

# Time Patterns for Process-aware Information Systems

Andreas Lanz · Barbara Weber · Manfred Reichert

Received: 29 December 2011 / Accepted: 28 July 2012

**Abstract** Companies increasingly adopt process-aware information systems (PAISs) due to their promising perspectives for improved business process support. Although the proper handling of temporal constraints is crucial in this context, existing PAISs vary significantly regarding their support of the temporal perspective of a business process. To make PAISs comparable in respect to their ability to deal with temporal constraints and to facilitate the development of a time-aware PAIS, this paper suggests 10 time patterns. All patterns are based on empirical evidence we gathered in case studies. Additionally, they are validated through a systematic literature review. Based on the time patterns, we then provide an in-depth evaluation of selected PAISs and academic approaches. Altogether, the 10 time patterns will not only facilitate the selection of technologies for realizing time- and process-aware information systems, but can also be used as reference for implementing time support in PAISs.

**Keywords** Process-aware Information System, Workflow Patterns, Time Patterns, Temporal Perspective

---

Andreas Lanz  
Univ. of Ulm, Institute of Databases and Information Systems  
James-Frank-Ring, 89069 Ulm, Germany  
E-mail: andreas.lanz@uni-ulm.de

Barbara Weber  
University of Innsbruck, Department of Computer Science  
Technikerstraße 21a, 6020 Innsbruck, Austria  
E-mail: barbara.weber@uibk.ac.at

Manfred Reichert  
Univ. of Ulm, Institute of Databases and Information Systems  
James-Frank-Ring, 89069 Ulm, Germany  
E-mail: manfred.reichert@uni-ulm.de

## 1 Introduction

Today's enterprises crave for comprehensive knowledge about and control over their business processes. In particular, a profound understanding and control of these processes can lead to competitive advantages. In this context, process-aware information systems (PAISs) offer promising perspectives by enabling enterprises to define their business processes based on explicit process models as well as to execute these models in a controlled and efficient manner [69, 52].

The specification of a business process can be viewed from different perspectives [2, 26, 52]. The *control-flow perspective* describes the activities of a business process as well as their ordering and execution constraints. In turn, the *data perspective* connects activities with business and process data. Furthermore, the *resource perspective* provides a link between the process specification and the organizational structure, e.g., based on user roles assigned to process activities [54]. Finally, the *operational perspective* refers to the application services executed in the context of activities. However, another important perspective required for the broad acceptance and use of PAISs has received too little attention so far, namely the *temporal perspective*. Although there exists considerable work related to specific aspects of time in PAIS, so far, there has been no comprehensive framework considering the temporal perspective as a whole. In today's fast paced world, where even small delays can cause significant problems, it is crucial for any enterprise to know the temporal properties of its business processes [15, 41, 22, 20]. This becomes even more important in the context of long-running business processes like patient treatment or automotive engineering [20, 42].

## 1.1 Problem Statement

Both the formal specification and the operational support of temporal constraints constitute fundamental challenges for any PAIS. So far, however, time support has been rather limited in existing PAISs and there is a lack of criteria for systematically assessing and comparing the time capabilities provided by PAISs. To make PAISs better comparable and to facilitate the selection of PAIS-enabling technologies in a given context, workflow patterns have been introduced for most of the described process perspectives [3, 59, 58, 67]. Respective patterns provide a means for analyzing the expressiveness of process modeling languages and tools regarding different process perspectives. Additionally, patterns can be used for eliciting and analyzing requirements described by formal process specifications. Existing workflow patterns, for example, cover control flow [3], data flow [59], resources [58], activities [66], exceptions [57], and process change [67, 55]. However, a framework for systematically evaluating process specifications in respect to their requirements concerning the temporal perspective as well as for evaluating PAISs in respect to their ability to deal with the temporal perspective is still missing. To discuss some of the challenges being relevant in this context we first introduce a simple example:

*Example 1 (Patient Treatment Process)* Consider the simplified patient treatment process depicted in Fig. 1. First, a doctor orders a medical procedure for his patient. Then, the responsible nurse makes an appointment with the department (e.g., radiology) the procedure will take place. Before the actual treatment, the patient needs to be informed about the procedure and prepared for it. Shortly before the treatment starts, a specific preparation is required. After the treatment, the doctor writes a short report if requested. Finally, aftercare is provided and the doctor creates a final medical report, which is then added to the patient record.

When considering the temporal perspective of this rather simple process, a number of constraints can be observed:

1. The *appointment* of the **treatment**, which is made during activity **make appointment**, needs to be observed during process execution. In particular, this affects the scheduling of preceding activities as well. For example, the patient needs to be prepared *exactly 1 day before* actual **treatment** takes place. Hence, the **preparation** needs to be scheduled in accordance with the *appointment* of the **treatment**.
  2. The **preparation** of the patient may only be done during opening hours of the anesthesia department, i.e., from *Monday till Friday between 8am and 4pm*.
- This further restricts possible execution times of activity **prepare patient** and hence the ones of the **treatment** itself.
3. Due to a tight schedule of the treatment room, **preparation** of the treatment must *not take more than 1 hour*; otherwise, **treatment** is delayed.
  4. During the execution of activity **perform aftercare**, different drugs are given to the patient according to the *treatment plan* defined by activity **perform treatment**. Such a treatment plan may state, for example, that drug A has to be administered every day at 8 am, 1 pm, and 6 pm, and drug B every two hours except when drug A is administered within the same hour.
  5. Activity **create report** needs to be *completed no later than 1 week after* activity **perform treatment** is completed.
- To support the implementation of such a process and its temporal constraints, a variety of temporal concepts (e.g., deadlines, minimum time lags, maximum durations, periodicity) need to be supported by the PAIS. Hence, respective concepts should be expressible with the process specification language used. Yet, there has been no reference system for systematically comparing PAISs regarding their support of temporal constraints. This challenge is picked up by this paper.

## 1.2 Contribution

The contributions of this paper are as follows:

1. We suggest 10 *time patterns* to foster the comparison of existing PAISs with respect to their ability to deal with temporal aspects (cf. Section 4). The proposed time patterns complement existing workflow patterns. They were systematically identified by analyzing a large collection of process models in healthcare, automotive engineering, aviation industry, and other domains (cf. Section 3).
2. We conduct a *systematic literature review* to evaluate the completeness and validity of the 10 time patterns. To ensure completeness, we do not only consider PAIS-specific sources, but also literature from other research areas (cf. Section 5).
3. We provide an *in-depth evaluation* of selected approaches from both industry and academia based on the proposed time patterns (cf. Section 6). The evaluation does not only consider process management systems, but also calendar systems and project planning tools, in which temporal constraints play an important role.

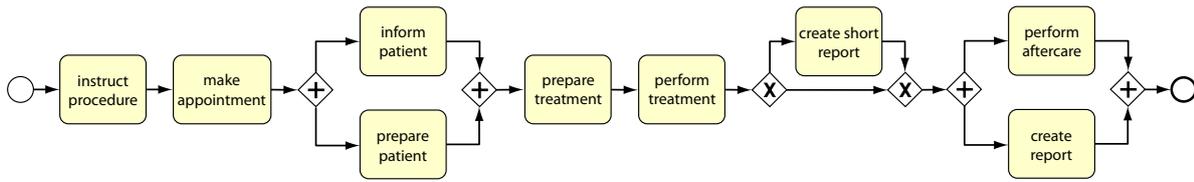


Fig. 1 Simplified Treatment Process (in BPMN Notation)

The pattern-based analysis shows that these different technologies (i.e., process management systems, calendar systems, and project planning tools) all support temporal constraints. Moreover, our work makes evident that the integration of a PAIS with the described time patterns offers promising perspectives.

This paper significantly extends the work we presented in [31]. While [31] only described selected patterns rather briefly, this paper provides an in-depth description of all time patterns we identified, provides empirical evidence for them, and discusses each pattern in detail. Additionally, we include the results of a systematic and thorough literature review to support our findings. Finally, we provide an in-depth evaluation of selected approaches from both industry and academia based on the proposed time patterns. Similar to the workflow patterns initiative [3], we expect further systems to be evaluated over time and software vendors to extend their PAISs by a more complete support of the temporal perspective. To foster this, we also incorporate a discussion of other aspects to be taken into account when implementing the proposed time patterns in a PAIS.

The remainder of this paper is organized as follows: Section 2 summarizes basic notions and Section 3 presents the research method we employed for identifying the time patterns. Section 4 describes 10 time patterns, which are sub-divided into 4 categories. In Section 5, we discuss the findings of the systematic literature review we conducted. Section 6 then summarizes the results we obtained when evaluating academic approaches and tools. Section 7 discusses further aspects relevant for implementing the proposed time patterns in a PAIS. We conclude with a summary and outlook in Section 8.

## 2 Basic Notions

This section summarizes basic concepts and notions needed for understanding this paper. A *process-aware information system* (PAIS) is a specific type of information system providing process support functions [52]. It is characterized by the separation of process logic from application code. Usually, at build-time the process logic

must be explicitly defined based on the constructs provided by a *process meta model*. An overview of such a process meta model is given in Fig. 2 in terms of an ontology describing basic constructs. At run-time, a PAIS executes the processes according to the defined logic. Further, it enables the integration of users and other resources.

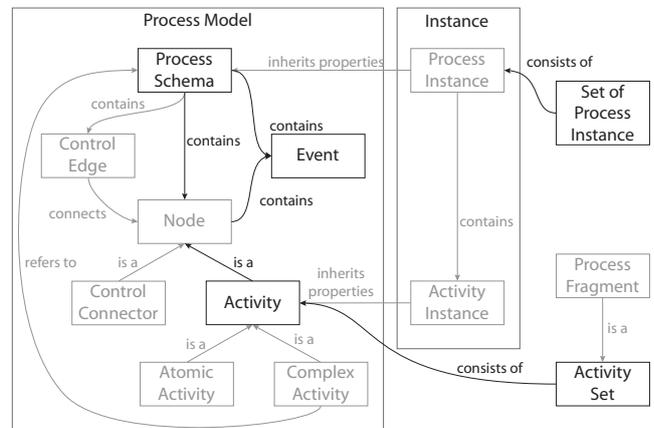


Fig. 2 Basic Constructs of a Process Meta Model

For each business process to be supported, a *process type* represented by a *process schema* has to be defined (cf. Fig. 2 and Fig. 3). In this paper, a process schema corresponds to a directed graph, which comprises a set of *nodes* – representing *activities* and *control connectors* (e.g., Start-/End-Nodes, XOR-Splits, or AND-Joins) – and a set of *control edges* between them. The latter specify precedence relations between nodes. Furthermore, activities can be either *atomic* or *complex*. While an *atomic activity* is associated with an application service, a complex activity refers to a *sub-process* or, more precisely, a *sub-process schema*. This enables modularity through the hierarchical decomposition of process schemes.

Furthermore, this paper uses the notion of *activity set* to refer to a subset of the activities of a process schema. The elements of an activity set do not have to comply with any structural requirement. When referring to specific regions of a process schema, in turn, we use the notion of *process fragment*. To be more precise, a *process fragment* refers to a sub-graph of a process

schema with single entry and single exit node (also denoted as single-entry, single-exit region; i.e., *SESE-region*).

Fig. 3 shows a process schema consisting of 7 activities and 4 control connectors: Activity A is succeeded by the parallel execution of either activity B or C, activities D and E, and activities F and G. Activities A to F are atomic, whereas G constitutes a complex activity; i.e., a sub-process with its own process schema. The region of the process schema containing activities B and C together with the depicted control connectors (i.e., the XOR-Split and XOR-Join) constitutes an example of a SESE-region. Finally, any non-empty subset of activities A . . . G constitutes an activity set.

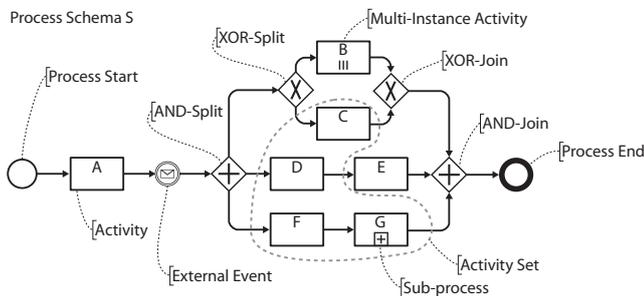


Fig. 3 Core Concepts of a Process Meta Model

At run-time, *process instances* are created and executed according to a predefined process schema. In turn, *activity instances* represent executions of single process steps (i.e., activities) of a particular process instance. Finally, this paper uses the notion of *process instance set* to refer to a set of process instances executed by the PAIS. These process instances, in turn, may either be enacted based on the same process schema, but may also run on different process schemas.

