Time Patterns for Process-aware Information Systems: A Pattern-based Analysis

Revised version

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 very limited in existing process management systems. By contrast, different kinds of planning tools (e.g., calendar systems and project management tools) provide more sophisticated facilities for handling task-related time constraints, but lack an operational support for business processes. This paper presents a set of 10 time patterns to foster the systematic design and comparison of these different technologies in respect to the time perspective. These time patterns are all based on empirical evidence from several large case studies. In order to ease use and implementation for each time pattern we provide a precise formal semantics. In addition, we provide an in-depth evaluation of selected process management systems, calendar systems and project management tools based on the suggested patterns. The presented work will not only facilitate comparison of these different technologies in respect to their support of time constraints, but also make evident that their integration offers promising perspectives in respect to time support for long-running business processes. 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 time constraints 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 and flight planning) [1, 2, 3, 4], time support is rather limited in existing process management systems [1, 5]. 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) in the following).

To make PAIS better comparable and to facilitate the selection of appropriate PAIS-enabling technologies, workflow patterns have been introduced [6, 7, 8, 9]. 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 [6], data flow [7], resources [8], activities [10], exceptions [11], and process change [9]. However, a framework for systematically evaluating PAIS in respect to their ability to deal with the time perspective is missing and is picked up by this paper. Our contributions are as follows:

1. We suggest 10 *time patterns* to foster the comparison of existing PAIS with respect to their ability to deal with time aspects. The proposed time patterns complement existing workflow patterns and were systematically identified by analyzing a large collection of process models in healthcare, automotive engineering, aviation industry, and other domains.
2. In order to avoid disambiguities and to ease both use and implementation of the time patterns, for each time pattern we define a precise *formal semantics*. The description of this semantics is independent of a particular process meta model and is based on the (temporal) execution traces producible on a time constrained process schema.
3. We provide an *in-depth evaluation* of selected approaches from both industry and academia based on the proposed time patterns. The evaluation does not only consider process management systems, but also calendar systems and project planning tools in which time aspects play an important role.

Our pattern-based analysis shows that these different technologies all provide support for time aspects. The presented work will not only facilitate their comparison in respect to the support of time constraints, but also foster the selection of appropriate time components when designing PAIS. Moreover, our work makes evident that their integration offers promising perspectives in respect to more sophisticated time support for long-running business processes, i.e., knowing the commonalities and differences will be a first step to integrate these technologies (e.g., process management and calendar systems).

Section 2 summarizes basic notions. Section 3 presents the research method employed for identifying the time patterns. Section 4 describes 10 time patterns sub-dividing them into 4 categories. In Section 5 we provide a formal semantics for each of the 10 patterns. Section 6 summarizes the results from our evaluation of selected approaches and tools. We present related work in Section 7 and conclude with a summary and outlook in Section 8.

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.

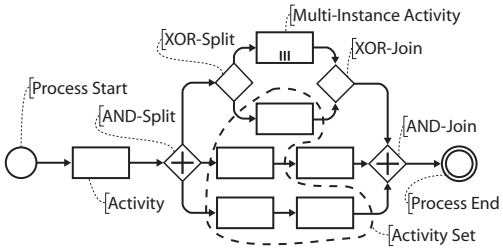


Fig. 1. Core Concepts of a Process Model

3 Research Method

The overall goal of this paper is to complement existing workflow patterns (e.g., [6, 7, 8, 9, 10, 11]) 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 [4] and product engineering [12]).

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 a high coverage of real-world scenarios, and not on specific time features of a PAIS like verification of time constraints [2, 5, 1, 3], escalation management [13], or scheduling support [14, 15].

Sources of Data and Data Collection. As sources for our patterns we consider results of comprehensive case studies we performed in different domains, including healthcare, automotive engineering, aviation industry, and others.

One of our major data sources is a large *healthcare* project in which we designed core processes of a Women's Hospital [16]. 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 major data source we use process models from the *automotive industry*. We consider a case study on electronic change management (ECM) [17] and process models described in [18]. Some of the models related to ECM have been also published by the German Association of the Automotive Industry (VDA) [17]. The process models described in [18], in turn, refer to car repair and maintenance in garages, in-house change management, and product development. With several hundred activities the product development is the most complex process we consider. In total this case 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 and documented the flight planning and post flight phase. As the 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 comprises *healthcare processes* from a large Medical University Hospital. We consider 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-enabling technologies. 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, which is presented in Section 4.

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).

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 10 time patterns, which are described in detail in the following. For each pattern we provide a name, synonyms, a brief description of the addressed problem, design choices, remarks regarding its implementation including a visualization, illustrating examples from our case studies, a reference to related patterns, and known uses of the pattern summarized in a table (cf. Fig. 4 - Fig. 15).

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 approaches 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 general 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 10 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. These design choices are shortly described in the following. The time parameters of a time pattern can be set at built-time, at instantiation time, or at run-time (Design Choice A). This has specifically great impact on the question whether, when and how the time constraints of a particular process can be validated. Furthermore the time parameters of each time pattern can have different granularities (depending on what granularities are supported by the time reference system). Typical granularities are years, months, weeks, days, hours, minutes and seconds, but also system-defined granularities (e.g., business days) or user-defined ones (e.g., Wednesday afternoon) (Design Choice B). At this point we neither consider how time granularities can be expressed nor how they can be used when verifying the temporal constraints of a process. For both of these questions several solutions have been proposed in literature [5, 1]. Finally, Design Choice C describes to which process elements the pattern can be applied.

For each time pattern we provide a description using the aforementioned schema (cf. Fig. 4 - Fig. 15).

4.1 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

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. In addition to General Design Choice D, TP1 defines one other design choice. Design Choice E describes whether the time lags describe a start-start relation (e.g., between the start event of two different activities), a start-end relation, an end-start relation, or an end-end relation.

Pattern TP2 (Durations) is described in Fig. 5. It enables specification of duration of process elements. For example, Design Choices C[a] and D[b] refer to a variant of TP2 where a maximum duration for a single activity is described (e.g., the assembly of a new engine must not take longer than 30 minutes).

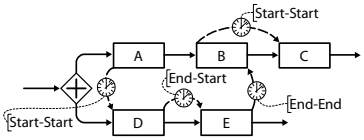
Time Pattern TP1: Time Lags between two Activities	
Also known as	Upper and Lower Bound Constraints, Inter-Task Constraints, Temporal Relations
Problem	There is a given time lag between two activities which needs to be respected. Time Lags may not only exist between succeeding activities, but also between 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 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

Pattern TP3 (Time Lags between arbitrary Events). TP3 is described in Fig. 6. It enables specification of time lags between two discrete events. Thus, opposed to TP1, TP3 provides more generic support 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 which are not bound to a specific activity (e.g., event "delivery of all parts" requires several activities/processes to complete) or to events which are triggered inside an activity (e.g., milestone of an activity or subprocess, occurrence of exceptions, subprocess reaches a special state, special condition occurred).

4.2 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.

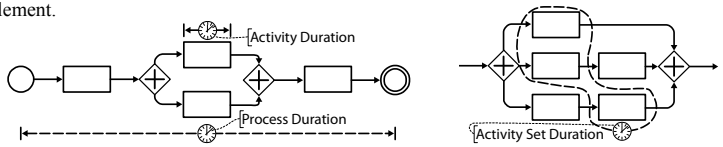
Time Pattern TP2: Durations	
Also known as	-
Problem	A particular process element has a certain duration restriction. Durations result from both waiting and processing times. Durations are often determined by external benchmarks (e.g., regulations, policies, QoS agreements). The duration may or may not 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.  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. 5. TP2 - Durations

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. 7).

Pattern TP4 (Fixed Date Element). TP4 is described in Fig. 8. It provides support for specifying a deadline. For a particular process element it can be fixed whether it has to be started after, started before or completed before a particular date (Design Choice F). In many cases, fixed date elements implicitly determine latest (earliest) start (end) time of preceding (succeeding) activities as well.

Pattern TP5 (Schedule Restricted Element). TP5 is described in Fig. 9. It enables us to restrict the execution of a particular element by a schedule; i.e., a timetable (e.g., a bus schedule). The schedule itself is known at built-time, whereas the concrete dates are specified either at instantiation or run-time. The schedule may comprise several discrete points in time, but also one or more time frames (Design Choice G).

Pattern TP6 (Time Based Restrictions) enables us to restrict the number of times a particular process element can be executed within a predefined time

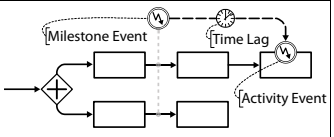
Time Pattern TP3: Time Lags between Arbitrary Events	
Also known as	-
Problem	There is a given time lag between two discrete events which needs to be respected. Events occur, for example, when instantiating or completing a process instance, when reaching a milestone in a process instance, or when triggering 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. 6. 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. 7. General Design Choices for Category II

frame (cf. Fig. 10). Design Choice H describes to which process element the pattern may be applied (e.g., activity instances of a single process instance or of multiple process instances).

Pattern TP7 (Validity Period) enables us to restrict the lifetime of a process element to a given validity period (cf. Fig. 11). Design Choice F allows specifying the earliest / latest start (completion) date for the respective process element.

4.3 Pattern Category III (*Variability*).

This category comprises **Pattern TP8 (*Time Dependent Variability***, see Fig. 12**)** which allows varying control flow depending on the execution time, or time lags between activities/events (Design Choice J). As example consider the left diagram in Fig. 12. Depending on the time the XOR-Split is executed either the upper or the lower path is chosen.

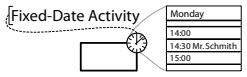
Time Pattern TP4: Fixed Date Elements	
Also known as	Deadline
Problem	A particular element has to be executed at a particular date. Fixed Date Elements often determine the latest or earliest start / completion time of preceding / succeeding activities as well. If the deadline is missed the activity or 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). 
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. 8. TP4 - Fixed Date Elements

4.4 Pattern Category IV (*Recurrent Process Elements*).

This category comprises patterns to express cyclic elements and periodicity. Design Choice K is a general design choice for Category IV which describes whether the number of cycles is determined explicitly, is calculated based on time lags and end dates, or is depending on an exit condition (cf. Fig. 13).

