.) Instances of single activity or group of activities within same process instance b.) Instances of single activity or group of activities within different process instances (potentially sharing some common characteristics) c.) Instances of a process or group of processes I.) There are two types of restrictions which can be expressed by Time Based Restrictions a.) Number of concurrent executions (at same time / with overlapping time frames) or b.) Number of executions per time period
Solution	To implement this pattern a constraint expressing a particular Time Based Restriction is associated with the process elements affected by this restriction. Additionally, the constraint specifies the respective time period and the number of executions. During runtime an observer can be used to monitor the number of running instances per time period and to raise an exception in case the maximum number of executions is exceeded. 
Context	The number of executions needs to be accessible by the observer before any of the respective process elements is started.
Examples	• Two invasive examinations must not be performed on same day (Design Choices I[b]). • For USD 19.90 10 different online books can be read per month. If the book tokens are consumed no more books can be read in the current month. At beginning of next month the book tokens get renewed (Design Choices H[a] I[b]). • During your stay at a wellness hotel you can select one treatment (free of charge) per day (Design Choices H[a] I[b]).
Related Patterns	TP5 – Schedule Restricted Elements; While the execution point of a schedule restricted element is constrained by a schedule, time based restrictions constrain the amount of activity instances / time period.
Known uses	-

Fig. 11. TP6 - Time Based Restrictions

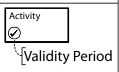
Time Pattern TP7: Validity Period	
Also known as	-
Problem	A particular process element may be only executed within a particular validity period, i.e., its lifetime is restricted to the validity period. The respective process element may only be instantiated within this validity period. In general, different versions of a process element may exist, but only one is valid at a specific point in time. Validity dates are especially relevant in the context of process evolution to restrict the remaining lifetime of an obsolete process implementation and to schedule rollout of the new process.
Design Choices	C.) A validity period can be applied to an activity (a.) or process instance (c.) (cf. Fig. 2b) F.) A validity period can restrict all three types of dates (cf. Fig. 8)
Solution	To realize this pattern a validity period is attached to the respective element. Upon instantiation of the respective process element, its validity period needs to be checked. If the element does not lie within its validity period or the duration of the element (see Fig. 5) leads to the end event being outside of the validity period, appropriate error handling is required. 
Context	The validity period needs to be known at process type level. If the validity period is bound to an activity it may apply to several different process types.
Examples	• Starting from Jan 1st patients need to be informed about any risks before the actual treatment takes place (Design Choice C[c] F[a]). • From next week on the new service version should get life (Design Choice C[a] F[a]). • Due to changed law, process A may only be used until January 1st. After this date no new process instances can be instantiated based on A, but process B has to be used instead (Design Choice C[c] F[b]).
Related Patterns	TP5 – Schedule Restricted Elements TP8 – Time Dependent Variability
Known uses	MQ Workflow

Fig. 12. TP7 - Validity Period

have shown that suggested time patterns are highly relevant in practice and complement existing workflow patterns with another fundamental dimension. In future work we will provide a reference implementation. Furthermore, we will conduct a comprehensive study of time support features (e.g., verification of time constraints, escalation management, scheduling support), in addition to the proposed time patterns, and also consider the resource dimension in this context.

References

1. Combi, C., Gozzi, M., Juarez, J., Oliboni, B., Pozzi, G.: Conceptual modeling of temporal clinical workflows. In: Proc. TIME 2007, pp. 70–81 (2007)
2. Eder, J., Panagos, E., Rabinovich, M.: Time constraints in workflow systems. In: Jarke, M., Oberweis, A. (eds.) CAiSE 1999. LNCS, vol. 1626, pp. 286–300. Springer, Heidelberg (1999)
3. Dadam, P., Reichert, M., Kuhn, K.: Clinical Workflows - The Killer Application for Process-oriented Information Systems?. In: Proc. BIS 2000, pp. 36–59 (2000)
4. Bettini, C., Wang, X.S., Jajodia, S.: Temporal reasoning in workflow systems. In: Distributed and Parallel Databases, pp. 269–306 (2002)
5. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. Distributed and Parallel Databases 14, 5–51 (2003)
6. Russell, N., ter Hofstede, A., Edmond, D., van der Aalst, W.: Workflow Data Patterns. Technical Report FIT-TR-2004-01, Queensland Univ. of Techn. (2004)

7. Weber, B., Reichert, M., Rinderle-Ma, S.: Change Patterns and Change Support Features -Enhancing Flexibility in Process-Aware Information Systems. Data and Knoweldge Engineering 66, 438–466 (2008)
8. Russell, N., van der Aalst, W., ter Hofstede, A.: Exception Handling Patterns in Process-Aware Information Systems. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 288–302. Springer, Heidelberg (2006)
9. van der Aalst, W.M.P., Rosemann, M., Dumas, M.: Deadline-based escalation in process-aware information systems. Decision Support Systems 43, 492–511 (2007)
10. Combi, C., Pozzi, G.: Task scheduling for a temporal workflow management system. In: Proc. TIME 2006, pp. 61–68 (2006)
11. Eder, J., Pichler, H., Gruber, W., Ninaus, M.: Personal schedules for workflow systems. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 216–231. Springer, Heidelberg (2003)
12. Schultheiß, B., Meyer, J., Mangold, R., Zemmler, T., Reichert, M.: Designing the processes for chemotherapy treatment in a Women's Hospital (in German) (1996)
13. German Association of the Automotive Industry: Engineering Change Management. Part 1: Engineering Change Request, V 1.1., Doc. No. 4965 (2005)
14. Lanz, A., Weber, B., Reichert, M.: Time patterns in process-aware information systems - a pattern-based analysis - revised version. Technical Report UIB-2009, University of Ulm, Germany (2009)
15. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language. Oxford University Press, New York (1977)
16. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1995)
17. Puhlmann, F., Weske, M.: Using the Pi-Calculus for Formalizing Workflow Patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 153–168. Springer, Heidelberg (2005)
18. Rinderle-Ma, S., Reichert, M., Weber, B.: On the formal semantics of change patterns in process-aware information systems. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 279–293. Springer, Heidelberg (2008)
19. Barros, A., Dumas, M., ter Hofstede, A.: Service Interaction Patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 302–318. Springer, Heidelberg (2005)
20. Sadiq, W., Marjanovic, O., Orlowska, M.E.: Managing change and time in dynamic workflow processes. Int. J. Cooperative Inf. Syst. 9, 93–116 (2000)