During the execution of a process instance, *events* may be triggered, either by the process instance itself (e.g., start/end event of the process instance), by a node belonging to the process (e.g., start/end event of an activity instance, cf. Fig. 4), or by an external source (e.g., receipt of a message). We use the notion of event as general term for something happening during process execution.

We assume that activity instances are executed according to the *activity life cycle* depicted in Fig. 4. When a process instance is started, its activities are in state *Not Activated*. As soon as an activity may be executed, its state switches to *Activated*. When a user starts the activity instance, its state switches to *Started* and a *Start Event* is generated. As soon as the user finishes work, the state of the activity instance switches to *Completed* and an *End Event* is generated. Finally, non-executed activi-

ties (e.g., activities of an outgoing path of an XOR-Split not selected during run-time) are in state *Skipped*.

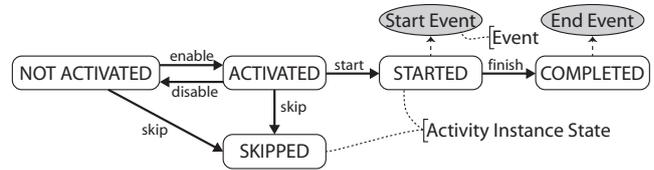


Fig. 4 Activity Life Cycle and Relevant Events

Even though we use selected elements defined by BPMN (Business Process Model and Notation) [45] for illustration purpose (cf. Fig. 3), the described time patterns are not language-specific, i.e., they can be integrated in any process modeling language supporting the aforementioned basic concepts. We use additional symbols that are not part of BPMN to illustrate our time patterns.

### 3 Research Method

This paper complements existing workflow patterns with a set of time patterns. Our goal is to provide a framework for guiding future extensions of PAISs regarding the temporal perspective of business processes. In this section, we describe the *selection criteria* of our time patterns, the *data sources* they are based on, and the *procedure* applied for identifying the time patterns.

#### 3.1 Selection Criteria

We consider patterns covering temporal aspects relevant for the modeling and control of business processes and activities respectively. We target a high coverage of real-world scenarios, i.e., we focus on the expressiveness required from a PAIS to properly model temporal aspects encountered in real-world processes. Specifically, we introduce time patterns affecting the control perspective. The extension towards other process aspects (e.g., data flow or process change) constitutes complementary work and is outside the scope of this work. Furthermore, this paper focuses on expressiveness rather than on specific time features of a PAIS. The latter deal with the consistency of temporal constraints and their verification [41, 12, 15, 22], escalation management [5], or scheduling support [18, 23, 10]. Finally, temporal aspects in conjunction with process monitoring [62], process analysis [25], and process mining [6, 33, 4] are also outside the scope of this paper.

A) Healthcare Domain (Women Hospital)	
Administrative Processes	8 processes
In-Patient Chemotherapy	13 processes
Ovarian Cancer Surgery	21 processes
Ambulatory Chemotherapy	8 processes
Endoscopic Surgery in a Day Hospital	21 processes
Laboratory Test	8 processes
Radiologic Test	9 processes
Total: 88 processes	
B) Healthcare Domain (Internal Medicine)	
Medical order handling and result reporting (e.g. radiology, cardiology, gastroenterology, and clinical lab)	84 processes
Patient admission / transfer / discharge	4 processes
Pharmacy Management	2 processes
Medication	2 processes
Consile Handling	1 process
Total: 93 processes	
C) Automotive Domain	
Engineering Change Request	11 processes
Engineering Change Management	16 processes
Car Repair and Maintenance in Garages	20 processes
In-house Change Management	1 process
Product Planning	1 process
Road vehicles – Functional safety (ISO 26262)	5 processes
Release Management for Electric/Electronic Components	1 process
Total: 55 processes	
D) Other domains	
On-demand Air Service	2 processes
Airline Catering Services	1 process
Transportation	9 processes
Software Engineering	10 processes
Event Marketing	1 process
Financial Service	10 processes
Municipality Processes	4 processes
Total: 37 processes	

**Table 1** Data Sources

### 3.2 Data Sources and Data Collection

As sources of the time patterns, we consider results from several case studies. These were performed in different domains, including healthcare, automotive engineering, and aviation industry (cf. Table 1).

Our first major data source stems from a large *healthcare* project in which we analyzed the core processes of a Women Hospital [63,64,49] and implemented selected processes using existing workflow technology. Furthermore, time aspects were explicitly elicited and documented. In total, this data source comprises 88 process models covering both administrative processes (e.g., order handling) and treatment processes (e.g., chemotherapy and ovarian cancer surgery) (cf. Table 1 A).

Our second major data source comprises *healthcare processes* from a large Medical University Hospital. This data source contains 93 different processes, related to diagnostic and therapeutic procedures, i.e., examinations in medical units like radiology, gastroenterology, and clinical chemistry [27,28,34,32,35] (cf. Table 1 B).

As third data source we use process models from the *automotive industry*. We consider engineering change management (ECM) [24] as well as the process models described in [13]. Some of the models related to ECM have been published by the German Association of the Automotive Industry (VDA) [24]. The process models described in [13], in turn, refer to car repair and maintenance in garages, in-house change management, and product development. With hundreds of activities the product development process is the most complex one we consider. In total, this case provides us with 55 process models (cf. Table 1 C for an overview).

As fourth data source we consider processes stemming from several projects we conducted in various other domains (cf. Table 1 D). Due to copyright & secrecy restrictions not all of these processes can be published. Respective domains include on-demand air service [?,?], airline catering services, transportation [?], software engineering [?,?], event marketing, financial services, and municipality processes [?].

### 3.3 Pattern Identification Procedure

Following the approach taken in [67] and to ground the time patterns on a solid basis, we first create a list of candidate patterns. For this purpose, we make use of the experience we gathered when applying PAIS-enabling technologies and our knowledge in this field of research. Next, we thoroughly analyze the aforementioned cases and process models respectively in order to find empirical evidence for our time patterns and -if necessary- to extend the pattern candidate list. As a pattern is defined as reusable solution to a commonly occurring problem [7], we require each time pattern to be observed at least *three times* in different domains of our samples. Therefore, only those patterns, for which enough empirical evidence exists, are included in the final list of patterns. The analysis of respective cases and process models further led to a refinement of our initial set of patterns (e.g., additional design choices were added), and to the insight that in the context of recurrent process element our initial pattern had to be split into two more specific ones (i.e., TP9 and TP10).

Analyzing the process models in respect to the use of the time patterns is a rather complex task. First, while some process specifications contain many details about temporal aspects, others mention time constraints only as a side note. Second, temporal aspects are often described implicitly as text added to the process specification. Further, temporal aspects may also depend on the way a process is implemented. This often makes it hard to clearly identify and classify temporal aspects. During our pattern identification phase, we avoid this

problem by only considering those pattern occurrences, which can be both clearly identified and classified. To illustrate some of the challenges we face when analyzing the process models and their textual descriptions, consider the following excerpts from [64]:<sup>1</sup>

“One day prior to the examination, the physician must make an appointment with the department where the examination takes place. At the scheduled time, the patient visits the respective department to be examined.” [64, p. 30]

Having a closer look at this textual description one can identify two temporal constraints. First, an appointment for the examination has to be made and later be met. Second, there is a fixed time lag between the activity making the appointment and the examination itself, which has to be obeyed as well.

“It must be ensured that there is a time lag of at least six days between a lower gastrointestinal series and an upper gastrointestinal X-ray series.” [64, p. 47]

At first glance, this excerpt seems to describe a time lag between the two activities. However, taking the given context into account, it becomes clear that no fixed order exists between the two activities. Therefore, this example refers to some sort of time-based mutual exclusion.

“During the diagnostic and therapeutic treatment of a patient, several examination cycles are run in parallel. However, these cycles cannot be fully treated independent from one another. Temporal dependencies between examinations as well indicated sequences between them must be considered.” [64, p. 35]

This excerpt indicates the complexity of the temporal perspective of a process. When analyzing this description in detail, one can see that it shows an example of a set of recurrent activities. Thereby, several time constraints between different activities must be taken into account (cf. TP10).

#### 4 Time Patterns

During our analyses we identified 10 different time patterns. We divide these into 4 distinct categories based on their semantics (cf. Fig. 5). Particularly, the time patterns constitute solutions for representing commonly occurring temporal constraints in PAISs. *Pattern Category I (Durations and Time Lags)* provides support for expressing *durations* of different process granularities (i.e., activities, activity sets, processes, or sets of process instances) as well as *time lags* between activities or – more generally – between process events (e.g. milestones). *Pattern Category II (Restricting Execution Times)* allows

specifying constraints regarding possible execution times of single activities or entire processes (e.g., activity deadlines). *Category III (Variability)* provides support for expressing *time-based variability* during process execution (e.g., varying control-flow depending on temporal aspects). Finally, *Category IV (Recurrent Process Elements)* comprises patterns for expressing temporal constraints in connection with recurrent activities or process fragments (e.g., cyclic flows and periodicity).

Pattern Catalogue
<b>Category I: Durations and Time Lags</b>
TP1: Time Lags between two Activities
TP2: Durations
TP3: Time Lags between Arbitrary Events
<b>Category II: Restricting Execution Times</b>
TP4: Fixed Date Elements
TP5: Schedule Restricted Elements
TP6: Time-based Restrictions
TP7: Validity Period
<b>Category III: Variability</b>
TP8: Time-dependent Variability
<b>Category IV: Recurrent Process Elements</b>
TP9: Cyclic Elements
TP10: Periodicity

Fig. 5 Pattern Catalogue (TP = Time Pattern)

Fig. 5 gives an overview of the identified time patterns. For each pattern, we provide a name, synonyms, a description of the problem addressed, design choices characterizing pattern variants, remarks regarding pattern implementation, minimal requirements in respect to the pattern’s application context, use cases we identified when analyzing practical scenarios, and a reference to related patterns (cf. Fig. 10 - Fig. 25). Thereby, two patterns are considered to be mutually related if they have any common element.

Following the approach described in [67], we use *design choices* for parameterizing the time patterns. This enables us to keep the number of different patterns manageable. For example, there exist four different variants of pattern TP1 (*Time Lags between Activities*) as time lags between activities can be specified either between the start of two activities, between the start of the first and the end of the second activity, between the end of the first and the start of the second activity, or between the end of the two activities. Additional variance is introduced due to the different semantics of time lags, e.g., the need to distinguish between minimum and maximum time lags (cf. Fig. 10).

Design choices not only relevant for a particular pattern, but for a whole pattern group, are described only once for this group. Typically, an existing PAIS does not support all design choices regarding a specific

<sup>1</sup> translated from German

pattern. In this context, we denote the combination of design choices supported by a particular approach as *pattern variant*. Fig. 6 depicts two system-specific design choices relevant for most of the time patterns: *Design Choice A* describes when the parameters (e.g., duration) of the respective patterns can be set, i.e., at build-time, process instantiation-time, or run-time. *Design Choice B* describes which time granularities are supported by the PAIS. This includes basic time granularities (e.g., second, minute, hour, etc.) as well as system- and user-defined time granularities (e.g., business days and personal working hours). Since design choices *A* and *B* are system-specific, we do not repeat them in the tables describing the specific patterns. However, if not all options of design choices *A* or *B* are valid for a given time pattern, this is commented.

System-specific Design Choices
A) Parameters of a pattern may be set at <ul style="list-style-type: none"> <li>a) build-time (i.e., during process modeling)</li> <li>b) process instantiation-time (i.e., when a process instance is instantiated)</li> <li>c) run-time (i.e., during process execution)</li> </ul>
B) Time parameters can be specified in different time granularities <ul style="list-style-type: none"> <li>a) Basic (i.e., years, months, weeks, days, hours, minutes, seconds)</li> <li>b) System-defined (e.g., business days)</li> <li>c) User-defined (e.g., Wednesday afternoon)</li> </ul>

Fig. 6 System-specific Design Choice

*Design Choice C*, which is depicted in Fig. 7, is a general design choice being valid for several patterns. It describes the process granularities (i.e., activity, activity set, process, set of process instances) to which the respective pattern can be applied (e.g., to constrain the duration of a single activity or of all instances created from a particular process schema). In this context, a *process* refers to a process schema as well as to corresponding process instances. In general, a pattern is applied to the respective process schema, whereas its parameters may be specific for the particular process instance (i.e., each process instance has its own value). For example, consider an appointment for an activity. While the existence of the appointment is known at process schema level, its date and time is determined during the execution of the respective process instance. The instances belonging to a *set of process instances*, in turn, may be enacted based on the same process schema, but may also run on different process schemes. In the latter case, these process instances share common characteristics linking them with each other (e.g., process instances referring to different medical examinations of

the same patient). Though we do not explicitly consider sub-processes in the following discussions, all patterns that may be applied to a process, can be applied to sub-processes as well. For each pattern, the process granularities to which the pattern may be applied are explicitly listed.