Pattern TP9 (Cyclic Elements) allows specifying cyclic elements which are performed iteratively considering time lags between cycles (cf. Fig. 14). Design Choice L specifies whether time lags between cycles are fixed (i.e., have always same length) or may vary from iteration to iteration. Design Choice N describes whether the time lags describe a start-start relation (e.g., between the start event of two different activities), a start-end relation, an end-start relation, or an end-end relation. This pattern represents a variant / extension of pattern TP1 in which the second activity lies in the iteration succeeding the one to which the first activity belongs.

Pattern TP10 (Periodicity). TP10 allows specifying periodically recurring process elements according to an explicit *periodicity* rule (cf. Fig. 15). In contrast to TP9, emphasis of TP10 is on possible execution dates of the recurrent element and not on the time lags between the iterations. Design Choice O describes whether the periodicity rule may contain one or more dates.

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. 9. TP5 - Schedule Restricted Element

5 Formalization of Time Patterns

In this section we provide a formalization of the time patterns proposed in the last section. First we give some general definitions of relevant notions like *Calender* and *Time Distance*.

5.1 Basic Notions

Definition 1 (Calendar). *A Calendar $\mathcal{C}$ is an infinite set of absolute time points without gaps (e.g. the Gregorian Calendar).*
Properties:

– *$\mathcal{C}$ is a total order; i.e., it has an ordering relation $\leq_C$ and*

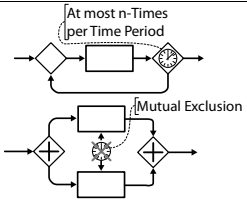
Time Pattern TP6: Time Based Restrictions	
Also known as	Some occurrences of this pattern are often referred to as "Mutual Exclusion"
Problem	Particular process elements may only be executed a limited number of times within a given timeframe. Time Based Restrictions are often needed to express the influence 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. 10. TP6 - Time Based Restrictions

$$\forall x, y \in \mathcal{C} : x \leq_C y \vee y \leq_C x$$

Definition 2 (Time Distance). *A Time Distance describes a relative distance between two points in time using a particular time granularity. Let $\mathcal{D}$ denote the set of all Time Distances.*
Properties:

- *$\mathcal{D}$ is a partial order; i.e., it has a ordering relation $\leq_D$*
- *The addition of a time point of a Calendar C and a Time Distance is given as*

$$+ : \mathcal{C} \times \mathcal{D} \mapsto 2^{\mathcal{C}}$$

Regarding an absolute point in time $t \in \mathcal{C}$ and a time distance like "5 $workingdays$" $\in \mathcal{D}$, the precise semantics of the expression $t + 5\,workingdays$ is unclear although t represents an absolute point in time (e.g., "Mon 19.10.2009 17:32.324") and this expression can be intuitively calculated. Consequently, the exact value or meaning of $t_1 + 5\,workingdays \leq t_2$ is unclear since it is generally not known what "5 working days" or even "5 days" exactly means in the given

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. 11. TP7 - Validity Period

context. It could mean, for example, that this inequation will be not fulfilled anymore at 17:30.001 on the designated day but it could also mean, that 23:59.999 the same day or even 17:29.999 the next day are still ok. This wiggle room is a general problem of time units or more exactly the meaning of time units in human understanding. For example, the statement "at most one day later" (i.e., a maximum distance of 1 day) cannot be generally mapped to an exact distance.

Since we do not want to make any restriction regarding the meaning of expressions like $t_1 + d \leq t_2$, we assume that in the context at hand it is always possible to decide whether or not such expression is true. Therefore we also do consider Design Choice B when defining execution semantics of the time patterns.

Furthermore, Design Choice A has no impact on the execution semantics proposed in this paper. The reason is that we just consider the impact the time patterns have on temporal execution traces (i.e., the execution history of a process instance). Thus the "incarnation" of the temporal constraints has already taken places and the corresponding point in time has no impact on the validity of the constraints.

We now define the notion of events as used throughout the remainder of this paper. An event denotes the occurrence of some sort of trigger during the execution of a process instance. The start and end of each activity corresponding to a process instance for example, constitutes such an event (cf. Fig. 16). However, there are also external events like milestones, receipt of a message, occurrence

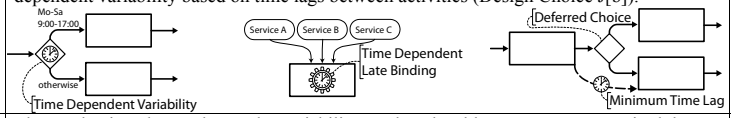
Time Pattern TP8: Time Dependent Variability	
Also known as	-
Problem	Depending on time aspects the control flow may vary; e.g., different branches of a process are executed or different sub process fragments are chosen.
Design Choices	J.) There are different time aspects which may be considered by an instance of this pattern a.) Execution time of an activity / process instance b.) Time lags between activities / events
Solution	Time dependent variability can be achieved in different ways. The simplest approach is to explicitly capture the required variability in the process model through enumerating all different options. Alternatively, techniques like late binding can be used to select appropriate activity implementations during run-time depending on time. Both of these implement the variability based on the time of execution (Design Choice J[a]). Finally, the Deferred Choice Pattern may be used in combination with triggers to achieve time dependent variability based on time lags between activities (Design Choice J[b]). 
Context	The mechanism that evaluates the variability needs to be able to access any required data when determining which of the possible alternatives should be chosen.
Examples	• Samples which are collected between 18 and 20 o'clock and which are sent to the Department of Clinical Chemistry, need to be marked as express requests in the request form. Outside the opening hours of the clinic only emergency cases are treated (Design Choice J[a]). • When issuing a passport the processing usually takes 4-6 weeks. If the person needs the passport earlier than 4 weeks an interim passport can be issued (Design Choice J[a]). • Patients admitted in the hospital between 6pm and 8am are always assigned to the emergency unit (for the first night); afterwards they are transferred to a normal ward (Design Choice J[a]). Between 8am and 6pm, in turn, patients are directly admitted by the ward.
Related Patterns	TP7 – Validity Dates
Known uses	BPMN

Fig. 12. TP8 - Time Dependent Variability

General Design Choice for Pattern Category IV
K.) The Number of cycles is a.) determined by a fixed / dynamic number of iterations, b. depends on time lag and end date or c. depends on exit condition

Fig. 13. General Design Choices for Category IV

of a heart stroke, and so forth (cf. Fig. 16). Thus, in our context we use the the notion of *event* as general term for something that happens during the execution of a process instance.

Definition 3 (Event, Event occurrence). *Let $\mathcal{PS}$ be the set of all process schemes. The set of all events which may occur during the execution of process schema $S \in \mathcal{PS}$ is denoted as $\mathcal{E}_S$ (without loss of generality we assume unique labeling of events in the given context).*

Let $\mathcal{C}$ be the total set of absolute timepoints of a calendar. Then

$$\varphi = (e, t) \in \mathcal{E}_S \times \mathcal{C}$$

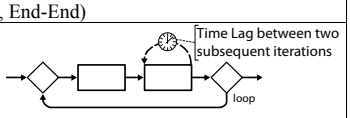
Time Pattern TP9: Cyclic Elements	
Also known as	-
Problem	A particular process element shall be performed iteratively considering time lags between the cycles.
Design Choices	C.) A cyclic element can be applied to all four kinds of process elements (cf. Fig. 2b) L.) Time Lag between cycles a.) is fixed (e.g., 3 hours) or b.) may vary M.)There are three kinds of restrictions (see also Design Choice D) a.) Minimum value, b.) Maximum value and c.) Time interval [min…max] N.) Time Lags can be realized based on four different time relations (cf. Design Choice E) 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 special time constraint is introduced between the start and / or end event of the two process elements where the respective event of the second process element lies in the succeeding iteration of the event of the first process element.  This pattern can be realized at runtime similar to TP1 with additional attention being paid to the iterations of the respective activities.
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. Additionally Time Lags may vary between iterations (cf. Design Choice L).
Examples	• Administer 50 to 75 mg in equally divided doses every 12 hrs for 5 subsequent days (Design Choices K[a] L[a] M[c], N[c]). • Maintenance activities for a particular aircraft have to be performed after every N flight hours (Design Choices K[c] L[a] M[b], N[c]).
Related Patterns	TP10 – Periodicity
Known uses	MS Project, BPMN [?], Combi et al. [?]

Fig. 14. TP9 - Cyclic Elements

denotes the occurrence of event $e \in \mathcal{E}_S$ *at time point* $t \in \mathcal{C}$*,i.e.,* t *defines the exact point in time at which event* e *occurred.*

Furthermore $\Phi_S = \mathcal{E}_S \times \mathcal{C}$ *denotes the set of all possible event occurrences of events* $e \in \mathcal{E}_S$*.*

Let $\mathcal{A}_S$ be the total set of activities (or more precisely activity labels) based on which process schemes $S \in \mathcal{PS}$ are specified (without loss of generality we assume unique labeling of activities in the given context). Let further e_{A_S} denote the start-event and e_{A_E} the end-event of activity $A \in \mathcal{A}_S$ (cf. Fig. 16). Then, for set $\mathcal{E}_S$ the following equation holds:

$$\{e | e \equiv e_{A_S} \vee e \equiv e_{A_E} \text{ for } A \in \mathcal{A}_S\} \subseteq \mathcal{E}_S$$

Based on events and event occurrences we now can define the notion of temporal execution trace (or *trace* for short). We assume that all events related to the execution of a process instance are recorded in a temporal execution trace together with a timestamp designating their time of occurrence. Formally, temporal execution traces are defined as follows:

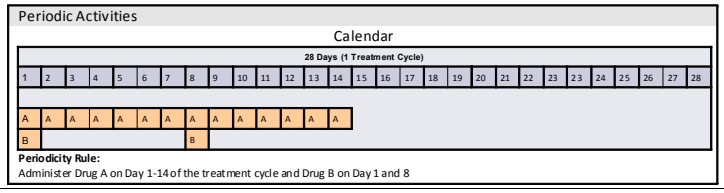
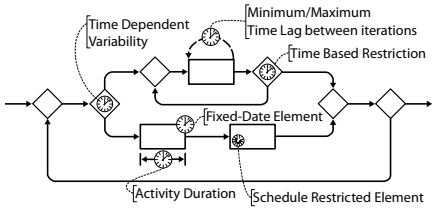
Time Pattern TP10: Periodicity	
Also known as	Recurrence, Appointment Series
Problem	A particular process element shall be performed periodically; i.e., according to a particular periodicity rule. Periodically implies some regularity, but does not necessarily mean equally distanced. A periodicity rule describes the reoccurrence pattern of the respective element (e.g., every Monday at 11:30) as well as start and end points (e.g., starting from next Monday until end of the year, 5 times). The reoccurrence patterns usually use a reference system over the given domain (e.g., a calendar). Periodicity rules may contain exceptions (e.g. every year except leap years) 
Design Choices	C.) A Periodicity can be applied to all four kinds of process elements (cf. Fig. 2b) O.) The periodicity rule contains a.) only one date (e.g., Monday at 10:00) or b.) more than one date (e.g., Monday morning and evening)
Solution	Periodicity rules can be realized by using combinations of patterns TP1-6, TP8 and TP9. Even for simple periodicity rules this can lead to complex processes where the periodicity rule cannot easily be recognize anymore. Therefore the periodicity as an additional layer of abstraction is necessary to describe such processes in an understandable way. 
Context	The contexts of the participating time patterns need to be fulfilled.
Examples	• Starting with next Monday group meetings will take place every two weeks at 11:30 (Design Choices K[c], O[a]). • Each day at 7:00 the responsible assistant physician of the Gynaecological Clinic is informing the assistant medical director about the patients (Design Choices M[c], O[a]). • Course ``Business Processes and Workflows'' takes place every Monday from 8:00 to 11:00 starting on Oct 6th and ending on Jan 26th. On Dec 8th, 22nd, 29th and on Jan 5th there will be no lectures taking place (Design Choices K[b], O[a]). • Stationary chemotherapy usually comprises 6 treatments which are performed every 14 days. At the end of one treatment cycle the date for the next chemotherapy is scheduled (Design Choices K[a], O[a]). • Administer Drug A on day 1 to 14 of each of the 6 treatment cycles and Drug B on the 1st and the 8th day. At the end of each treatment cycle the starting date for the next cycle is scheduled (Design Choices K[a], O[a]).
Related Patterns	TP9 – Cyclic Elements
Known uses	MS Project, BPMN [?], Combi et al. [?]

Fig. 15. TP10 - Periodicity

Definition 4 (Temporal Execution Trace). *Let $\mathcal{PS}$ be the set of all process schemes and let $\mathcal{E}_S$ be the set of all events which may occur in process schema $S \in \mathcal{PS}$ during its execution. Let further $\mathcal{C}$ be the total set of absolute timepoints of a calendar.*

Let further $\mathcal{T}_S$ denote the set of all possible temporal execution traces producible on process schema $S \in \mathcal{PS}$. Let Φ_S be the set of all possible event occurrences and $\varphi = (e, t) \in \Phi_S$ denote the occurrence of event e at time point t.

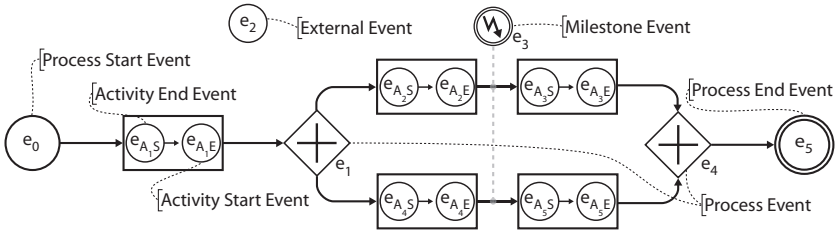


Fig. 16. Events in a Process

A particular temporal execution trace $\tau_S \in \mathcal{T}_S$ *is then defined as ordered set of event occurrences* φ_i*:*

$$\tau_S =< \varphi_1, \ldots, \varphi_n >$$

(with $\varphi_i \in \Phi_S, i = 1, \ldots, n,\ n \in \mathbb{N}$*) where the order of* $\varphi_i = (e_i, t_i)$ *in* τ_S *reflects the temporal order in which the events* e_i *occurred during the execution of a process instance running on process schema* S*, i.e.*

$$\forall \varphi_k, \varphi_j \in \tau_S : k < j \Rightarrow t_k < t_j.$$

Here, we assume that events in τ_S *do not occur at exactly same point in time.*

Additionally, in Table 1 we give some useful notions based on Definition 4 which facilitate formalization of the time patterns.

Let $\tau_S =< \varphi_1, \ldots, \varphi_n >\in \mathcal{T}_S$ be a temporal execution trace on process schema S. Then:
$\lvert\tau\rvert$ cardinality of τ
$\tau(i) = \varphi_i$ i-th item in temporal trace τ
$\varphi \in \tau \Longleftrightarrow \exists i \leq \lvert\tau\rvert$ with $\tau(i) = \varphi$
$e \in \tau \Longleftrightarrow \exists t \in \mathcal{C} : (e, t) = \varphi \in \tau$
$\varphi^e = e$ and $\varphi^t = t$ with $\varphi = (e, t)$

Table 1. Useful notions based on Definition 4

Furthermore, we define following functions, which returns all event occurrences for a particular event or process element:

Definition 5 (Trace Occurrences). *occurrences*(S, e, τ_S) *is a function which returns all occurrences* $\varphi = (e, \cdot)$ *of event* e *within temporal trace* τ_S *on process schema* S*. Formally:*

$$occurrences : \mathcal{S} \times \mathcal{E}_S \times \mathcal{T}_S \mapsto 2^{\Phi_S}$$

with

$$occurrences(S, e, \tau_S) = \{\varphi | \exists t \in \mathcal{C} : \varphi = (e, t) \in \tau_S\}$$

occurrences$(S, A, \tau_S$, *in turn, returns all occurrences of activity* A *within temporal trace* τ_S *of process schema* S*. An occurrence of an activity is identified by the occurrence of its start and end event, i.e.* (φ_S, φ_E)*. Formally:*

$$occurrences : \mathcal{S} \times \mathcal{A}_S \times \mathcal{T}_S \mapsto 2^{\Phi_S \times \Phi_S}$$

with

$$occurrences(S, A, \tau_S) = \{(\varphi_S, \varphi_E) | \varphi_S = (e_{A_S}, \cdot) \text{ and } \varphi_E = (e_{A_E}, \cdot)\}$$

Note that Definition 5 implies that $occurrences(S, e, \tau_S) = \emptyset$ holds, if $e \notin \tau_S$.

There are cases in which the iteration of a loop surrounding an activity needs to be explicitly taken into account, when defining patterns semantics. For example Time Pattern TP9 (cf. Fig. 14) requires the second activity to lie in the iteration succeeding the one to which the first activity belongs. For the sake of simplicity we presume nested loops here. To formally express this we define the iteration of a loop as follows:

Definition 6 (Iteration). *The iteration of a loop is defined as ordered set* $I = (e_0{:}n_0, e_{L_1}{:}n_{L_1}, \ldots, e_{L_k}{:}n_{L_k}) \in 2^{\mathcal{E} \times \mathbb{N}}$ *which uniquely identifies each loop and its current iteration counter with respect to a possibly surrounding loop. 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;* 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* $e_{L_{i-1}}$*. Thereby the iteration counter* n_0 *for the process instance, i.e. the start event* e_0 *of the process instance, always has the value* 1*.*

The Set of all possible iteration values for process schema $S \in \mathcal{PS}$ *is given as* $\mathcal{I}_S \subset 2^{\mathcal{E} \times \mathbb{N}}$*.*

Then: $iteration(S, \varphi, \tau_S)$ *is a function which returns the current iteration of the innermost loop surrounding event* φ^e*, or* $(e_0{:}1)$ *if there is no surrounding loop. Formally:*

$$iteration : \mathcal{S} \times \Phi_S \times \mathcal{T}_S \mapsto \mathcal{I}_S$$

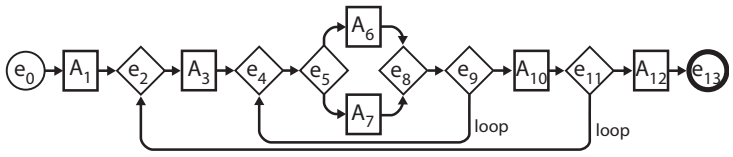
with

$$iteration(S, \varphi, \tau_S) = \begin{cases} (e_0{:}1) & \text{if } \varphi^e \text{ is not surrounded by a loop} \\ (e_0{:}1, e_{L_1}{:}n_{L_1}, \ldots, e_{L_k}{:}n_{L_k}) & \text{else} \end{cases}$$

with $e_{L_i}, \ldots, e_{L_k}$ *being the first events of all loops surrounding* φ^e*.*

This definition implies that events not belonging to the control flow are still part of the process instance and thus always have an iteration value of $iteration(S, \varphi, \tau_S) = (e_0{:}1)$. As example consider Figure 17. It shows a process graph with two nested-loops. Below the process graph, a possible execution trace is given together with the respective value of $iteration(S, \varphi, \tau_S)$ for each of the events in the trace.

Again Table 2 gives some useful notions based on Definition 6 which facilitate formalization of our time patterns.



Possible trace (without timestamps) and respective iterations:

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

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

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

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

Fig. 17. Nested Loops and Iterations

Let $\tau_S =< \varphi_1, \ldots, \varphi_n > \in \mathcal{T}_S$ be a temporal execution trace on process schema S. Then:

Iteration I occurs in trace τ_S, denoted as $\tau_S \vdash I$, iff

$$\exists \varphi \in \tau_S : iteration(S, \varphi, \tau_S) = I$$

For an iteration $I = (e_0{:}n_0, e_{L_1}{:}n_{L_1}, \ldots, e_{L_k}{:}n_{L_k})$ the succeeding iteration $I\oplus$ is defined as

$$I\oplus = (e_0{:}n_0, e_{L_1}{:}n_{L_1}, \ldots, e_{L_k} : (n_{L_k} + 1))$$

Table 2. Useful notions based on Definition 6

5.2 Formalization

The semantics of each pattern is now defined 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. Formally:

Definition 7 (Compliance). *A temporal execution trace $\tau_S \in \mathcal{T}_S$ is compliant with the set of temporal constraints defined on process schema $S \in \mathcal{PS}$ if and only if τ_S is compliant with each of the temporal constraints corresponding to process schema S. Compliance of a trace with a single temporal constraints is defined by the pattern semantic of each constraint.*

Based on the meta model independent notions of Def. 3-7 we now describe formal semantics of the different time patterns introduced in the previous section. The given formal specifications do not contain any constraints specific to a particular meta model. This has to be achieved separately by associating time

patterns with meta model-specific pre-/post-conditions. Our specifications contain generally valid pre-conditions where necessary; e.g., a minimum duration needs to be greater or equal to 0.