General Design Choice
C) Patterns can be applied to different process granularities <ul style="list-style-type: none"> <li>a) Single activity (including multi-instance activities)</li> <li>b) Activity set</li> <li>c) Process model</li> <li>d) Set of process instances</li> </ul>

Fig. 7 General Design Choice

Finally, additional design choices only relevant for a specific pattern (category) are introduced in connection with this pattern (category).

#### 4.1 Pattern Category I: Durations and Time Lags

The first pattern category we consider comprises three time patterns expressing durations for different kinds of process granularities (e.g., activities) as well as time lags between activities or events. These patterns not only enable us to restrict processes and their activities in terms of minimum/maximum durations, but also describe general temporal properties of a process (e.g., a *critical path*<sup>2</sup>). Knowledge about temporal properties is especially relevant for scheduling activities and for allocating resources (e.g., expressing how long a particular resource may be used by a certain activity or process). Further, it can be used for predicting execution times of not yet enabled activities (e.g., “activity B will be enabled tomorrow morning”) [46,23].

Time lags and durations can be specified in terms of minimum/maximum values as well as time intervals (cf. Fig. 8, *Design Choice D*). The latter represent both a minimum and a maximum value at the same time; e.g., the constraint “between 2 and 3 hours” can be expressed using 2 hrs as minimum and 3 hrs as maximum. Hence, *Design Choice D* constitutes a general design choice being valid for all patterns from Category I (cf. Fig. 8).

##### 4.1.1 Pattern TP1 (Time Lags between two Activities)

This pattern allows defining time lags between two activities; i.e., to express a minimum or maximum temporal distance between them. For example, pattern TP1 may

<sup>2</sup> A critical path is a sequence of activities within a process schema, which add up to the longest overall process duration. Its length determines the shortest time to complete the process.

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

Fig. 8 General Design Choice for Category I

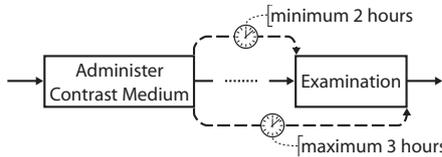


Fig. 9 Minimum and Maximum Time Lags between Activities

be applied to ensure compliance with business rules and global regulations.

*Example 2 (Minimum and maximum time lag between two activities)* Consider a medical examination conducted in the radiology department and assume that a contrast medium must be given to the patient prior to the examination (cf. Fig. 9). On the one hand, in order to ensure that the contrast medium is properly distributed in the patient's body, this activity has to be accomplished *at least 2 hours* before the radiological examination takes place. On the other hand, the contrast medium must not be administered *more than 3 hours* ahead of the examination; otherwise it might have been already decomposed in the body and thus the examination would not lead to any meaningful result.

In Example 2, there exists a *minimum time lag* of 2 hours between delivery of the contrast medium and start of the medical examination. Furthermore, there is a *maximum time lag* of 3 hours between delivery of the drug and completion of the examination (cf. Fig. 9). Such time lags can be expressed using time pattern TP1. Details about this pattern are given in Fig. 10. It can only be applied to activities (*Design Choice C[a]*). Additional pattern variants can be expressed using design choices D and E. *Design Choice D* (cf. Fig. 8) allows choosing among maximum time lag, minimum time lag, and time interval (cf. Example 2). In turn, *Design Choice E* (cf. Fig. 10) specifies whether a *Time Lag* represents a start-start relation (i.e., referring to the start event of two different activities), a start-end relation, an end-start relation, or an end-end relation. Reconsider Example 2, for which the minimum time lag between the two activities corresponds to an end-start relation and the maximum time lag to an end-end relation.

In general, *Time Lags* cannot be only expressed between two directly succeeding activities, but also between arbitrary activities (e.g., contained in different regions of a sequence or belonging to different paral-

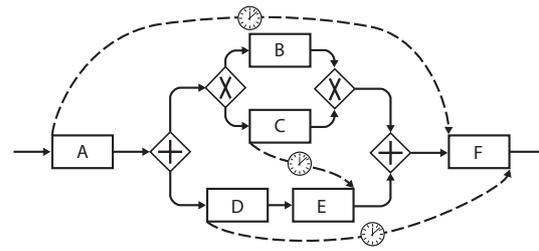


Fig. 11 Time Lags between Arbitrary Activities

lel branches), presuming that these may belong to the same process instance. In particular, a *Time Lag* is not allowed between two alternative activities. For example, consider the process model from Fig. 11. Here, *Time Lags* are specified between activities A and F, activities D and F, and activities C and E. However, no *Time Lag* must be specified between B and C since these two activities will never be executed together for the same process instance.

#### 4.1.2 Pattern TP2 (Durations)

*Durations* constitute one of the most frequently used temporal constraint in PAISs. Usually, they are required to ensure compliance of processes with global rules and regulations, to guarantee adherence to external benchmarks (e.g., policies, service level agreements), or to limit resource usage.

*Example 3 (Maximum duration)* Due to service level agreements between an on-line book store and its customers, the handling of an order must not take longer than 1 day (express shipping). Otherwise, the customer is entitled to get a partial refunding (exception handling).

*Minimum durations* are frequently required for planning purpose, e.g., to determine how long a resource will be at least occupied or to identify the earliest possible point in time for scheduling succeeding activities.

*Example 4 (Minimum duration)* Cleaning an operating room after a surgery takes at least 30 minutes. Therefore, a surgery must not be scheduled earlier than 30 minutes after completing the previous one.

Time pattern TP2 is described by Fig. 12. It allows specifying the duration of an element of any process granularity (i.e., activity, activity set, process, or set of process instances) (cf. *Design Choice C*, Fig. 12). In addition to *Design Choice C*, pattern TP2 supports *Design Choice D* (cf. Fig. 12).

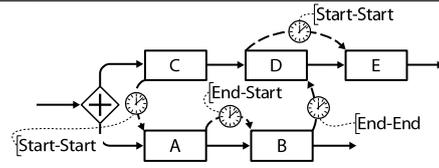
Time Pattern TP1: Time Lags between two Activities	
<b>Also known as</b>	Upper and Lower Bound Constraints, Inter-Task Constraints, Temporal Relations
<b>Problem</b>	There is a time lag between two activities to be obeyed. Such Time Lags may not only be defined between directly succeeding activities, but also between arbitrary ones (presuming that the activities may belong to the same process instance).
<b>Design Choices</b>	C) Time lags only constrain the execution of activities C[a] (cf. Fig. 7) D) Time lags may represent all three kinds of restrictions (cf. Fig. 8) E) Time lags define the distance between the <ol style="list-style-type: none"> <li>start of two activities (i.e., Start-Start relation)</li> <li>start of the first and the completion of the second activity (i.e., Start-End)</li> <li>completion of the first and the start of the second activity (i.e., End-Start)</li> <li>completion of two activities (i.e., End-End)</li> </ol>
<b>Solution</b>	A time constraint is introduced between the start / end event of the two activities.  <p>Timers can be used to realize this pattern at run-time. For example, to implement an end-start relation between activities A and B, the timer starts after completing A. If the time lag between A and B represents a minimal one, B must not be started before the timer has expired; i.e., the execution of activity B is delayed until the time lag is satisfied. If the time lag expresses a maximum distance, B may be started immediately, but has to be started latest when the timer expires. In case the timer expires an exception handling may be triggered. For time intervals both cases apply.</p>
<b>Context</b>	The mechanism evaluating the constraint (i.e., starting the timer) needs to be able to access the value of the time lag when starting the timer.
<b>Examples</b>	<ul style="list-style-type: none"> <li>The <i>maximum time lag</i> between discharging a patient from a hospital and sending out the discharge letter to the general practitioner treating the patient is 2 weeks (Design Choices D[b] E[d])</li> <li>Patients must not eat <i>at least 12 hours</i> before a surgery takes place. The latest point in time at which the patient can have a meal is therefore determined by the date of the surgery (Design Choices D[a] E[c])</li> <li>The time lag between registering a Master thesis and submitting it must not exceed 6 months (Design Choices D[b] E[a])</li> </ul>
<b>Related Patterns</b>	TP2 (Durations): TP1 and TP2 both refer to the time lag between process events. TP3 (Time Lags between Events): TP1 can be implemented based on pattern TP3. TP9 (Cyclic Elements): TP9 is an extension of TP1, considering cycles and iterations.

Fig. 10 TP1 - Time Lags between Activities

#### 4.1.3 Pattern TP3 (Time Lags between Arbitrary Events)

Certain events cannot be bound to the start or end of an activity or process. For example, there may be events triggered by external sources not controllable by the PAIS. e.g., receiving a message from a partner process or an event in the physical world [70]. In addition, there may be events not bound to a specific activity, e.g., event “delivery of all parts” requires several activities and processes, respectively, to be completed. Finally, there may be events triggered inside a long-running activity, e.g., when reaching a milestone during the execution of a long-running activity or sub-process. Time pattern TP3 enables the specification of *Time Lags* between two arbitrary discrete events.

*Example 5 (Minimum time lag between two message events)* In product development, usually, well-defined

processes for realizing product changes and for getting the approval required from the concerned partners exist. For a change management process (cf. Fig. 13), typically, a maximum time lag between sending a *request for approval* (for a product change) to a supplier (event) and getting a corresponding response (event) can be found.

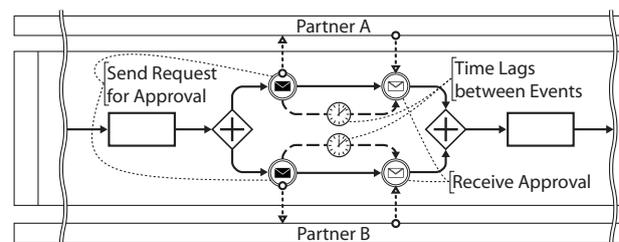


Fig. 13 Time Lag between two Message Events

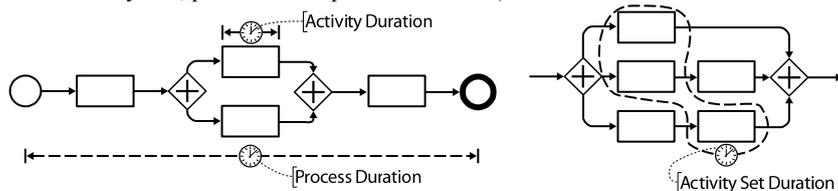
Time Pattern TP2: Durations	
Also known as	-
Problem	A particular activity, activity set, process, or set of process instances has to obey a certain duration restriction. <i>Durations</i> result from both waiting (i.e., activity is suspended) and processing times during activity execution. However, a duration does not cover the time span the activity (activity set, process or process instance) is offered in user worklists but has not been started by any user yet (i.e., the time span between activation and start of the activity).
Design Choices	C) Durations may be applied to all four kinds of process granularities (cf. Fig. 7) D) Durations may represent all three kinds of restrictions (cf. Fig. 8)
Solution	A time constraint is introduced between the start and end events of an activity (the same applies to activity sets, processes and process instances).  Timers can be used to enable run-time support for durations. For minimum (maximum) durations the respective activity must not complete before (after) the timer has expired, otherwise appropriate exception handling needs to be initiated. For time intervals, the end event has to occur within the time 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 starting the activity (activity set, process or process instance).
Examples	<ul style="list-style-type: none"> <li>• The assembly of a new engine must <i>not take longer than 30 minutes</i> (task work) (Design Choices C[b] D[b])</li> <li>• Depending on its severity and the patient's state, ovarian cancer surgeries <i>take 1 to 10 hours</i> (Design Choices C[a] D[c]).</li> <li>• Maintenance issues need to be resolved <i>within 1hr</i> (Design Choices C[c] D[b])</li> <li>• Processing 100 requests must <i>not take longer than 1 second</i> (Design Choices C[d] D[b])</li> </ul>
Related Patterns	TP1 (Time Lags between Activities): TP1 and TP2 both refer to the time lag between process events. TP3 (Time Lags between Events): TP2 can be implemented based on TP3.