Pattern Category I The time patterns in this category all restrict the temporal distance between different sorts of events in the execution of a process instance. Thus their semantics can be formalized based on the time lag between those events.

The definition of the pattern semantics also needs to be valid in connection with loops. As illustrated in Fig. 18, considering time lags between two activities (TP1), for which one of the activities resides inside a loop and the other one outside that loop (e.g. Activities A1 and A3 or Activities A3 and A6 in Fig. 18) is problematic, since the exact semantics for such a case is unclear. This becomes even more complicated when considering nested loops.

Consider for example a time lag between activities A_1 and A_6 (cf. Fig. 4) in Fig. 18. There are several possible semantics for this case imaginable. Firstly, time lags coming from one loop or the process itself and entering a nested loop (cf. Fig. 18) could be restricted to only apply to the first or last iteration of the nested loop. Secondly, such an "incoming" time lag could also be applied to all iterations of the inner loop, and thirdly the time lag could be augmented by meta information which describes the iteration or iterations the time lag apply to. The same is possible for time lags starting inside a nested loop and ending outside this loop (cf. Fig. 18). As sketched in Fig. 19 there are even way more complex cases imaginable. Hence such cases need to be excluded when defining the semantics of pattern TP1.

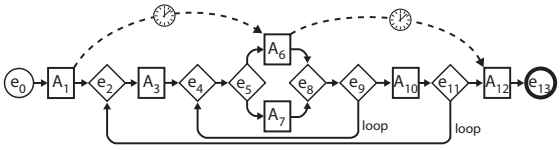


Fig. 18. Time Lags and Loops

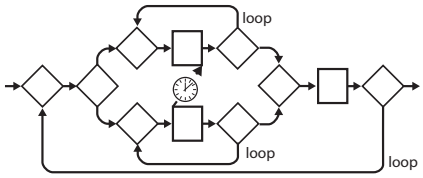


Fig. 19. Loops in parallel Branches

Pattern Sematics 1 (TP1: Time Lags between Activities). A time lag between two activities A and B can be mapped onto a time lag between the respective start and/or end events of these activities based on Design Choice E (cf. Fig. 4). Thus we denote the respective event of activity A as e_A and the respective event of activity B as e_B. Based on design choice E we then have the following relations between activity A and event e_A and activity B and event e_B respectively:

Design Choice E

E[a] e_A is the start event of activity A and e_B is the start event of activity B (start-start)

E[b] e_A is the start event of activity A and e_B is the end event of activity B (start-end)

E[c] e_A is the end event of activity A and e_B is the start event of activity B (end-start)

E[d] e_A is the end event of activity A and e_B is the end event of activity B (end-end)

This way, for example, start/start and end/start time lags can be described the same way. A time lag between events e_A and e_B can now be applied to a temporal trace if:

$$e_A \in \tau_S \text{ and } e_B \in \tau_S$$

Compliance of a given trace with such a time lag now means that all occurrences $\varphi = (e_A, \cdot)$ and $\psi = (e_B, \cdot)$ of the two events fulfill the given time lag. Based on design choice D a time lag between activities can now have three slightly different semantics which lead to slightly different definitions of compliance:

Design Choice D

D[a] A temporal trace τ_S complies with a minimum time lag $t_{min} \geq 0$[1] between two activities A and B if for the respective events e_A and e_B the following condition holds true

$$\begin{aligned} &\forall \varphi \in occurrences(S, e_A, \tau_S) \forall \psi \in occurrences(S, e_B, \tau_S) : \\ &\quad iteration(S, \varphi, \tau_S) = iteration(S, \psi, \tau_S) \\ &\quad \Rightarrow \varphi^t + t_{min} \leq \psi^t \end{aligned}$$

D[b] A temporal trace τ_S complies with a maximum time lag $t_{max} \geq 0$ between activities A and B if for the respective events e_A and e_B the following condition holds true

[1] A negative minimum time lag is be equal to maximum time lag with the same positive value and activities A and B swapped.

$$\forall \varphi \in occurrences(S, e_A, \tau_S) \forall \psi \in occurrences(S, e_B, \tau_S) : \\ iteration(S, \varphi, \tau_S) = iteration(S, \psi, \tau_S) \\ \Rightarrow \varphi^t + t_{max} \geq \psi^t$$

D[c] A temporal trace τ_S complies with an interval time lag $[t_{min}, t_{max}]$ with $t_{max} \geq 0$ and $t_{min} \leq t_{max}$ between activities A and B if for the respective events e_A and e_B the following condition holds true

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

The condition for a minimal time lag for example expresses, that for all occurrence of Events e_A and e_B which are in the same Iteration the time of the occurrence of Event e_A needs to happen at least t_{min} time units before the occurrence of Event e_B, i.e. that there is a minimum time lag of t_{min} time units between the respective Events of the two activities A and B.

Pattern Sematics 2 (TP2: Durations). The semantics for pattern TP2 can easily be defined based on the compliance rules of pattern semantics 1. Thereby, e_A corresponds to the start event of the respective process element, while e_B corresponds to its end event. Based on Design Choice C we then have the following relations for events e_A and e_B.

Design Choice C

C[a] e_A is the start event and e_B is the end event of the activity

C[b] e_A is the start event of the first activity to start of the activity set and e_B is the end event of the last activity to end of the activity set

C[c] e_A is the start event and e_B is the end event of the process

C[d] e_A is the start event of the first process instance of the set of process instances to be started and e_B is the end event of the last process instance of the set of process instances to be finished

Beside these differences in events e_A and e_B the same rules apply as for pattern semantics 1.

Design Choice D

D[a] A temporal trace τ_S complies with a minimum duration $t_{min} \geq 0$ for activity A if the following condition holds true

$$\forall (\varphi_S, \varphi_E) \in occurrences(S, A, \tau_S) : \\ \varphi_S^t + t_{min} \leq \varphi_E^t$$

D[b] A temporal trace τ_S complies with a maximum duration $t_{max} \geq 0$ for activity A if the following condition holds true

$$\forall(\varphi_S, \varphi_E) \in occurrences(S, A, \tau_S) : \\ \varphi_S^t + t_{max} \geq \varphi_E^t$$

D[c] A temporal trace τ_S complies with a interval duration $[t_{min}, t_{max}$ for activity A if the following condition holds true

$$\forall(\varphi_S, \varphi_E) \in occurrences(S, A, \tau_S) : \\ \varphi_S^t + t_{min} \leq \varphi_E^t \leq \varphi_S^t + t_{max}$$

Pattern Sematics 3 (TP3: Time Lags between Events). Again, the semantics for time pattern TP3 can easily be defined based on pattern semantics 1. However, no restriction regarding events e_A and e_B apply. Thus e_A and e_B can now be arbitrary Events in $\mathcal{E}$. Besides this exactly the same rules apply as for pattern semantics 1.

The definition of the last three pattern semantics easily leads to the assumption, that we could just drop the first two time patterns and only use pattern TP3 combine with pattern semantics 3. From a point of view of the just defined pattern semantics this may be true. However, the main difference between time pattern TP1, TP2 and TP3 does not lie in their execution semantics, but in their semantics in respect to other topics like escalation and escalation handling, how agents responsible for a missed time constraint are determined and how the compliance with a respective time constraint can be enforced by the process engine.

For example, for a duration constraint there can always be one or more agents determined which can be held responsible for the fulfillment of the respective time constraint, namely the agent responsible for the activity or the process instance. This is not always true for time lags between activities or time lags between arbitrary events. In fact it may only be true in very rare cases where the time lag more or less corresponds to a duration. Thus other ways need to be found to enforce the fulfillment of such a time constraint, like for example reducing the duration of the involved activities, or escalating the whole process instead of just one activity.

At this point we would like to stress one other important point concerning pattern semantics 1-3: They do not require, that any on the respective events needs to occur in a temporal execution trace or an instance type. This means that a maximal time lag can be fulfilled although it's second event never occurred. For events bound to the start or end of an activity or a process this poses no problem since their occurrence can be guaranteed be the control flow. For other events, which are e.g. triggered by an external source, other way's for ensuring their occurrence need to be found if necessary.

Pattern Category II The time patterns in this category restrict the execution points of process elements and thus the possible time points for the occurrence of the respective Events.

In most cases the value of a fixed date element (cf. Fig. 8) not only depends on process schema S, but also on the current process instance (i.e., trace τ_S) and the current iteration I of the respective process element. To capture this we define the function fde:

Definition 8 (Fixed Date Element). *fde is a function, which returns for each process element A with a fixed date element and each iteration I the current value of the fixed date element. Formally:* $fde : \mathcal{PS} \times \mathcal{A}_S \times \mathcal{I}_S \times \mathcal{T}_S \mapsto \mathcal{C}$

Therefore fde effectively represents the Fixed Date attached to each Fixed Date Element (cf. Fig. 8). Based on Definition 8 we can now provide a formalization of the semantics of time pattern TP4:

Pattern Sematics 4 (TP4: Fixed Date Elements). A fixed date element for a process element A restricts the allowed time points for either the start or the end event. Thus we denote the respective event of process element A as e. Design Choice F now determines whether e is the start event (F[a] and F[b]) or the end event (F[c]) of the respective activity or process A (Design Choice C[a] vs. C[c]).

Compliance of a given trace with a fixed date element now means that all occurrences $\varphi = (e, \cdot)$ of event e need to fulfill the following compliance rules based on Design Choice G:

Design Choice F

F[a] A temporal trace τ_S is compliant with an earliest start date on process element A iff

$$\forall \varphi \in occurrences(S, e_{A_S}, \tau_S) : fde(S, A, iteration(S, e_{A_S}, \tau_S), \tau) \leq \varphi^t$$

where e_{A_S} is the start event of process element A.

F[b] A temporal trace τ_S is compliant with a latest start date on process element A iff

$$\forall \varphi \in occurrences(S, e_{A_S}, \tau_S) : fde(S, A, iteration(S, \varphi, \tau_S), \tau) \geq \varphi^t$$

where e_{A_S} is the start event of process element A.

F[c] A temporal trace τ_S is compliant with a latest completion date on process element A iff

$$\forall \varphi \in occurrences(S, e_{A_E}, \tau_S) : fde(S, A, iteration(S, \varphi, \tau_S), \tau) \geq \varphi^t$$

where e_{A_E} is the end event of process element A.

The schedule attached to a schedule restricted element (TP5, cf. Fig. 9) is an abstract representation of a possibly infinite set of time points or time frames, e.g. Mo-Fr 8:00-17:00 (cf. Fig. 20). Since we do not want to make any restrictions regarding the representation of such a schedule we simply require that it can be materialized as a set of subset of the time points of a calender $\mathcal{C}$.

Definition 9 (Schedule). *A schedule s is a possibly infinite set of subsets of the time points of a calendar $\mathcal{C}$, i.e. $s \subseteq 2^{\mathcal{C}}$.*

A schedule can either be a set of discrete time points $s = \{t | t \in \mathcal{C}\}$ or a set of intervals $s = \{[t_{min}, t_{max}] | [t_{min}, t_{max}] \subseteq \mathcal{C}\}$.

It is important to note, that the subsets of a schedule may have overlapping values.

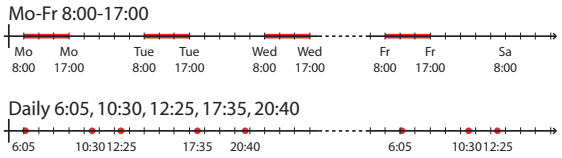


Fig. 20. Schedules

Based on this general definition of a schedule the pattern semantics of pattern TP5 (cf. Fig. 9) can now be formalized as follows:

Pattern Sematics 5 (TP5: Schedule Restricted Elements). As for pattern semantics 4, the schedule restricted element for a process element A restricts the allowed time points for either the start or the end event of A. Thus again we denote the respective event of process element A as e and it is determined by Design Choice F whether this is the start (F[a] and F[b]) or the end event (F[c]) of process element A.

Depending on Design Choice G the schedule associated with process element A is either materialized as a set of time points $s_A = \{t | t \in C\}$ or as a set of intervals $s_A = \{[t_1, t_2] | [t_1, t_2] \subseteq C\}$. An exception to the schedule is then materialized by removing and/or adding the respective time points or time intervals from/to the schedule.

To verify that a particular activity (process) instance complies with the schedule it needs to be checked whether the timestamp t of the respective event constitutes an element of the schedule (i.e. $t \in s_A$). Based on design choice F the compliance of a given trace τ_S with a schedule restricted element A is then defined as follows:

Design Choice F

F[a]and F[b] A temporal trace τ_S is compliant with a schedule s_A for the start date on process element A iff

$$\forall \varphi \in occurrences(S, e_{A_S}, \tau_S) : \varphi^t \in s_A$$

where e_{A_S} is the start event of process element A.

F[c] A temporal trace τ_S is compliant with a schedule s_A for the end date on process element A iff

$$\forall \varphi \in occurrences(S, e_{A_E}, \tau_S) : \varphi^t \in s_A$$

where e_{A_E} is the end event of process element A.

Time Pattern TP6 (Time Based Restriction) restricts the number of occurrences of a process element or a set of process elements within a time frame. There we first define the function *executionsPerTime* as follows:

Definition 10 (Executions Per Time frame). *executionsPerTime returns the number of executions of a particular activity within a given time frame. Formally:* $executionsPerTime : \mathcal{PS} \times \mathcal{A} \times \mathcal{C} \times \mathcal{C} \times \mathcal{T}_S \mapsto \mathbb{N}$

$$executionsPerTime(S, A, t_{min}, t_{max}, \tau_S) = |\{(\psi_S, \psi_E) \in occurrences(S, A, \tau_S) | [t_{min}, t_{max}] \cap [\psi_S^t, \psi_E^t] \neq \emptyset\}|$$

Based on Definition 10 we can now formalize the semantics of time pattern TP6 as follows:

Pattern Sematics 6 (TP6: Time Based Restrictions). Depending on Design Choice H a time based restriction is applied to a set of activities or processes denoted as Γ in the following. For a set of activities within one process instance compliance of a given trace τ_S with a Time Based Restriction on Γ is based on design choice I then defined as follows:

Design Choice I

I[a] A temporal trace τ_S is compliant with a maximum number of concurrent executions n on a set of activities Γ iff

$$\forall A \in \Gamma : \forall (\varphi_S, \varphi_E) \in occurrences(S, A, \tau_S) : \sum_{A' \in \Gamma} executinosPerTime(S, A', \varphi_S^t, \varphi_E^t, \tau_S) \leq n$$

I[b] As with pattern semantics 5 the time periods of a time based restriction on the number of executions per time period can be materialized as a schedule $s = \{[t_1, t_2] | [t_1, t_2] \subseteq C\}$ according to definition 9. A temporal trace τ_s is then compliant with a maximum number of executions n per element of schedule s iff:

$$\forall [t_{min}, t_{max}] \in s : \sum_{A \in \Gamma} executionsPerTime(S, A, t_{min}, t_{max}, \tau_S) \leq n$$

For a set of activities within different process instances or a set of processes this needs to apply for the respective set of temporal traces.

Time Pattern TP7 is a Variant of Time Pattern TP4 (Fixed Date Element) with the difference being that the value of a fixed date element depends on the current instance of process S and it's current iteration (cf. Pattern Semantics 4) and the value of a validity period does not. Thus the definition of semantics of pattern TP7 is similar to pattern semantics 4.

Pattern Sematics 7 (TP7: Validity Period). A validity period for a process element A restricts the allowed time points for either the start or the end event. Thus we denote the respective event of process element A as e. Design Choice F now determines whether e is the start event (F[a] and F[b]) or the end event (F[c]) of the respective activity or process A (Design Choice C[a] vs. C[c]).

Compliance of a given trace with a validity period now means that all occurrences $\varphi = (e, \cdot)$ of event e need to fulfill the following compliance rules based on Design Choice F:

Design Choice F

F[a] A temporal trace τ_S is compliant with an earliest validity date $v \in \mathcal{C}$ on the start event e_{A_S} of process element A iff

$$\forall \varphi \in \mathit{occurrences}(S, e_{A_S}, \tau_S) : v \leq \varphi^t$$

where e_{A_S} is the start event of process element A.

F[b] A temporal trace τ_S is compliant with a latest validity date $v \in \mathcal{C}$ on the start event e_{A_S} of process element A iff

$$\forall \varphi \in \mathit{occurrences}(S, e_{A_S}, \tau_S) : v \geq \varphi^t$$

where e_{A_S} is the start event of process element A.

F[c] A temporal trace τ_S is compliant with a latest validity date $v \in \mathcal{C}$ on the end event e_{A_E} of process element A iff

$$\forall \varphi \in \mathit{occurrences}(S, e_{A_E}, \tau_S) : v \geq \varphi^t$$

where e_{A_E} is the end event of process element A.

Pattern Category III Time Pattern TP8 allows varying the control flow depending on time aspects. Thus it restricts the set of events a compliant temporal trace may consist of, since the events of the selected path or activity need to occur within the trace and the events of the deselected paths or activities may not occur in the trace.

To formalize this pattern we define function *eval*, which allows to evaluate the respective condition for each of the possible paths with respect to the current

iteration I and the current process instance. Depending on the outcome the subtraces of the respective path (i.e. the traces producible by this path) must occur or must not occur in the trace.

First we give a formal definition of a path

Definition 11 (Path). *A Path P within a process schema $S \in \mathcal{PS}$ is a connected part of process schema S with a single entry and a single exit node where for each of the contained split-nodes the respective join-node is also contained and vice versa.*

A Path P can be represented as subset $\mathcal{E}_P$ of the events of process schema S, i.e. $\mathcal{E}_P \subset \mathcal{E}_S$, where $\mathcal{E}_P$ contains all events which may occur during the execution of path P.

In case of late binding a path only consists of a single activity representing the service chosen in this path. As an example a process schema with three paths can be found in figure 21.

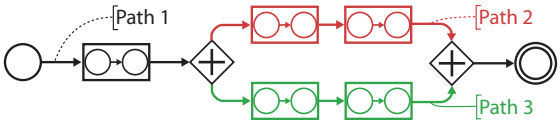


Fig. 21. Paths of a process schema

Based on the definition of paths (cf. Definition 11) we now define the function *eval* which evaluates the respective condition for a path:

Definition 12 (Eval). *eval is a function, which evaluates the condition associated with a path $\mathcal{E}_P$ with respect to the current iteration I. Formally*

$$eval : \mathcal{PS} \times 2^{\mathcal{E}_S} \times \mathcal{I}_S \times \mathcal{T}_S \mapsto \{true, false\}$$

The concrete implementation of the function *eval* depends on the formalizm used to desribe the conditions for path selection. Based on this definition the semantics of pattern TP8 is defined as follows:

Pattern Sematics 8 (TP8: Time Dependent Variability). A temporal trace τ_S is compliant with a time dependent variability iff for all paths P of trace τ_S the following condition is fulfilled

$$\begin{aligned}
&\forall I \in \mathcal{I}_S : \tau_S \vdash I \Longrightarrow \\
&\quad ((eval(S, \mathcal{E}_P, I, \tau_S) = false) \\
&\qquad \Rightarrow (\forall e \in \mathcal{E}_P : (\forall \varphi \in occurrences(S, e, \tau_S) : iteration(S, \varphi, \tau_S) \neq I))) \wedge \\
&\quad ((eval(S, \mathcal{E}_P, I, \tau_S) = true) \\
&\qquad \Rightarrow (\exists e \in \mathcal{E}_P : (\exists \varphi \in occurrences(S, e, \tau_S) : iteration(S, \varphi, \tau_S) = I)))
\end{aligned}$$

Pattern Category IV The time patterns in this category allow to express temporal relations between different occurrences of recurrent activities. Therefor loops and the current iteration of these loops play an important role in the definition of their semantics.

As stated earlier, Time Pattern TP9 is a variant of Time Pattern TP1. Instead of enforcing the two events to be inside the same iteration of a loop, the second event now lies in the iteration succeeding the one to which the first event belongs.