Fig. 12 TP2 - Durations

Pattern TP3 is described by Fig. 14. As opposed to pattern TP1, which only supports time lags between activities, TP3 provides more generic support for expressing arbitrary time lags in PAISs. In general, time patterns TP1 and TP2 can be expressed using pattern TP3. However, since their semantics and use cases are quite different (e.g., regarding escalation handling or the enforcement of compliance with a temporal constraint in the PAIS), we added all three patterns to our pattern catalogue. Similarly to time patterns TP1 and TP2, pattern TP3 supports *Design Choice D* (cf. Fig. 14) in order to specify whether a time lag expresses a minimum value, a maximum value, or a time interval.

#### 4.2 Pattern Category II: Restricting Execution Times

This category comprises four patterns for restricting the execution times of an activity or process (e.g., earliest start or latest completion time). Time patterns from this

category enable a PAIS to properly time the execution of activities and process instances (cf. TP4: Fixed Date Element), to bind the execution of an activity or process to an external schedule (cf. TP5: Schedule Restricted Element), to limit the number of executions of an activity (process) within a particular time frame (cf. TP6: Time-based Restrictions), or to restrict the lifetime of an activity or process (cf. TP7: Validity Period).

Regarding time patterns from Category II, *Design Choice F* describes what kind of date is specified by the respective constraint, i.e., we differentiate between earliest start, latest start, earliest completion, and latest completion date (cf. Fig. 15). As example consider the submission deadline of a Master thesis. Enforcing a certain completion date, for example, would be too strict as it is also possible to submit the thesis prior to the submission deadline. Therefore, it should be possible to not only restrict the start or completion date of an activity, but alternatively to only restrict the earliest

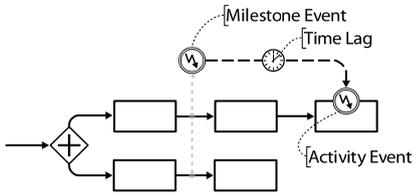
Time Pattern TP3: Time Lags between Arbitrary Events	
Also known as	-
Problem	A time lag between two discrete events needs to be obeyed. Respective events occur, for example, when instantiating or completing a process instance, when reaching a milestone in a process instance, or when triggering specific events inside an activity.
Design Choices	D) Time lags between events may represent all three kinds of restrictions (cf. Fig. 8)
Solution	<p>A time constraint is introduced between two events.</p> <p>Again timers can be used to realize this pattern at run-time (cf. TP1 in Fig. 10). Additionally, an observer, monitoring external events and notifying the time management component, is required.</p> 
Context	The mechanism evaluating the constraint (i.e., starting the timer) needs to be able to access the value of the time lag prior to its activation (i.e., the occurrence of the start event) in order to start the respective timer.
Examples	<ul style="list-style-type: none"> <li>• The time lag between the delivery of all parts (milestone) and the assembly of the car's chassis (milestone) should be <i>no more than 2 hours</i> (e.g. just-in-time production) (Design Choices D[b]).</li> <li>• The time lag between two heavy maintenance visits of an aircraft is <i>4-5 years</i> (Design Choice D[c])</li> </ul>
Related Patterns	<p>TP1 (Time Lags between Activities): TP3 is a generalization of TP1.</p> <p>TP2 (Durations): TP3 is a generalization of TP2.</p>

Fig. 14 TP3 - Time Lags between Events

start, latest start, earliest completion, or latest completion date. In the given case, for example, only the latest completion date of the respective activity should be restricted.

General Design Choice for Pattern Category II
F) Patterns can restrict four dates of an activity (process)
a) Earliest start date,
b) Latest start date,
c) Earliest completion date,
d) Latest completion date

Fig. 15 General Design Choice for Category II

#### 4.2.1 Pattern TP4 (Fixed Date Element)

Besides durations, *TP4: Fixed Date Elements* constitutes another frequently applied pattern for specifying temporal constraints (according to our analysis). In particular, *TP4: Fixed Date Elements* allows specifying that a particular activity or process instance must be executed at a fixed date.

*Example 6 (Fixed date element)* A patient has an appointment for a medical examination next Monday at 10 am. Due to a tight schedule of the physician, it may well be that after his arrival the patient has to wait some time before the examination can start (i.e., the earliest possible start date is given).

Time pattern TP4 allows expressing such deadline constraints. It is described in Fig. 16. *Fixed Date Elements* can be applied to an activity or process (*Design Choice C[a, c]*, cf. Fig. 16). Parameter values of a Fixed Date Element, however, are specific to a process instance, i.e., they are not known before creating a process instance. Therefore, it is not possible to set all parameter values of the respective pattern already at build-time, i.e., *Design Choice A[a]* (cf. Fig. 16) is not applicable to pattern TP4. For a particular activity or process instance, it can be defined whether it has to be started after, started before, completed after, or completed before a particular date (*Design Choice F*, cf. Fig. 16).

*Fixed Date Elements* may have a great influence on the temporal properties of a process (e.g., length of the critical path, scheduling of other activities) as they fix execution time of activities. Thus, they implicitly restrict the latest (earliest) start (completion) time of preceding (succeeding) activities as well.

*Example 7 (Deadline)* The flight from Munich to Amsterdam leaves at 6:05 am. All process activities prior to the flight need to be completed by this time.

#### 4.2.2 Pattern TP5 (Schedule Restricted Element)

*TP5: Schedule Restricted Elements* allows restricting the execution of a particular activity or process through a

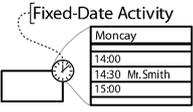
Time Pattern TP4: Fixed Date Elements	
<b>Also known as</b>	Deadline
<b>Problem</b>	A particular activity or process instance has to be executed in relation to a particular date.
<b>Design Choices</b>	A) Parameters of this pattern may only be set during instantiation-time A[b] or run-time A[c] (cf. Fig. 6). C) A fixed date can be applied to an activity C[a] or a process C[c] (cf. Fig. 7) F) A fixed date can restrict all four types of dates (cf. Fig. 15)
<b>Solution</b>	A fixed date is attached to the respective activity or process.  Fixed dates can be realized using a timer which starts as soon as the value of the fixed date is known and which expires at the respective date. For example, if for a latest start date the respective activity (process) has not been started before the timer has expired, appropriate exception handling routines may be initiated. Other restrictions can be handled analogously (cf. Fig. 10 for an example). 
<b>Context</b>	The value of the fixed date needs to be available before the respective activity or process is enabled.
<b>Examples</b>	<ul style="list-style-type: none"> <li>• During a chemotherapy cycle the physician has to inform the pharmacy about the dosage of the cytostatic drug <i>until 11 am</i>. If the deadline is missed the pharmacy checks back by phone for the exact dosage (escalation mechanism) (Design Choices C[a] F[d]).</li> <li>• For each paper submitted to a scientific conference three review requests are sent to members of the program committee. Reviews for all submitted papers have to be entered into the submission system <i>by a particular deadline</i> (Design Choices C[a] F[d]).</li> </ul>
<b>Related Patterns</b>	TP5 (Schedule Restricted Elements): Like TP4, TP5 constrains possible execution times.

Fig. 16 TP4 - Fixed Date Elements

schedule; i.e., a timetable (e.g., a bus schedule). Typically, the structure of the schedule is known at built-time, whereas the concrete dates are determined either at creation or run-time of a process instance, e.g., when a particular activity is scheduled. The schedule provides restrictions on when the respective activity or process may be executed.

*Example 8 (Schedule restricted element)* Opening hours of the dermatological clinic are MO - FR, 7 am - 6 pm except for public holidays. Dermatological examinations can only be scheduled within this time frame.

Time pattern TP5 allows expressing constraints of this kind (cf. Fig. 17). In detail, it allows restricting the possible execution times of an activity or process through a *schedule*. A schedule itself consists of a (possibly infinite) set of *time slots* described by a finite expression based on a calendar (e.g., “MO - FR, 8 am - 5 pm”) (see [?,?]). Each time slot, in turn, specifies a *time frame* during which the respective event of the activity may occur.

*Example 9 (Schedule and time slots)* Consider the schedule presented in Example 8. It is described by expression “MO - FR, 8 am - 5 pm except for public holidays”. For example, this schedule contains the time slots “Monday, February 14. 2011, 8 am till 5 pm”, “Tuesday, February 15. 2011, 8 am till 5 pm”, and so forth.

As listed in Fig. 17, a *Schedule Restricted Element* can be applied to an activity and process, respectively (*Design Choice C[a, c]*). Further, it can be used to restrict the earliest start, latest start, earliest completion, and latest completion of the respective activity or process (*Design Choice F*). More precisely, as a schedule consists of several time slots, it always restricts both the earliest and latest start or the earliest and latest completion date of the respective activity or process. Additionally, exceptions related to a given schedule can be explicitly defined (e.g., “every Monday except for public holidays”).

For highly restricted schedules, even small delays during process execution may have significant consequences; e.g., if a particular time slot of the schedule is no longer valid, i.e., the activity may no longer be executed within the respective time slot. This will be especially critical if schedule restricted elements being on a *critical path* are affected by the delay or the concerned path itself becomes a critical path due to this delay.

*Example 10 (Effects caused by a delay)* The local weekly newspaper has as a submission deadline for advertisements every Thursday at 11 am. If this deadline is missed, there will be a delay of 1 week for publishing the respective advertisement causing delays for succeeding activities as well.

<b>Time Pattern TP5: Schedule Restricted Elements</b>	
<b>Also known as</b>	-
<b>Problem</b>	The execution of a particular activity or process is restricted by a schedule. This schedule provides restrictions on when the respective element may be executed, i.e., the schedule defines time slots in which the respective activity may be started or completed. Usually, the structure of the schedule is known at build-time, while the concrete date is set during runtime (i.e., at the process instance level). Schedules may contain exceptions (e.g., every year except leap years).
<b>Design Choices</b>	C) A schedule can be applied to an activity C[a] or process C[c] (cf. Fig. 7) F) A schedule can restrict all four types of dates (cf. Fig. 15)
<b>Solution</b>	A schedule is attached to the respective activity or process.  A schedule restriction can be realized using a timer which is started at process instantiation time and expires at the first endpoint of one of the respective time slots (i.e., when entering or leaving a valid time frame of the particular schedule). The timer is then reset and its expiration date is set to the next endpoint of one of the time slots. This is repeated until the respective activity (process) has been started / completed (cf. Design Choice F) or no more valid time slots are available according to the schedule. Outside of a valid time slot the start of the respective activity (process) should be prevented by the system. If the completion of the respective activity (process) does not occur within a valid time slot or there is no longer a valid time slot available according to the schedule, exception handling is required.
<b>Context</b>	The structure of the schedule needs to be known before the activity or process becomes available for execution.
<b>Examples</b>	<ul style="list-style-type: none"> <li>From Munich to Amsterdam there are flights at 6:05 am, 10:30 am, 12:25 pm, 5:35 pm and 8:40 pm (Design Choice C[a]).</li> <li>Comprehensive lab tests in a hospital can only be done from MO – FR between 8 am and 5 pm (Design Choices C[a])</li> </ul>
<b>Related Patterns</b>	<p>TP4 (Fixed Date Elements): <i>Schedule Restricted Elements</i> often become Fixed Date Elements when a certain element of the schedule gets selected.</p> <p>TP6 (Time-based Restrictions): Like <i>Schedule Restricted Elements</i>, they constrain possible execution times.</p> <p>TP8 (Time-dependent Variability): <i>Time-dependent Variability</i> is often used to provide alternatives for <i>Schedule Restricted Elements</i>.</p>

Fig. 17 TP5 - Schedule Restricted Element

As a consequence, *Schedule Restricted Elements* are often treated like *Fixed Date Elements* as soon as the possible execution time is constrained, i.e., a particular time slot of the schedule is selected (e.g., a particular week is chosen for publishing an advertisement in the local weekly newspaper).

#### 4.2.3 Pattern TP6 (Time-based Restrictions)

Sometimes, it becomes necessary to restrict the number of times a particular activity or process may be executed within a predefined time frame.

*Example 11 (Time-based restriction)* Two invasive examinations for one and the same patient must not be performed at the same day.