Since the value of the time lag between the different iterations may depend on the current iteration we define function *cycleTime* as follows:

Definition 13 (Cycle Time). *cycleTime is a function, which returns for each tuple of process elements A and B which are Cyclic Elements and each iteration I the current value of the time lag. Formally:*

$$cycleTime : \mathcal{PS} \times \mathcal{A}_S \times \mathcal{A}_S \times \mathcal{I}_S \times \mathcal{T}_S \mapsto 2^{\mathcal{C}}.$$

Where the result of $cycleTime(S, A, B, I, \tau_S)$ *is a continuous subset of* $\mathcal{C}$*, i.e. an interval, or a single element of* $\mathcal{C}$*.*

Pattern Sematics 9 (TP9: Cyclic Elements). A time lag between two activities A and B can be mapped onto a time lag between the respective start and/or end events of these activities based on Design Choice N (cf. Fig. 14). Thus we denote the respective event of activity A as e_A and the respective event of activity B as e_B. Based on design choice E we then have the following relations between activity A and event e_A and activity B and event e_B respectively:

Design Choice N

N[a] e_A is the start event of activity A and e_B is the start event of activity B (start-start)

N[b] e_A is the start event of activity A and e_B is the end event of activity B (start-end)

N[c] e_A is the end event of activity A and e_B is the start event of activity B (end-start)

N[d] e_A is the end event of activity A and e_B is the end event of activity B (end-end)

Compliance of a given trace with such a time lag between two cyclic elements now means that all occurrences $\varphi = (e_A, \cdot)$ and $\psi = (e_B, \cdot)$ of the two events fulfill the given time lag:

$$\begin{aligned} &\forall \varphi \in occurrences(S, e_A, \tau_S) \forall \psi \in occurrences(S, e_B, \tau_S) : \\ &\quad iteration(S, \varphi, \tau_S) \oplus = iteration(S, \psi, \tau_S) \\ &\quad \Rightarrow \psi^t - \varphi^t \in cycleTime(S, A, B, iteration(S, \varphi, \tau_S), \tau_S) \end{aligned}$$

Based on Design Choice L the value of $cycleTime(S, A, B, I, \tau_S)$ is either a single value, a time interval or may vary depending on the current iteration.

Last but not least, Time Pattern TP10 allows specifying periodically recurring process elements according to an explicit periodicity rule (cf. Fig. 15). Pattern TP10 is actually a rather abstract concept covering a lot of different cases related to recurring process elements.

To formalize Pattern TP10 we assign a schedule to each of the process element participating in the periodicity, which constitutes the respective periodicity rule. More formally:

Definition 14 (Periodicity Rule). *A periodicity rule can be materialized as a set of schedules $\mathcal{S}$, where each of the events e_A of the process elements A participating in the periodicity receives its own schedule s_{e_A}. These schedules are not independent from each other but correlated by the respective periodicity rule.*

Exceptions to the periodicity rule can be applied to the schedules by either adding or removing the respective exceptional elements to or from the schedules.

Based on this "materialization" of the periodicity rule the semantics for pattern TP10 can be formalized as follows:

Pattern Sematics 10 (TP10: Periodicity). A temporal trace τ_S is compliant with a periodicity rule if all occurrences of the respective events occur within an element of the respective schedule and for each element of the schedule there is a corresponding event occurrence. Let $\mathcal{E}_P$ be the set of events involved in the periodicity, then a temporal trace τ_S is compliant with the periodicity rule represented by the schedules in $\mathcal{S} = \{s_e | e \in \mathcal{E}_P\}$ iff

$$\begin{aligned} &\forall e \in \mathcal{E}_P : \\ &\qquad (\forall \varphi \in occurrences(S, e, \tau_S) : \varphi^t \in s_e) \wedge \\ &\qquad (\forall i \in s_e : \exists! \varphi \in occurrences(S, e, \tau_S) : \varphi^t \in i) \end{aligned}$$

where s_e is the schedule associate with event e and $i \in s_e$ is a single subset of schedule s_e (cf. Definition 9).

6 Evaluation

In the following we describe the evaluation of selected approaches from academia and industry regarding their support for time patterns. Section 6.1 describes our evaluation methodology, while Section 6.3 shows our evaluation results.

6.1 Evaluation Methodology

This section sketches the methodology employed for conducting our evaluation. In particular, we describe evaluation goal, evaluation objects, evaluation criteria, evaluation metrics, and the evaluation procedure.

Definition of Evaluation Goal. The goal of our evaluation is to measure how well current PAISs cope with time aspects.

Selection of Evaluation Objects. As evaluation objects we choose process management systems, calendar systems, and project planning tools from both academia and industry. In terms of academic approaches our evaluation considers the proposals made by Eder [3], Bettini [5], and Combi [1]. As samples for commercial systems our evaluation includes the process management systems MQSeries Workflow and Tibco iProcess Modeller, for which we have hands-on experience as well as running installations in our labs. Further, we include the widely used calendar systems Outlook, Google Calendar and Lightning, and the well-known project management tool MS Project. Finally, with BPMN our evaluation comprises a commonly used process modeling language.

Definition of Evaluation Criteria and Metrics. Evaluation criteria are the 10 change patterns described in Section 4. We measure the ability of a PAIS to deal with time aspects as the degree of support for the described evaluation criteria. For each evaluation criterion we differentiate between *supported*, *partially supported*, *not supported*, and *not specified*. If an evaluation object provides support for a particular criterion the supported design choices are listed. If a particular evaluation object is only partially supported this is additionally labeled with "*". No support is labeled with "-" and *not specified* with "?". Assume that an evaluation object supports Pattern TP4 with Design Choices C and F. Further assume that for Design Choice C Option a is partially supported and for Design Choice F Option c is supported. This would result in the String "C[a*], F[c]" (e.g., Eder et al. in Fig. 22).

Analyzing the Evaluation Objects along the Evaluation Criteria. For the academic approaches we base our evaluation on a comprehensive literature study. Regarding commercial systems, support for time patterns was determined based on the installations in our lab and on our hands-on experience with respective systems. A summary of our evaluation results is given in Fig. 22. An in-depth description of each of the evaluated approaches including a detailed description of all supported design choices can be found in the next section. Note that this evaluation only considers time patterns. Time features like verification of time constraints, escalation mechanisms or scheduling support are outside of the scope of this paper.

6.2 Evaluation Details

Evaluation Results: Outlook - Lightening - Google Calendar *MS Outlook 2007*, *Mozilla Lightening 0.9* and *Google Calendar* are widely used calendar systems. Due to their similarities in respect to time support the evaluation re-

sults of these three calendar systems are discussed in conjunction with each other.

General Design Choices. In terms of general design choices the evaluated calendar systems support Design Choice A with Option c, while Options a and b are not applicable. Design Choice B, in turn, is supported with Options a and b (Design Choice B[a,b]). As basic granularities, Day, Week, Month and Year are supported. In addition, Minute and Hour are supported when specifying durations. All calendar systems support system-defined granularities. *Google Calendar* supports Working Days, each Monday, Wednesday and Friday as well as each Tuesday and Thursday. *Lightening*, in turn, supports Bi-Weekly and Working Days while *Outlook* only supports Working Days.
Category I - Durations and Time Lags. The evaluated calendar systems only support Pattern TP2 (*Durations*) from Category I. Thereby, durations can only be specified for single activities (i.e., tasks) or events (i.e., calendar items) (Design Choice C[a]) and are treated as maximum values (Design Choice D[b]).
Category II - Restrictions of Process Execution Points. In this pattern category support is provided for Pattern TP4 (*Fixed Date Elements*). In particular, the pattern can be applied to single activities. *Google Calendar* and *Outlook* additionally support the invitation of different users to appointments. This can be considered as partial support for multiple instances of a particular activity (Design Choice C[a,b*[2]]). Both *Outlook* and *Lightening* provide support options a and c of Design Choice F, as earliest start dates and latest completion dates can be specified (Design Choice F[a,c]). *Google Calendar*, in turn, only provides support for Option a (Design Choice F[a])
Category III - Variability. This pattern category is not supported by any of the calendar systems.
Category IV - Reoccurring Process Elements. This pattern category is supported pretty well by all evaluated calendar systems. Pattern TP9 (*Cyclic Elements*) is supported with Design Choice L[a], i.e., time lags between the different cycles are always the same. The number of cycles can either be fixed (i.e., 5 cyles) or be calculated depending on the time lag and the end date (e.g., every 2 weeks until the end of the year) (Design Choice K[a,b]). Pattern TP10 (*Periodicity*), in turn, is supported with Design Choice O[a], i.e., the periodicity rule can only contain one date. Like for TP9 the number of cycles can either be fixed (i.e., 5 cyles) or be calculated depending on the time lag between cycles and the end date (e.g., every 2 weeks until the end of the year) (Design Choice K[a,b]). Finally exceptions to a periodicity rule have to be defined manually by deleting single appointments.

Evaluation Results: MS Project 2007 *MS Project* is a commonly used commercial project planning system.

General Design Choices. In terms of general design choices *MS Project* supports Design Choice A with Options a and c, while Option b is not applicable

[2] A * means, the respective design choice is only partially supported based on some workaround.

(Design Choice A[a,c]). Design Choice B, in turn, is supported with Options a and b (Design Choice B[a,b]). As basic granularities Day, Week, Month, and Year are supported. In addition, Minute and Hour are supported when specifying durations. In terms of system-defined granularities *MS Project* supports Working Days as well.

Category I - Durations and Time Lags. *MS Project* supports Pattern TP1 (*Time Lags between Activities*) and Pattern TP2 (*Durations*) from Category I. In the context of Pattern TP1, Design Choice E is fully supported; i.e., start-start relations, start-end relations, end-start relations, and end-end relations can be expressed (Design Choice E[a,b,c,d]). Time lags are minimum values (Design Choice D[a]). Time lags can only be expressed between two directly succeeding activities, since the control flow is modeled through time dependencies. Regarding Pattern TP2 durations can be only specified for single activities (i.e., tasks) (Design Choice C[a]), and can only be maximum ones (Design Choice D[b]).

Category II - Restrictions of Process Execution Points. In this pattern category, support is provided for Patterns TP4 (*Fixed Date Elements*) and TP5 (*Schedule Restricted Elements*). Pattern TP4 can be applied to single activities, but also to a whole process instance (i.e., an entire project) (Design Choice C[a,c]). Design Choice F is supported with Option a, allowing for the specification of the earliest start dates only (Design Choice F[a]). Regarding Pattern TP5, in turn, Design Choice C is supported for single activities (i.e., for each activity a special calendar can be created) (Design Choice C[a]). Schedule entries correspond to time frames (i.e., special working hours can be specified) (Design Choice G[b]). Finally, it is possible to specify exceptions like days off work.

Category III - Variability. This pattern category is not supported by *MS Project*.

Category IV - Reoccurring Process Elements. This pattern category is reasonably supported. For Pattern TP9 (*Cyclic Elements*), Design Choice L is supported with Option a, i.e., time lags between different cycles are always the same (Design Choice L[a]). In addition, support for Design Choice K[a,b] is provided – the number of cycles can either be fixed (i.e., 5 cyles) or be calculated depending on the time lag and the end date (e.g., every Monday at 11:30 until the end of the year). Pattern TP10 (*Periodicity*), in turn, is supported with Design Choice O[a], i.e., any periodicity rule may only contain one date. Like for TP9 the number of cycles can either be fixed (i.e., 5 cyles) or be calculated depending on the time lag and the end date (Design Choice K[a,b]). Finally exceptions to a periodicity rule have to be defined manually by deleting single appointments.

Evaluation Results: Business Process Modelling Notation 1.2 The *Business Process Modelling Notation (BPMN)* [19] is a process modelling standard published by the Object Management Group (OMG). It was specifically designed to provide standardized notations and diagramming conventions for the description of business processes.

General Design Choices. Using *BPMN* it becomes possible to set time parameters during build-time as well as determining them during run-time at the time they are needed for further process execution (Design Choice A[a,c]). Concerning Design Choice B, it might be possible to use basic time granularities, as well as system-defined and user-defined ones (Design Choice B[a?[3],b?,c?]); since the formalism used for specifying time parameters is not preassigned no definite conclusion can be made.
Category I - Durations and Time Lags. Regarding Pattern TP1, *BPMN* only allows to express end-start relations (Design Choice E[c]). Additionally, only relations between two directly succeeding activities are supported. Regarding Design Choice D, Option a is fully supported by adding an Intermediate-Timer-Event on the sequence flow connecting the activities; Option b may be emulated by adding an event-based decision between the two activities with an Intermediate-Timer-Event parallel to the second activity. *BPMN* supports time pattern TP2 with Design Choices C[a,c*] and D[b], i.e., it allows to specify the maximum duration of an activity by attaching an Intermediate-Timer-Event to the activity. By adding a cancelling discriminator [6] together with an Intermediate-Timer-Event in parallel to the whole process it also becomes possible to emulate Option c of Design Choice C.
Category II - Restrictions of Process Execution Points. In this pattern category *BPMN* only provides support for pattern TP4 (*Fixed Date Elements*). Design Choice C is fully supported, i.e., it is possible to add a fixed date to an activity or multi-instance activity (by putting an Intermediate-Timer-Event on the sequence flow leading to the respective element) or to a process (by using a Start-Timer-Event) (Design Choice C[a,b,c]). Due to the modeling technique using timers, for Design Choice F only Option a is supported.
Category III - Variability. This Pattern Category is supported by using an event-based XOR in combination with an Intermediate-Timer-Event (Design Choice J[a,b]).
Category IV - Reoccurring Process Elements. *BPMN* supports this pattern category to a certain degree. Concerning TP9 (*Cyclic Elements*) design choice L[a*,b*] is partially supported by adding an Intermediate-Timer-Event on the sequence flow connecting the iterations. Additionally, support for Design Choice K[a,c] is provided as the number of cycles can either be fixed (e.g. 5 iterations) or be dependent on an exit condition. Regarding pattern TP10 (*Periodicity*), Design Choice O is partially supported with Option a, whereas support for Option b depends on the formalism used for specifiying time, which as stated before, is not preassigned. As with TP9 the number of cycles may either be fixed or be dependent on an exit condition (Design Choice K[a,c]). Whether or not exceptions to the periodicity rule are supported again depends on the formalism used for specifying time constructs.

Evaluation Results: IBM WebShere MQ Workflow Buildtime 3.4 *MQ Workflow Buildtime* is the graphical process definition tool that ships with IBM

[3] A ? means, that based on the available material we could not decide whether or not the respective design choice is supported.

WebSphere MQ Workflow.

General Design Choices. Regarding general design choices, *MQ Workflow* supports setting the time parameters during build-time or specifying a data container which provides them during runtime (Design Choice A[a,c]). Furthermore, *MQ Workflow* supports the basic time granularities Year, Month, Week, Day, Hour, Minute and Second (Design Choice B[a]).
Category I - Durations and Time Lags. In Category I, *MQ Workflow* only supports pattern TP2. More precisely, it is possible to specify the maximum duration (Design Choice D[b]) of a single activity or process (Design Choice C[a,c]).
Category II - Restrictions of Process Execution Points. As the only one of the systems evaluated by us, *MQ Workflow* allows for the specification of a Validity Date (TP7). Thereby, it is possible to specify the earliest possible start date (Design Choice F[a]) of a process (Design Choice C[c]). None of the other patterns in this category is supported.
Category III - Variability. This pattern category is not supported by *MQ Workflow*.
Category IV - Reoccurring Process Elements. This pattern category is not supported by *MQ Workflow*.

Evaluation Results: Tibco iProcess Modeller 10.3.5 *Tibco iProcess Modeller* is one of the most popular workflow tools in practice.

General Design Choices. *Tibco iProcess Modeller* allows specifying time parameters during build-time as well as determining them during process execution (Design Choice A[a,c]). Concerning Design Choice B, *Tibco iProcess Modeller* supports basic time granularities; i.e., Year, Month, Week, Day, Hour, and Minute. In addition, it is possible to use Working Days when specifying time expressions (Design Choice B[a,b]).
Category I - Durations and Time Lags. From this category only Pattern TP2 (*Durations*) is supported. Thereby, it is only possible to specify the maximum (Design Choice D[b]) duration of a single activity (Design Choice C[a]).
Category II - Restrictions of Process Execution Points. *Tibco iProcess Modeller* does not allow for the restriction of process execution points in any way.
Category III - Variability. It is not possible to vary the control flow depending on time time aspects.
Category IV - Reoccurring Process Elements. None of the patterns of Category IV is supported.

Evaluation Results: Eder et al. [3] Eder et al. [3] discusses an approach for calculating activity deadlines such that all time constraints are satisfied and the overall process deadline can be met.

General Design Choices. Concerning general design choices, time parameters may be set during build-time, be fixed at process instantiation time or be determined during runtime (Design Choice A[a,b,c]). Only one basic granularity is considered, i.e., [3] does not provide direct support for Design Choice B.
Category I - Durations and Time Lags. In this category, TP1 (*Time Lags between Activities*) and Time Patterns TP2 (*Durations*) are considered. Time Lags between activities (TP1) may be specified in terms of end-end relations. However, since the durations of the activities are considered to be deterministic, it is possible to also simulate start-start, start-end and end-start relations by adding the duration of the first and/or substracting the duration of the second activity from the time lag (Design Choice E[a*,b*,c*,d]). Time Lags can be represented as minimum, as maximum or as time intervall (Design Choice D[a,b,c]). Regarding Time Pattern TP2, Option a of Design Choice C is supported; Option c can be simulated by using a time lag between the first and the last activity of the process (Design Choice C[a, c*]). Regarding Design Choice D, Options a and b are supported (Design Choice D[a,b]), but only one duration value may be specified per activity, the concrete kind of which is set by the respective implementation. Thus each implementation only supports one of these three Options.
Category II - Restrictions of Process Execution Points. [3] indicates support of Fixed Date Elements (TP4) by using a schedule with only one valid date. However, the case in which for a specific point in time no further date is available from the Schedule is not considered in the algorithm (Design Choice C[a*]). Since [3] just considers end events of activities it is possible to specify the latest completion time of an activity (Design Choice F[c]). As aforementioned, TP5 (*Schedule Restricted Elements*) is supported. In particular it is possible to specify a schedule for an activity (Design Choice C[a]), which can either consist of several discrete points in time or time frames (Design Choice G[a,b]). Additionally these schedules can support exceptions. Detailed information on the implementation of the schedules is not available.
Category III - Variability. Support for this pattern is not explicitly considered, but depends on the underlying workflow management system.
Category IV - Reoccurring Process Elements. Since [3] does not consider the repetitive execution of process elements, no support for this category is given.

Evaluation Results: Bettini et al. [5] Bettini et al. [5] investigate the calculation of enactment schedules for activities, which guarantee, that all temporal dependencies are met.

General Design Choices. Regarding general design choices, [5] supports Design Choice A with Option a, while Options b and c are not discussed (Design Choice A[a,b?,c?]). Moreover support for basic time granularities as well as user-defined granularities (e.g. business days) is provided (Design Choice B[a,c]).
Category I - Durations and Time Lags. Pattern Category IV is broadly supported by this approach. For Pattern TP1 (*Time Lags between Activities*) all four types of relations are supported, i.e., start-start, start-end, end-start, and

end-relation (Design Choice E[a,b,c,d]). Again, time lags can be set in terms of minimum, maximum or time interval (Design Choice D[a,b,c]). *Durations* (TP2) can be specified for a single activity or for a whole process (by specifying a time lag between the start of the first and the end of the last activity) (Design Choice C[a,c]). Like for TP1, minimum, maximum or time intervals can be set for durations (Design Choice D[a,b,c]). Pattern TP3 is supported with Design Choice D[a,b,c], i.e., time lags between events can be represented as minimum, as maximum or as time interval.
Category II - Restrictions of Process Execution Points. [5] supports the specification of *Fixed Date Elements* (TP4) for single activity instances (Design Choice C[a]) by adding a time lag between an artificial event at time point 01.01.0000 and the activity in question. Thereby it is possible to specify the the earliest start date , the latest start date, the latest completion date (Design Choice F[a,b,c]) of the activity.
Category III - Variability. Support for this pattern category is not explicitly expressed, but depends on the used workflow management system.
Category IV - Reoccurring Process Elements. [5] does not consider the repetitive execution of process elements, i.e., no support for this pattern category is provided.

Evaluation Results: Combi et al. [1] Combi et al. [1] discusses the conceptual modelling of temporal constraints in the medical domain and provides rather broad support for our time patterns.