Additionally, *Time-based Restrictions* are often required to express the influence of resource restrictions (e.g. resource shortage) on process execution.

*Example 12 (Mutual exclusion)* For different patients two X-ray activities cannot be executed at the same time as the X-ray machine cannot be shared.

Time pattern TP6 enables such restrictions. It is described in Fig. 18. *Design Choice G* describes to which process granularities the pattern may be applied (cf. Fig. 18). Depending on the context, for example, two X-ray examinations of the same patient may be modelled as two different activities in a single process, two activities belonging to different process instances sharing a common characteristics (i.e., the patient), or two different process instances. In addition, *Design Choice H* allows specifying whether a minimum or maximum number of executions of the respective activities (processes) is given (cf. Fig. 18). Finally, TP6 either restricts the number of concurrent executions of the activity (process) or the overall number of executions per time period (*Design Choice I*, cf. Fig. 18).

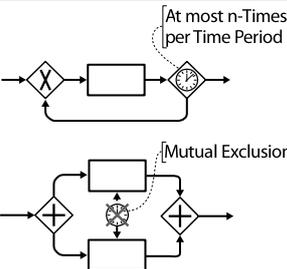
Time Pattern TP6: Time-based Restrictions	
<b>Also known as</b>	A particular variant of this pattern is often referred to as “Mutual Exclusion”
<b>Problem</b>	A particular activity or process may only be executed a limited number of times (has to be executed at least a certain number of times) within a given time frame.
<b>Design Choices</b>	<p>G) Time-based restrictions can be applied to different types of process granularities</p> <ol style="list-style-type: none"> <li>Instances of a single activity or a group of activities in the context of the same process instance</li> <li>Instances of a single activity or a group of activities in the context of different process instances</li> <li>Instances of a process or a group of processes</li> </ol> <p>H) There are two kinds of restrictions</p> <ol style="list-style-type: none"> <li>Minimum number of executions</li> <li>Maximum number of executions</li> </ol> <p>I) There are two types of restrictions which can be expressed</p> <ol style="list-style-type: none"> <li>Number of concurrent executions (at the same time / with overlapping time frames)</li> <li>Number of executions per time period</li> </ol>
<b>Solution</b>	<p>To implement this pattern a constraint expressing a particular time-based restriction is associated with the activities (processes) affected by this restriction. Additionally, the constraint specifies the respective time period and the number of executions.</p>  <p>During run-time an observer can be used to monitor the number of running instances per time period and to raise an exception in case the maximum (minimum) number of executions is exceeded (fallen short of).</p>
<b>Context</b>	The minimum / maximum number of executions needs to be known to the observer before any of the respective activities or processes is started.
<b>Examples</b>	<ul style="list-style-type: none"> <li>Several examinations for a particular patient are performed within a limited timeframe; Thereby, it has to be ensured that the patient is not x-rayed several times (Design Choices G[b] H[c] I[b]).</li> <li>For USD 19.90 ten different e-books can be read <i>per month</i>. If the e-book tokens are consumed no more books can be read in the current month. At the beginning of the next month the book tokens get renewed (Design Choices G[a] H[b] I[b]).</li> <li>For a specific lab test at least 5 different blood samples have to be taken <i>within 24 hrs</i> (Design Choices G[c] H[a] I[a])</li> </ul>
<b>Related Patterns</b>	TP5 (Schedule Restricted Elements): While the execution time of a <i>Schedule Restricted Element</i> is constrained by a schedule, <i>Time-based Restrictions</i> constrain the number of activity instances per time period.

Fig. 18 TP6 - Time-based Restrictions

#### 4.2.4 Pattern TP7 (Validity Period)

In general, different versions of an activity or process may exist, but only one version shall be valid at a specific point in time. *Validity Periods* are relevant, for example, in the context of process model evolution [53] to restrict the remaining lifetime of an obsolete process implementation and to schedule the rollout of the new process version.

*Example 13 (Validity period)* Due to a changed law, process model *Revision A* may only be used until January 1st. Afterwards, no process instances can be created based on *Revision A* anymore, but process model *Revision B* has to be used instead.

Time pattern TP7, which is described in detail in Fig. 19, enables us to restrict the lifetime of an activity

or process to a given validity period. Similar to time pattern TP4, a *Validity Period* can be also applied to activities or processes (*Design Choice C[a, c]*). Further, it can be used to restrict the earliest start, latest start, earliest completion, or latest completion of the respective activity or process (*Design Choice F*). Since a *Validity Period* may prohibit the execution of an activity or process, its parameters need to be known at build-time or at least at creation time of a process instance, i.e., *Design Choice A[c]* (parameter values are set during run-time, cf. Fig. 7) is not applicable to this pattern.

Generally, a *Validity Period* cannot only be bound to an activity or process, but to an invocable application service as well. In this case, *Validity Period* applies to all activities being associated with the respective

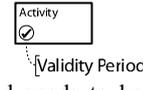
Time Pattern TP7: Validity Period	
<b>Also known as</b>	-
<b>Problem</b>	A particular activity or process may be only executed within a particular validity period, i.e., its lifetime is restricted to this validity period. The respective activity or process may only be instantiated within this validity period.
<b>Design Choices</b>	A) Parameters of this pattern may only be set at build- A[a] or instantiation-time A[b] (cf. Fig. 6). C) A validity period can be applied to an activity C[a] or process C[c] (cf. Fig. 7) F) A validity period can restrict all four types of dates (cf. Fig. 15)
<b>Solution</b>	To realize this pattern a validity period is attached to the respective activity or process respectively.  Upon instantiation of the respective activity or process, its validity period needs to be checked. If the current time does not match with the activities (processes) validity period or the minimum duration of the activity (process) (see Fig. 12) will result in the completion of the activity (process) being outside of the respective validity period, appropriate exception handling is required.
<b>Context</b>	The validity period needs to be known before the respective process is executed.
<b>Examples</b>	<ul style="list-style-type: none"> <li>Starting from Jan 1st patients need to be informed about any risk before the actual treatment takes place (Design Choice C[c] F[a]).</li> <li>From next week on the new service version should get life (Design Choice C[a] F[a]).</li> </ul>
<b>Related Patterns</b>	TP8 (Time-dependent Variability): TP8 is often required to switch between activities having different validity periods.

Fig. 19 TP7 - Validity Period

application service, i.e., same *Validity Period* may apply to activities in several different process schemes.

#### 4.3 Pattern Category III: Variability

This category comprises a single pattern for specifying varying control flow, depending on time aspects (TP8: Time-dependent Variability).

##### 4.3.1 Pattern TP8 (Time-dependent Variability)

Depending on time aspects, in certain processes, different paths of the control flow may be chosen.

*Example 14 (Time-dependent variability)* Between 7 am and 6 pm the medical lab may analyze all parameters of a sample, while at other times, only limited services are provided and thus only a limited set of lab parameters may be analyzed.

*TP8: Time-dependent Variability* allows for a varying control flow depending on time aspects. As described in Fig. 20, this may either be the execution time of a node within a process schema (i.e., activity or control connector) or time lags between two activities (e.g., if the first activity of a specific path is not started within a certain time frame, an alternative path will be chosen) (*Design Choice J*). As example consider the process depicted in Fig. 21. Depending on the point in time the respective process is executed or, more precisely, the

point in time the XOR decision is evaluated, either the upper or the lower path is chosen. For example, when evaluating the XOR decision between 8 am and 6 pm the upper path will be chosen, otherwise the lower one.

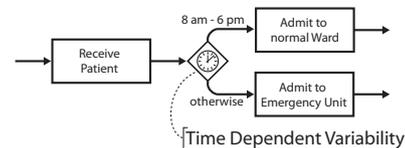


Fig. 21 TP8 - Time-dependent XOR Decision

#### 4.4 Pattern Category IV: Recurrent Process Elements

This category comprises patterns to express restrictions regarding cyclic activities / process fragments (cf. TP9: Cyclic Elements) as well as periodicity (cf. TP10: Periodicity). While emphasis of time pattern TP9 is on time lags between loop iterations, pattern TP10 emphasizes possible execution dates of the recurrent activities. Therefore, time pattern TP10 allows expressing more complex repetition patterns compared to TP9.

##### 4.4.1 Pattern TP9 (Cyclic Elements)

Regarding iteratively performed activities or process fragments, it might become necessary to restrict time

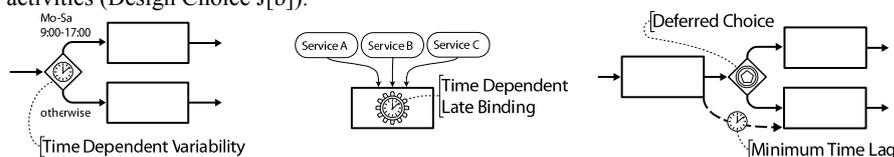
Time Pattern TP8: Time-dependent Variability	
Also known as	-
Problem	Depending on temporal conditions the control flow may vary; e.g., different branches of a process are executed or different sub-process fragments are chosen.
Design Choices	C) A time-dependent variability may only be applied to an activity C[a] (cf. Fig. 7) J) There are different time aspects which may be considered by an instance of this pattern a) Execution time of an activity 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 by enumerating all possible options. Alternatively, techniques like late binding can be used to select appropriate activity implementations during run-time depending on temporal conditions. Both approaches realize the variability based on the execution time of the respective node (Design Choice J[a]). Finally, the Deferred Choice Workflow Pattern [2] 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 condition needs to be able to access any required data when determining which of the possible alternatives shall be chosen.
Examples	<ul style="list-style-type: none"> <li>• If no offer is received 7 days after having sent the request the request may be canceled (Design Choice J[b]).</li> <li>• When issuing a passport the processing <i>usually takes 4-6 weeks</i>. If the person needs the passport <i>earlier than 4 weeks</i> an interim passport can be issued (Design Choice J[a]).</li> <li>• A patient admitted to a hospital <i>between 6 pm and 8 am</i> is always assigned to the emergency unit (for the first night). If no threatening situation exists the following day the patient is transferred to a normal ward. <i>Between 8 am and 6 pm</i>, in turn, a patient is usually directly admitted to the ward unless there is an emergency (Design Choice J[a]).</li> </ul>
Related Patterns	TP5 (Schedule Restricted Element): <i>Time-dependent Variability</i> often refers to some sort of a schedule. TP7 (Validity Dates): TP7 requires different paths to be taken depending on the execution time.

Fig. 20 TP8 - Time-dependent Variability

lags between two activity instances belonging to different iterations of a loop structure. This may be either instances of the same activity or instances of two different activities belonging to the same loop structure.

*Example 15 (Cyclic element)* Administer 1 ml of a particular drug every 2 to 3 hours until symptoms improve (or a certain end date is reached).

The respective pattern TP9 is described in Fig. 22. It represents a special variant of pattern TP1 (*Time Lags between two Activities*, cf. Fig. 10). Pattern TP9 enables us to define time lags between two activities contained within a loop structure. Thereby, TP9 defines a time lag across different loop-iterations. As example consider the process fragment depicted in Fig. 23. It has a *Cyclic Element* between B and A. When applying the *Cyclic Element* during run-time, there exists an instance of the Cyclic Element for each iteration of the respective loop. For each instance of the pattern, the

particular instance of the target activity belongs to the loop-iteration succeeding the one to which the instance of the source activity belongs. Consequently, after each invocation of B, there exists a time lag between the respective instance of B and the invocation of A in the next iteration of the loop (as depicted in the lower part of Fig. 23).

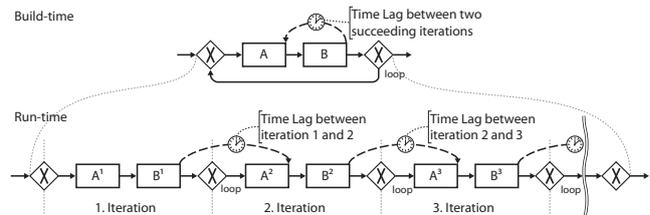


Fig. 23 Cyclic Elements between Succeeding Iterations

*Cyclic Elements* may only be specified for activities (*Design Choice C[a]*). Additionally, *Design Choice D*

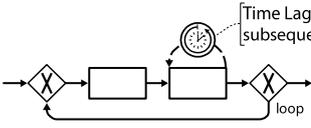
Time Pattern TP9: Cyclic Elements	
<b>Also known as</b>	-
<b>Problem</b>	In the context of iteratively performed activities or process fragments, there is a given time lag between activities, where respective instances of these activities may belong to different iterations of the loop structure. This may either be instances of a single activity or of two different activities belonging to the same cycle.
<b>Design Choices</b>	C) A cyclic element may only restrict the time lag between activities (a) (cf. Fig. 7) D) Cyclic elements may represent all three kinds of restrictions (cf. Fig. 8) E) Cyclic elements can be realized based on all four time relations (cf. Fig. 10) K) Cyclic element may restrict the time lag between <ol style="list-style-type: none"> <li>two directly succeeding iterations</li> <li>two subsequent activity instances belonging to arbitrary iterations</li> </ol> L) Time lag between cycles <ol style="list-style-type: none"> <li>is fixed</li> <li>may vary between iterations</li> </ol>
<b>Solution</b>	A special time constraint is introduced between the start / end events of the activities where the respective event of the second activity is considered to be in a succeeding iteration of the event referring to the first activity.  This pattern can be realized at run-time similar to pattern TP1 with additional attention being paid to the iterations of the respective activities.
<b>Context</b>	The mechanism evaluating the constraint (i.e., starting the timer) needs to be able to access the value of the time lag when it starts the timer. Additionally, time lags may vary between iterations (cf. Design Choice L). Therefore, these requirements need to be fulfilled for each iteration of the respective loop.
<b>Examples</b>	<ul style="list-style-type: none"> <li>Administer 50 to 75 mg in equally divided doses <i>every 12 hrs for 5 subsequent days</i> (Design Choices D[c] E[c] K[a] L[a]).</li> <li>Maintenance aircraft “C Checks” are performed <i>every 12-18 months</i> (Design Choices D[c] E[c] K[b] L[a])</li> <li>Cycle Time for maintenance aircraft “C Checks” is <i>at least 2 weeks</i> (Design Choices D[a] E[a] K[a] L[a])</li> </ul>
<b>Related Patterns</b>	TP10 (Periodicity): TP9 and TP10 both refer to iteratively performed activities. TP1 (Time Lags between Activities): TP9 is an extension of TP1 considering loops and iterations.

Fig. 22 TP9 - Cyclic Elements

allows choosing among maximum time lag, minimum time lag, and time interval. Similar to TP1, pattern TP9 may describe a start-start, start-end, end-start, or end-end relation between the respective activities (*Design Choice E*). *Cyclic Elements* may not only restrict the time lag between two activity instances of directly succeeding loop iterations, but also between two subsequent activity instances belonging to arbitrary loop iterations (*Design Choice K*). As example consider the process fragment depicted in Fig. 24, which has a *Cyclic Element* between two subsequent iterations of activity A. Since this activity is part of an exclusive choice, it is not necessarily executed in each iteration of the respective loop. Therefore, the time lag may apply to two arbitrary iterations of the loop (as indicated in the lower part of Fig. 24). Finally, *Design Choice L* specifies whether the time lags between cycles are fixed (i.e., have always same length) or may vary from iteration to iteration.

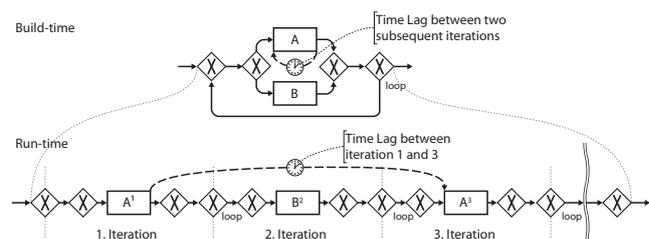


Fig. 24 Cyclic Elements between Subsequent Iterations

#### 4.4.2 Pattern TP10 (Periodicity)

Especially in the medical domain, there are frequently recurring tasks to be executed according to a periodical specification (e.g., a treatment plan) (e.g. [9,65]). Thereby, *periodical* implies some regularity, but does not necessarily mean equally distanced.

Time Pattern TP10: Periodicity	
<b>Also known as</b>	Recurrence, Appointment Series
<b>Problem</b>	A particular set of activities shall be performed periodically (i.e., according to a particular periodicity rule). Thereby, periodically implies some regularity, but does not necessarily mean equally distanced.
<b>Design Choices</b>	C) A Periodicity may only be applied to process fragments (i.e., activity sets) C[b] (cf. Fig. 7) M) The Number of cycles is a) determined by a fixed / dynamic number of iterations, b) depends on end date, or c) depends on exit condition
<b>Solution</b>	A variant of an Ad-hoc Sub-Process [52] may be used in combination with an associated periodicity rule.  Periodicity rules can be realized by combining patterns TP1-6, TP8 and TP9 (cf. Fig. Fig. 10 - Fig. 22) which are applied when scheduling the activities of the respective Ad-hoc Sub-Process.
<b>Context</b>	The context requirements of the participating time patterns need to be fulfilled.
<b>Examples</b>	<ul style="list-style-type: none"> <li>Starting with next Monday group meetings will take place <i>every two weeks at 11:30 am</i> (Design Choices M[c]).</li> <li><i>Each day at 7 am</i> the responsible assistant physician of the Gynecological Clinic is informing the assistant medical director about the patients (Design Choices M[c]).</li> <li>Course "Business Processes and Workflows" takes place <i>every Monday from 8:00 am to 11:00 am starting on Oct 6th and ending on Jan 26th</i>. On Dec 8th, 22nd, 29th and on Jan 5th no lectures will take place (Design Choices M[b]).</li> <li>An information letter is sent by the leasing company to each customer <i>within the first two weeks of each year</i> (Design Choice M[c]).</li> </ul>
<b>Related Patterns</b>	TP9 (Cyclic Elements): TP9 and TP10 both refer to iteratively performed activities.

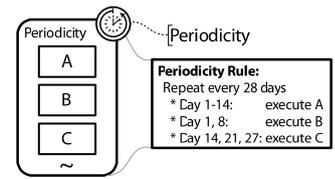


Fig. 25 TP10 - Periodicity

Example 16 (Periodicity) A chemotherapy, as illustrated in Fig. 26, may comprise 6 treatments performed every 28 days. On day 1 to 14 of each of the 6 treatment cycles Drug A is administered and on the 1st and the 8th day, additionally, Drug B is given. This is followed by 14 days without giving any drug.

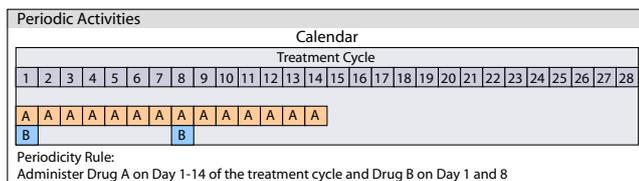


Fig. 26 A Simple Periodicity Rule

Time pattern TP10, which is further described in Fig. 25, allows specifying periodically recurring sets of activities according to an explicitly defined periodicity rule (see [9,65]). Such periodicity rule describes the recurrence schema of the respective activities (e.g., every Monday and Wednesday at 11:30 am) as well as some sort of *exit condition* (e.g., until end of the year, maximum 5 times) (Design Choice M). A periodicity rule may be described using one or more schedules (cf.

Example 9). Each of the activities of the periodicity is annotated with one of these schedules. Thereby, each activity has to be executed exactly once for each time slot of the respective schedule. However, note that the schedules used to describe a periodicity rule may be not independent from each other, i.e., one schedule may define exceptions for another schedule.

As opposed to time pattern TP9, emphasis of TP10 is on possible execution dates of the recurrent activities and not on the time lags between activity instances from different iterations. Additionally, as illustrated in Fig. 25, periodicity rules may involve more than two activities (i.e., periodicity rules may refer to three or more activities at the same time), whereas TP9 always connects exactly two activity instances. Finally, periodicity rules may contain exceptions, e.g. every year except leap years.

Periodicity can be realized by the combined use of patterns TP1-6, TP8, and TP9. However, even for simple periodicity rules this might lead to complex processes for which the underlying periodicity rule is quite hard to recognize, i.e., it is difficult to decide whether or not a set of loops and activities represent a periodicity, let alone identifying the periodicity rule itself. Furthermore, peri-

odicity rules are not always known at build-time (e.g., treatment plans are regularly decided during run-time of the process); in this case, it is not possible to pre-specify a corresponding process fragment. Hence, *Periodicity* as additional layer of abstraction becomes necessary to describe such processes in an understandable way.

#### 4.5 Summary

We have identified more than 100 different variants of the described time patterns (not considering system-specific variants). To cover this high number, we use design choices for parameterizing the respective patterns, thus keeping their number manageable (see Fig. 27). This results in a set of 10 representative time patterns, which are summarized in Fig. 27. An integration of the time patterns into the used process meta-model (cf. Fig. 2) can be found in Appendix A.

Revisiting our initial example (cf. Example 1 and Fig. 1) we can now classify the temporal constraints encountered by using the time patterns:

*Example 17 (Treatment Process Revisited)* Observing the temporal constraints discussed in Example 1 (see Fig. 1 and Fig. 28 respectively), it can be recapped that the *appointment* for **perform treatment** is a *Fixed Date Element* (TP4) restricting the earliest start date (Design Choice F[a]) of the respective activity (Design Choice C[a]). Further, its value will be set during run-time (Design Choice A[c]) by activity **make appointment**. Restricting activity **prepare patient** to be performed *exactly 1 day before perform treatment* is a *Time Lag between two Activities* from the start of **prepare patient** to the start of **perform treatment** (TP1, Design Choice D[c] and E[c]). The constraint regarding the execution time of **prepare patient**, in turn, represents a *Schedule Restricted Element* (TP5) and the one concerning **prepare treatment** a maximum *Duration* (TP2). Finally, the treatment plan for **perform aftercare** constitutes a *Periodicity* (TP10); to be more precise, it constitutes the underlying periodicity rule of the *periodicity* represented by activity **perform aftercare**.

Based on these considerations, we can compare different PAISs in respect to their ability to support the temporal perspective of this particular process. Subsequently, we may be able to identify a PAIS suitable for modeling and supporting the respective business process (see Section 6 for an evaluation of selected PAISs).

Regarding the domains we used as data sources (cf. Section 3.2) for identifying the time patterns, Table 2 gives an overview about the relevance of each time pattern in the different process scenarios considered. The

table shows that different domains may have different requirements regarding the support of time patterns. While in some domains, a high coverage of the more advanced time patterns like *Periodicity* is urgently needed (e.g., the healthcare domain), for others the more basic time patterns like *Durations* and *Fixed Date Elements* suffice (e.g., automotive domain).

## 5 Temporal Constraints in PAISs: A Systematic Literature Review

To further assess the relevance and completeness of the presented time patterns as well as to evaluate the potential bias caused by the data sources we selected for pattern identification, we conduct a systematic literature review [14] of the primary studies dealing with temporal constraints in the context of business process management. To our best knowledge, there has not been any effort to systematically select, review, and synthesise the literature on this topic so far. Our search strategy identified over 10.000 papers of which 73 were identified as primary papers relevant in the context of our research. Section 5.1 describes the review process applied and Section 5.2 discusses the results obtained.