General Design Choices. In terms of general design choices, [1] supports Design Choice A with Option a, while Option b and Option c are not discussed (Design Choice A[a,b?,c?]). Combi et al. [1] considers different time granularities, slicing the time domain into a sequence of granules. However, no details are provided on which granules are supported. User-defined granularities are partially supported (i.e., only granularities without laps are considered) (Design Choice B[a,c*]).
Category I - Durations and Time Lags. [1] supports Pattern TP1 (*Time Lags between Activities*), Pattern TP2 (*Durations*) and partially supports Pattern TP3 (*Time Lags between Events*). For Pattern TP1, Design Choice E is fully supported, i.e., start-start relations, start-end relations, end-start relations, and end-end relations can be expressed (Design Choice E[a,b,c,d]). Time lags can either be minimum values, maximum values or time intervals (Design Choice D[a,b,c]). Durations in the context of Pattern TP2 can be specified for single activities as well as for an entire process (by specifying a time lag between the start of the first and the end of the last activity) (Design Choice C[a,c]). Durations can either be minimum values, maximum values, or time intervals (Design Choice D[a,b,c]). Pattern TP3 is supported with Design Choice D[a*,b*,c*]. Time lags cannot be defined between arbitrary events as at least on of the events has to be a start or completion event of an activity (e.g., a time lag between the reaching of a milestone and the start of a subsequent activity).

Category II - Restrictions of Process Execution Points. In this pattern category support is provided for Pattern TP4 (*Fixed Date Elements*) and Pattern TP5 (*Schedule Restricted Elements*). Pattern TP4 can be applied to single activities, but also to multiple instances of a single activity (i.e., Multitask with Fixed Date Element) (Design Choice C[a,b]). Design Choice F is supported with Options a and c, allowing for the specification of earliest start date and latest end date (Design Choice F[a,c]). Thereby, absolute constraints restrict the interval during which an activity can be performed. For Pattern TP5, in turn, Design Choice C is supported for single activities (i.e., for each activity a special calendar can be created) (Design Choice C[a]). Schedule entries can be discrete points as well as time frames (Design Choice J[a,b]). Support for exceptions, in however, is not provided.
Category III - Variability. Support for this pattern is not explicitly addressed, but depends on the underlying workflow management system.
Category IV - Reoccurring Process Elements. This pattern category is supported very well by this approach. For Pattern TP9 (*Cyclic Elements*), Design Choice L is supported with Option a; i.e., the time lags between the different cycles are always the same (Design Choice L[a]). In addition, support for Design Choice K[a,b,c] is provided as the number of cycles can either be fixed (i.e., 5 cyles), be calculated depending on the time lag and the end date, or depend on the exit condition. Pattern TP10 (*Periodicity*), in turn, is supported with Design Choice O[a], i.e., periodic constraints on loop activities can be expressed. Like for TP9 the number of cycles can either be fixed (i.e., 5 cyles), be calculated depending on the time lag and the end date, or depend on the exit condition (Design Choice K[a,b,c]). Finally, it is not possible to specify exceptions to a periodicity rule.

6.3 Evaluation Results

Fig. 22 shows which time patterns are supported by our evaluation objects. Calendar systems like MS Outlook, Google Calendar and Lightening provide good support for Pattern TP4 (*Fixed Date Element*) and Pattern TP10 (*Periodicity*), while limited support is provided for specifying business rules and regulations (i.e., patterns TP1, TP2, TP3 and TP6). In addition to the support provided by calendar systems, project management tools like MS Project provide some support for specifying business rules and regulations. However, project management systems lack operational support for multiple concurrently executed process instances. BPMN as a representative of a process modeling language provides limited support for time aspects. The support for time constraints in commercial workflow management systems is even more limited and restricted to the definition of maximum execution durations. Academic approaches are comparably more expressive and provide good support for specifying business rules and regulations. However, except for the proposal of Combi et al. [1] the evaluated approaches do not consider any loops resulting in missing support for patterns of Category IV (*Reoccurring Process Elements*). Interestingly, despite their relevance for real world applications support for Pattern TP6 (*Time*

Based Restrictions) and Pattern TP7 (*Validity Periods*) is missing in almost all evaluation objects.

7 Related Work

Patterns were first used by Alexander [20] to describe solutions to recurring problems and best practices in architectural design. Patterns also have a long tradition in computer science. Gamma et al. [21] 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 [6, 22]. In this context, control flow patterns describe constructs to specify activities and their ordering. In addition, workflow data patterns [7] provide ways for modeling the data aspect in PAIS while workflow resource patterns [8] describe how resources can be represented in workflows. Furthermore, patterns for describing control-flow changes [9] and service interactions were introduced [23]. 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 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 [2, 5, 1, 3], escalation management [13], and scheduling support [14, 15]. The effect of ad-hoc changes on temporal constraints is investigated in [24]. A systematic investigation of requirements for time support from different heterogeneous application domains is missing so far.

8 Summary and Outlook

We have proposed 10 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 ability to cope with time constraints. We 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'07. (2007) 70 – 81

Patterns	Calendar Systems	Project Management	Standards	Commercial		Academic		
	Outlook 2007, Lightning 0.9, Google Calendar	MS Project	BPMN	MQ Workflow	TIBCO iProcess Modeller	Eder et al.	Bettini et al.	Combi et al.
General Design Choices	A[c], B[a, b]	A[a, c], B[a, b]	A[a, c], B[a?, b?, c?]	A[a, c], B[a]	A[a, c], B[a, b]	A[a, b, c]	A[a, b?, c?], B[a, c]	A[a, b?, c?], B[a, c*]
Category I: Durations and Time Lags								
TP1 – Time Lags between Activities	–	D[a], E[a, b, c, d], F[a]	D[a, b*], E[c], F[a]	–	–	D[a, b, c], E[a*, b*, c*, d], F[b]	D[a, b, c], E[a, b, c, d], F[b]	D[a, b, c], E[a, b, c, d], F[b]
TP2 – Durations	C[a], D[b]	C[a], D[b]	C[a, c*], D[b]	C[a, c], D[b]	C[a], D[b]	C[a, c*], D[a, b][1]	C[a, c], D[a, b, c]	C[a, c], D[a, b, c]
TP3 – Time Lags between Events	–	–	–	–	–	–	D[a, b, c]	D[a, b, c]*
Category II: Restrictions of Process Execution Points								
TP4 – Fixed Date Elements	C[a, b*], F[a, c]	C[a, c], F[a]	C[a, b, c], F[a]	–	–	C[a]*, F[c]	C[a], F[a, b, c]	C[a, b], F[a, c]
TP5 – Schedule Restricted Elements	–	C[a], F[a], G[b]	–	–	–	C[a], F[c], G[a, b]	–	C[a], F[a, c], G[a, b]
TP6 – Time Based Restrictions	–	–	–	–	–	–	–	–
TP7 – Validity Period	–	–	–	C[c], F[a]	–	–	–	–
Category III: Variability								
TP8 – Time Dependent Variability	–	–	J[a, b]	–	–	?	?	?
Category IV: Reoccurring Process Elements								
TP9 – Cyclic Elements	K[a, b], L[a], N[a]	K[a, b], L[a], N[a]	K[a, c], L[a*, b*], N[c]	–	–	–	–	K[a, b, c], L[a], N[a, b, c, d]
TP10 – Periodicity	K[a, b], O[a]	K[a, b], O[a]	K[a, c], O[a*, b?]	–	–	–	–	K[a, b, c], O[a]

Symbols:
A[x] Support of Design Choice A with option x.
A[x*] Option x is partially supported
– Not supported
? Not specified

[1] Only one duration value is support which can either be minimum or maximum, depending on the concrete implementation

Fig. 22. Evaluation Results

2. Marjanovic, O., Orlowska, M.E.: On modeling and verification of temporal constraints in production workflows. Knowl. Inf. Syst. **1** (1999) 157–192
3. Eder, J., Panagos, E., Rabinovich, M.: Time constraints in workflow systems. In: CAiSE'99. (1999) 286–300
4. Dadam, P., Reichert, M., Kuhn, K.: Clinical workflows – the killer application for process-oriented information systems? In: Proc. Int'l Conf. on Business Information Systems (BIS'00), Poznan, Poland (2000) 36–59
5. Bettini, C., Wang, X.S., Jajodia, S.: Temporal reasoning in workflow systems. In: Distributed and Parallel Databases. (2002) 269–306
6. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. Distributed and Parallel Databases **14** (2003) 5–51
7. 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)
8. Russell, N., ter Hofstede, A., Edmond, D., van der Aalst, W.: Workflow Resource Patterns. Technical Report WP 127, Eindhoven Univ. of Technology (2004)
9. 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** (2008) 438–466
10. Thom, L., Reichert, M., Iochpe, C.: Activity patterns in process-aware information systems: Basic concepts and empirical evidence. International Journal of Business Process Integration and Management (IJBPIM) **4** (2009) 93–110
11. Russell, N., van der Aalst, W., ter Hofstede, A.: Exception Handling Patterns in Process-Aware Information Systems. In: Proc. CAiSE'06. (2006) 288–302
12. Müller, D., Herbst, J., Hammori, M., Reichert, M.: IT Support for Release Management Processes in the Automotive Industry. In: Proc. BPM'06. (2006) 368–377
13. van der Aalst, W.M.P., Rosemann, M., Dumas, M.: Deadline-based escalation in process-aware information systems. Decision Support Systems **43** (2007) 492–511
14. Combi, C., Pozzi, G.: Task scheduling for a temporal workflow management system. In: Proc. TIME'06. (2006) 61–68
15. Eder, J., Pichler, H., Gruber, W., Ninaus, M.: Personal schedules for workflow systems. In: Proc. BPM'03. (2003) 216–231
16. Schultheiß, B., Meyer, J., Mangold, R., Zemmler, T., Reichert, M.: Designing the processes for chemotherapy treatment in a Women's Hospital (in German). (1996)
17. German Association of the Automotive Industry: Engineering Change Management. Part 1: Engineering Change Request, V 1.1., Doc. No. 4965 (2005)
18. Bobrik, R.: Konfigurierbare Visualisierung komplexer Prozessmodelle. PhD thesis, Univ. of Ulm (2008)
19. Object Management Group: Business Process Modeling Notation (BPMN) Version 1.2. http://www.omg.org/spec/BPMN/1.2 (2009)
20. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language. Oxford University Press, New York (1977)
21. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
22. Puhlmann, F., Weske, M.: Using the Pi-Calculus for Formalizing Workflow Patterns. In: Proc. BPM'05. (2005) 153–168
23. Barros, A., Dumas, M., ter Hofstede, A.: Service Interaction Patterns. In: Proc. BPM'05. (2005) 302–318
24. Sadiq, W., Marjanovic, O., Orlowska, M.E.: Managing change and time in dynamic workflow processes. Int. J. Cooperative Inf. Syst. **9** (2000) 93–116