### 5.1 The Review Process

The concept of temporal constraints in PAISs is not only relevant in various domains, but is also known under different synonyms. In particular, we identify the synonyms *time* and *temporal* as well as *constraint* and *restriction*. Similarly, *business processes* are also known under the synonym *workflow*. Based on these keywords we conduct the systematic literature review. More specifically, we derive the following keyword pattern for our search:

```
( "time constraint" OR "temporal constraint"
OR "time constraints" OR "temporal constraints"
OR "time restriction" OR "temporal restriction"
OR "time restrictions" OR "temporal restrictions" )
AND
( "business process" OR "workflow" )
```

A first try using the above search pattern reveals numerous hits stemming from the grid computing area. Since grid computing is different from our main research interest, we refine our search pattern by explicitly excluding the keyword *grid*. We then utilize the resulting search pattern for a full text search using Google's academic search engine.<sup>3</sup> To further narrow the focus of our search, we limit it to the three subject areas "*Business, Administration, Finance, and Economics*", "*Engineering, Computer Science, and Mathematics*", and

<sup>3</sup> <http://scholar.google.com>

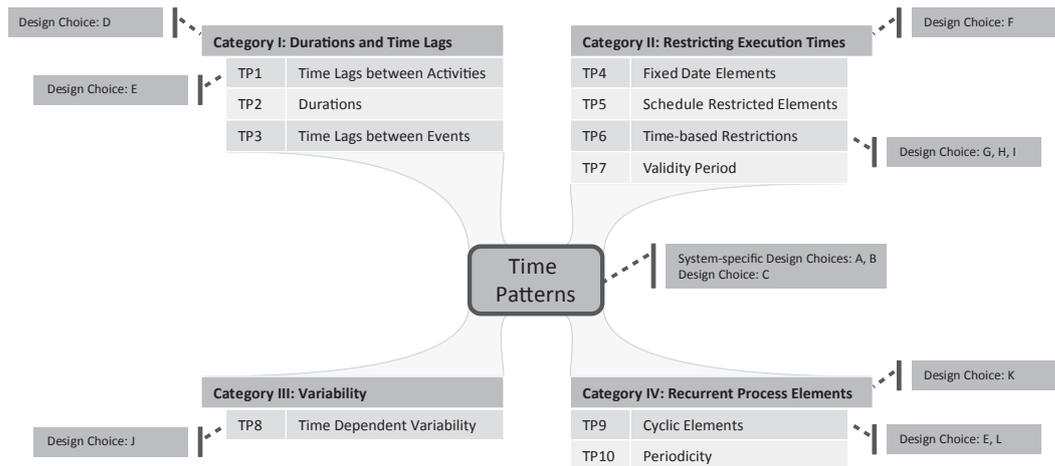


Fig. 27 Overview of Identified Time Patterns and Relevant Design Choices

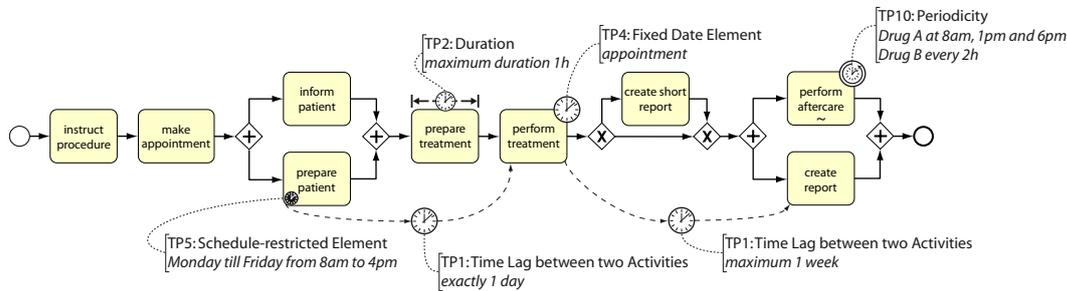


Fig. 28 Treatment Process with Temporal Constraints

Table 2 Relevance of Time Patterns for Different Domains

	TP1	TP2	TP3	TP4	TP5	TP6	TP7	TP8	TP9	TP10
Healthcare Domain	+	+	+	+	+	+	+	+	+	+
Automotive Domain	+	+	+	+				+	+	+
On-demand Air Service	+	+	+	+	+				+	
Airline Catering	+	+	+	+				+	+	
Transportation	+	+	+	+	+	+		+		
Software Engineering	+	+	+	+					+	+
Event Marketing	+	+	+	+	+				+	
Financial Service	+	+	+	+		+	+			
Municipality	+	+	+	+	+					

“*Medicine, Pharmacology, and Veterinary Science*”; we further exclude patents. This finally results in approximately 10.000 hits. We do not use any restriction with respect to the publication date, but are aware that an on-line search might provide only publications belonging to a certain time period. To limit our efforts to a manageable number of publications as well as to cope with restrictions imposed by Google Scholar we limit our review to the first 1.000 hits. We are aware that this constitutes a restriction. However, since Google Scholar sorts its results by relevance (i.e., mainly by the number of citations [56]), we believe that our literature review is still representative. Following this, we remove all publications not related to business processes, viewing temporal constraints only in the sense of ordering

relations (e.g., Allen’s interval algebra [8]), or only mentioning temporal constraints at an abstract level (i.e., no specific temporal constraints are discussed). Altogether, 73 relevant publications pass this manual filtering and hence are further analyzed.

## 5.2 Analysis Results

Having identified relevant publications and collected data from them, each paper is systematically checked for the temporal constraints mentioned. As the most important result, our analysis does not reveal any temporal constraint being relevant for the control-flow perspective, which has not been covered by our time patterns

yet. Albeit, the analysis reveals additional temporal constraints being relevant for other process perspectives. For example, [72, 71] discuss validity periods as well as maximum processing times for process data. The approach presented in [43], in turn, points to the necessity for temporally restricted process changes. However, these temporal constraints are not relevant in the context of our initial research question (i.e., the selection criteria defined in Section 3) and are therefore out of scope of this paper.

Finally, to evaluate the relevance of our time patterns, we analyse the identified 73 primary sources in respect to their support of the time patterns presented in this paper. Consolidated results are presented in Table 3. Due to lack of space, we aggregate the results taken from publications of the same research group in a single row. Additionally, we only show the results of the most active groups in Table 3 (i.e., groups having at least two papers within the 73 relevant publications).<sup>4</sup>

As Table 3 indicates, all time patterns identified (cf. Section 4) can be found in literature as well. Additionally, Table 3 shows that the time patterns we proposed are complete in the sense that they cover all temporal constraints being relevant for the control-flow perspective mentioned in literature. Finally, Table 3 shows that our time patterns provide the most complete framework regarding temporal constraint support in PAISs.

Table 3 further shows that certain patterns received more attention than others in the past. Moreover, the presentation format used in Table 3 conceals that in most cases not all pattern variants are considered by a single source, i.e., in most cases only a small sub-set of the identified design choices is considered. Therefore, it still remains unclear how much support for our patterns can be found in existing PAISs and research approaches. This challenge is picked up in Section 6.

## 6 Evaluation

In the following we describe the evaluation of selected approaches from academia and industry regarding their support of the time patterns. Section 6.1 describes our evaluation methodology, while Section 6.2 discusses evaluation results.

### 6.1 Evaluation Methodology

We first describe the methodology employed for conducting our pattern-based evaluation. In particular, we de-

scribe the evaluation goal, evaluation objects, evaluation criteria, evaluation metrics, and evaluation procedure.

**Defining the evaluation goal.** The goal of our evaluation is to measure how well current PAISs and related technologies cope with temporal constraints.

**Selecting evaluation objects.** As evaluation objects we choose process management systems from both academia and industry, calendar systems, and project planning tools. In terms of academic approaches, our evaluation considers the proposals made by Bettini [12], Combi [15], Eder [22], and Zhuge [73]. Respective approaches are selected based on our systematic literature review, and are required to consider at least 4 of the time patterns and to also provide an implementation for some of them. As samples of commercial systems, our evaluation includes the process management systems IBM WebSphere, IBM WebSphere Lombardi Edition, AristaFlow BPM Suite [19, 50, 30, 29], Intalio, and TIBCO Business Studio, for which we have hands-on experience as well as running installations in our lab. Furthermore, with BPMN and BPEL our evaluation comprises two commonly used process modeling languages. Finally, we include MS Outlook as representative of calendar systems as well as the project management tool MS Project. Both kinds of systems include support of temporal aspects.

**Defining evaluation criteria and metrics.** Evaluation criteria are the 10 time patterns presented in Section 4. We measure the ability of a PAIS to deal with the time perspective 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 (e.g., by a work-around) this is indicated by the additional label “\*” and if support is *not specified* this is indicated by the label “?”. Missing support is labeled with “-”.

Assume that a particular evaluation object supports Pattern TP4 with *Design Choices* C and F. Further assume that for *Design Choice* C the corresponding *Option* a is supported, while *Option* c is only partially supported. Furthermore, for *Design Choice* F, *Option* d is supported while *Option* a is only partially supported. This would result in String “C[a,c\*], F[a\*,d]” in our evaluation table (e.g., TP4 for IBM WebSphere Lombardi Edition in Table 5).

**Analyzing evaluation objects along the evaluation criteria.** For the considered academic approaches, we base our evaluation on the results obtained in Section 5 and on a more detailed literature study. Regarding commercial systems, calendar systems, and project

<sup>4</sup> On request the full list can be provided by the authors. Further details are available at [www.timepatterns.org](http://www.timepatterns.org).

**Table 3** Consolidated Results of our Systematic Literature Review

Research Group	Patterns									
	TP1	TP2	TP3	TP4	TP5	TP6	TP7	TP8	TP9	TP10
Bettini et al. (e.g., [12, 11])	X	X	X	X						
Combi et al. (e.g., [17, 15])	X	X	X	X	X	X	X		X	X
Eder et al. (e.g., [22, 21])	X	X		X	X					
Li et al. (e.g., [36, 37])	X	X		X						
Mans et al. (e.g., [39, 38])	X	X		X	X			X		
Marjanovic et al. (e.g., [41, 40])	X	X		X						
Müller et al. (e.g., [44, 43])		X		X						
Sadiq et al. (e.g., [61, 60])	X	X		X					X	
Zhugue et al. (e.g., [74, 73])	X	X		X					X	

management tools, support for time patterns is determined based on installations in our lab and hands-on experiences with respective systems. Process modeling notations (BPMN and BPEL) are evaluated by studying the specification of the respective standard. Note that this evaluation only considers time patterns. Time features, in turn, like the verification of temporal constraints, escalation mechanisms, or scheduling support are outside of the scope of this paper.

## 6.2 Evaluation Results

Tables 4 and 5 show which time patterns are supported by our evaluation objects.<sup>5</sup> Calendar systems like MS Outlook (cf. Table 4) provide good support for Patterns TP4 (*Fixed Date Element*) and TP9 (*Cyclic Elements*), while limited support is provided for specifying business rules and regulations (e.g., patterns TP1 and TP2). Due to the nature of these systems, no support can be provided for more complex patterns depending on the concept of a process or a control flow (e.g., patterns TP6 and TP8). In addition to the features of calendar systems, project management tools like MS Project (cf. Table 4) provide some support for specifying business rules and regulations (e.g., patterns TP1 and TP2). However, note that project management systems lack operational support for concurrently executed process instances.

Academic approaches (cf. Table 4) are comparably more expressive and provide good support for specifying business rules and regulations. However, except for the proposal made by Combi et al. [15], the evaluated approaches do not consider loops resulting in missing support for the time patterns of Category IV (*Recurrent Process Elements*). Additionally, academic approaches provide no support for time patterns related to variability during process execution (e.g., patterns TP6, and TP8). Note the differences between Table 3 and 4 regarding the support of time patterns by academic

approaches (e.g., time patterns TP4 and TP7 for Bettini et al., or time pattern TP6 for Combi et al.). These differences can be explained by the fact that while Table 3 contains every pattern mentioned by the respective publications, Table 4 only depicts patterns for which an implementation is available. Additionally, Table 4 contains patterns not explicitly mentioned, but which may be realized using the presented approach. Table 3 solely contains time patterns explicitly mentioned.

BPMN and BPEL as representatives of process modeling languages provide limited support of temporal constraints (cf. Table 5). In particular, they do not provide any support for more advanced temporal constraints (e.g., patterns TP5, TP6, and TP10).

The support of temporal constraints in commercial process management systems (cf. Table 5) is comparable to the one provided by process modeling languages. However, some extensions exist, e.g., TP7 is supported by IBM WebSphere, and TP5 is considered in IBM Lombardi.

Interestingly, despite its relevance for real world applications (cf. Section 4.5), support for patterns TP6 (*Time Based Restrictions*) and TP10 (*Periodicity*) is missing or very limited in almost all evaluation objects. Additionally, support for Pattern TP5 (*Schedule Restricted Elements*) and Pattern TP7 (*Validity Periods*) is very limited in commercial systems and process modeling standards.

## 7 Run-time Aspects

This section discusses selected issues to be considered when evaluating a PAIS in respect to the *run-time support* of the proposed time patterns.

A fundamental aspect regarding process execution in PAISs concerns the soundness of the process models and the consistency of their temporal specification [12, 16, 22, 41, 47], i.e., the consistency of the total set of temporal constraints of the respective process model. Both are essential to ensure robust and correct execution

<sup>5</sup> Further details are available at [www.timepatterns.org](http://www.timepatterns.org).

Table 4 Evaluation Results (Part 1)

Patterns	Calendar Systems	Project Management	Academic			
	Microsoft Outlook 2010	Microsoft Project 2010	Bettini et al.	Combi et al.	Eder et al.	Zhugue et al.
System-specific Design Choices	A[b,c], B[a*,b*]	A[a,c], B[a,b,c]	A[a,b?,c?], B[a,b,c]	A[a], B[a,b]	A[a,b,c], B[a*]	A[a,c]
<b>Category I: Durations and Time Lags</b>						
TP1 - Time Lags between Activities	—	D[a,b], E[a,b,c,d]	D[a,b,c], E[a,b,c,d]	D[a,b,c], E[a,b,c,d]	D[a,b,c], E[d]	D[a,b,c], E[c*]
TP2 - Durations	C[a], D[b]	C[a,c], D[b]	C[a,c], D[a,b,c]	C[a,c], D[a,b,c]	C[a,c], D[b]	C[a], D[a,b,c]
TP3 - Time Lags between Events	—	—	D[a,b,c]	D[a*,b*,c*]	—	—
<b>Category II: Restrictions of Process Execution Points</b>						
TP4 - Fixed Date Elements	C[a], F[a,b,d]	C[a,c], F[a,d]	C[a], F[a,b,c,d]	C[a], F[a,b*,c,d]	C[a], F[c]	C[a], F[b]
TP5 - Schedule Restricted Elements	—	C[a], F[a*,b*]	—	C[a], F[a,b]	C[a], F[c]	—
TP6 - Time Based Restrictions	—	G[a], H[b], I[a*]	—	—	—	—
TP7 - Validity Period	—	—	C[a?], F[a?,b?,c?,d?]	C[a], F[a,d]	C[a?], F[c?,d?]	C[a?], F[b?]
<b>Category III: Variability</b>						
TP8 - Time Dependent Variability	—	—	—	—	—	—
<b>Category IV: Reoccurring Process Elements</b>						
TP9 - Cyclic Elements	D[a], E[a, c], K[a], L[a]	D[a*], E[a*,c*], K[a], L[a]	—	D[a,b,c], E[a?,b?,c?,d?], K[a], L[b]	—	—
TP10 - Periodicity	—	—	—	M[a,b,c]	—	—

Table 5 Evaluation Results (Part 2)

Patterns	Standards		Commercial				
	BPMN 2.0	WS-BPEL4People 2.0	IBM Websphere Integration Developer 6.1	WebSphere Lombardi Edition 7.1	AristaFlow 1.0.1	Intalio 6.0.3	TIBCO Business Studio 3.4.2
System-specific Design Choices	A[a,b?,c], B[a?,b?,c?]	A[a,c], B[a?,b?,c?]	A[a,c], B[a]	A[a,c], B[a,c*]	A[a,c], B[a,b]	A[a,c], B[a]	A[a,c], B[a]
<b>Category I: Durations and Time Lags</b>							
TP1 - Time Lags between Activities	D[a,b*,c*], E[c*]	D[a,b*,c*], E[c]	D[a,b,c], E[c,d*]	D[a,b*,c*], E[a*,b*,c,d*]	D[b], E[c*,d*]	D[a,b*,c*], E[c]	D[a,b*,c*], E[c*]
TP2 - Durations	C[a,c*], D[b]	C[a,c*], D[b]	C[a,c], D[b]	C[a,c], D[b]	C[a], D[b]	C[a*,c*], D[b]	C[a,c*], D[b]
TP3 - Time Lags between Events	D[a,b*,c*]	D[a]	D[a]	D[a*]	—	D[a]	D[a,b*,c*]
<b>Category II: Restrictions of Process Execution Points</b>							
TP4 - Fixed Date Elements	C[a,b*], F[a,b?,d]	C[a], F[a,d]	C[a], F[a,b*,d*]	C[a,c*], F[a*,d]	C[a], F[b*,d]	—	C[a,c*], F[a,b*,d]
TP5 - Schedule Restricted Elements	—	—	—	C[a*,c*], F[a,b]	—	—	—
TP6 - Time Based Restrictions	—	—	—	—	—	—	—
TP7 - Validity Period	—	—	C[c], F[a]	—	—	—	—
<b>Category III: Variability</b>							
TP8 - Time Dependent Variability	J[a,b*]	J[a,b*]	J[a,b*]	J[a]	J[a]	J[a,b*]	J[a,b*]
<b>Category IV: Reoccurring Process Elements</b>							
TP9 - Cyclic Elements	D[a*], E[c*], K[a], L[a,b]	D[a*], E[c], K[a], L[a,b]	D[a*], E[a*,c], K[a], L[a,b]	D[a*], E[c*], K[a], L[a,b]	D[b], E[c*,d*], K[a], L[a,b]	D[a*], E[c], K[a], L[a,b]	D[a*], E[c*], K[a], L[a]
TP10 - Periodicity	—	—	—	—	—	—	—

of corresponding process instances. Note that soundness is not only relevant in connection with the specification of temporal constraints, but also has to be ensured for other process perspectives like control- and data-flow [1, 48, 51].

Generally, the verification of the temporal constraints defined for a process model at build-time is neither sufficient nor is it always possible. Note that in many cases not all parameters of the time patterns are known at build-time (see *Design Choice A*, cf. Fig. 6). For example, the date of a *Fixed Date Element* (cf. Fig. 16) is

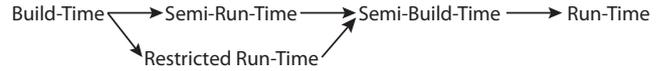
generally set during run-time, as it is specific for each process instance. Moreover, in most cases this date is either set by a preceding activity, an external data source (i.e., an external event), or process input parameters.

However, even if all parameters of a time pattern are already known at build-time, it may not be possible to verify the consistency of a temporal specification. As example consider two activities which may only be executed on working days due to the use of two *Schedule Restricted Elements*. Additionally, assume that there is a minimum time lag of 1 and a maximum time lag

of 3 days between the two activities. These temporal constraints will be met for every execution date of the first activity as long as neither Friday nor Monday are public holidays. However, for the latter case, the first activity must not be executed on a Thursday or Friday. To detect such situations at build-time, each time slot of the schedule of the first activity would have to be compared with each suitable time slot of the schedule of the second activity. This, in turn, is not feasible since both schedules may comprise an infinite set of time slots. Even if we limit the evaluation to a certain time frame, this will become increasingly complex when adding activities and time constraints to the process model.

As a result, according to our experience, specific occurrences of the introduced time patterns belong to five different verification classes. These differ regarding the time when the parameters of the pattern occurrence are set and the time when the validity of a temporal constraint representing a specific pattern may be verified:

- *Build-Time*. All parameters of the pattern are known at build-time. Validity of the pattern occurrence and consistency of the temporal specification can be verified during build-time as well.
- *Semi-Build-Time*. All parameters of the pattern are configured at build-time. Consequently, validity of the pattern occurrence can also be verified at build-time. However, consistency of the temporal specification cannot be verified at build-time.
- *Semi-Run-Time*. Certain properties of the pattern occurrence not configured at build-time are set when creating respective process instances. Only then, validity of the pattern occurrence and consistency of the temporal specification can be verified.
- *Run-Time*. The time pattern is applied at build-time, but certain properties of the pattern occurrence are set during run-time. Hence, validity of the pattern and consistency of the temporal specification can only be verified during run-time (i.e., validity cannot be verified before all parameter values of the time pattern are set).
- *Restricted Run-Time*. Properties of the time pattern are set at build-time, but may be updated during run-time. Thereby, build-time settings serve as restrictions for possible run-time values, i.e., values can only be updated in accordance with the respective build-time values. Hence, validity of the pattern and consistency of the temporal specification can be checked at build-time, but a final decision cannot be made before run-time.



**Fig. 29** Partial Order of Verification Classes

These five categories build a partial order as depicted in Fig. 29. Thereby, the temporal specification of a process model belongs to the smallest class containing all respective patterns. Depending on *Design Choice A* and their inherent properties, the time patterns themselves belong to the classes as depicted in Table 6.

Regarding run-time support and validity of the temporal constraints, additional issues emerge: What happens during run-time if the temporal specification of a process instance is no longer consistent? How can this be prevented? If the temporal specification of a process instance is no longer consistent, the execution cannot be continued as planned, i.e., appropriate exception handling needs to be triggered. Actions taken in this context may be aborting the instance, deciding to ignore the temporal specification of the process instance at all, or fixing the issue that has caused the violation of the consistency of the temporal specification. In the latter case, different *repair strategies* can be applied. The simplest one is to adjust run-time parameters of one or more time constraints (e.g. to adapt the date of a *Fixed Date Element*). If this is not possible, ad-hoc changes [48, 67] can be applied to change the underlying process model in such a way that consistency of the temporal specification is restored (e.g., by removing or reordering activities in the process model or by modifying the temporal specification of the process instance itself). In this context, the impact of different ad-hoc changes [48] on the consistency of the temporal specification still constitutes an open issue.

Generally, one should try to prevent run-time violations of the temporal specification. For example, this can be supported by differentiating between *soft* and *hard* temporal constraints. While a *hard temporal constraint* must be obeyed (i.e., exception handling is triggered when such a constraint is violated), the violation of a *soft temporal constraint* only raises a warning in the PAIS. In the latter case, the PAIS may use escalation techniques to notify the user about the violation of a soft temporal constraint. Hence, it is up to the user to apply counter measures (e.g. hurry up or reorder his priorities) before any hard time constraint is violated.

Besides reactive measures, pro-active measures may be applied as well. Temporal specifications can be used by the PAIS to assist users in planning their work by proposing them a suitable ordering of their tasks, which ensures that none of the temporal specifications of one of

	TP1	TP2	TP3	TP4	TP5	TP6	TP7	TP8	TP9	TP10
Build-Time	A[a]	A[a]	A[a]				A[a]	A[a] <sup>a</sup>	A[a] <sup>b</sup>	A[a] <sup>c</sup>
Semi-Build-Time					A[a]	A[a]		A[a] <sup>a</sup>	A[a] <sup>b</sup>	A[a] <sup>c</sup>
Semi-Run-Time	A[b]	A[b]	A[b]	A[b]	A[b]	A[b]	A[b]	A[b]	A[b]	A[b]
Run-Time	A[c]		A[c]	A[c]	A[c]	A[c]		A[c]	A[c]	A[c]
Restricted Run-Time	A[a, c]	A[a, c]	A[a, c]	A[a, c]		A[a, c]			A[a, c]	A[a, c]

<sup>a</sup>Depending on the complexity of the decision rule.

<sup>b</sup>Depending on the complexity of the loop-structure.

<sup>c</sup>Depending on the complexity of the periodicity rule.

**Table 6** Verification Classes of Time Patterns Depending on Design Choice A

the process instances executed by the PAIS will become inconsistent [23].

Finally, efficiency is extremely important for the run-time support of temporal constraints. When being confronted with hundreds or even thousands of concurrently executed process instances in the PAIS, efficiency becomes crucial. As most methods for verifying the temporal specification of a process model have been designed for build-time, run-time efficiency has not been a big issue yet. Besides, in approaches considering run-time support, build-time algorithms have been simply applied at run-time as well, leading to inefficient solutions not being able to deal with more than a few instances at a time. For the temporal perspective to be properly supported by a PAIS, more efficient methods for verifying the time patterns are required.

## 8 Summary and Outlook

We have proposed 10 time patterns to support the selection of appropriate PAISs and to facilitate their comparison regarding the support of the time perspective. We have shown that the suggested time patterns are highly relevant in practice. In addition, we have evaluated selected approaches and systems regarding their ability to deal with the temporal perspective. The introduction of time patterns complements existing workflow patterns and allows for a more meaningful evaluation of existing systems and approaches, particularly if temporal constraints are crucial. In combination with workflow patterns, the presented time patterns enable PAIS engineers to choose the process management technology meeting their requirements best or closest. Our evaluation shows that currently none of the evaluated systems provides a holistic and integrated support of the temporal perspective. However, in analogy to workflow patterns we expect vendors to evaluate their PAISs along these cri-

teria and to extend them towards better support of the temporal perspective.

Our future work will include time patterns for aspects other than control-flow (e.g., data or process changes). Due to the applied selection criteria, such extended time patterns have not been covered by this work. Further, we will evaluate additional academic and commercial systems. Currently, we are working on the definition of a precise formal semantics of the time patterns and will provide a reference implementation supporting them. Furthermore, we will conduct a comprehensive study on time support features along the entire process life cycle [68], e.g., enabling the verification of time constraints, escalation management, and advanced scheduling support. We will also consider the resource dimension in this context. Finally, we will evaluate the impact, process changes have on the time patterns and respective temporal specifications of business processes.

## A Ontology of a Process Meta-Model including Time Patterns

See Fig. 30.

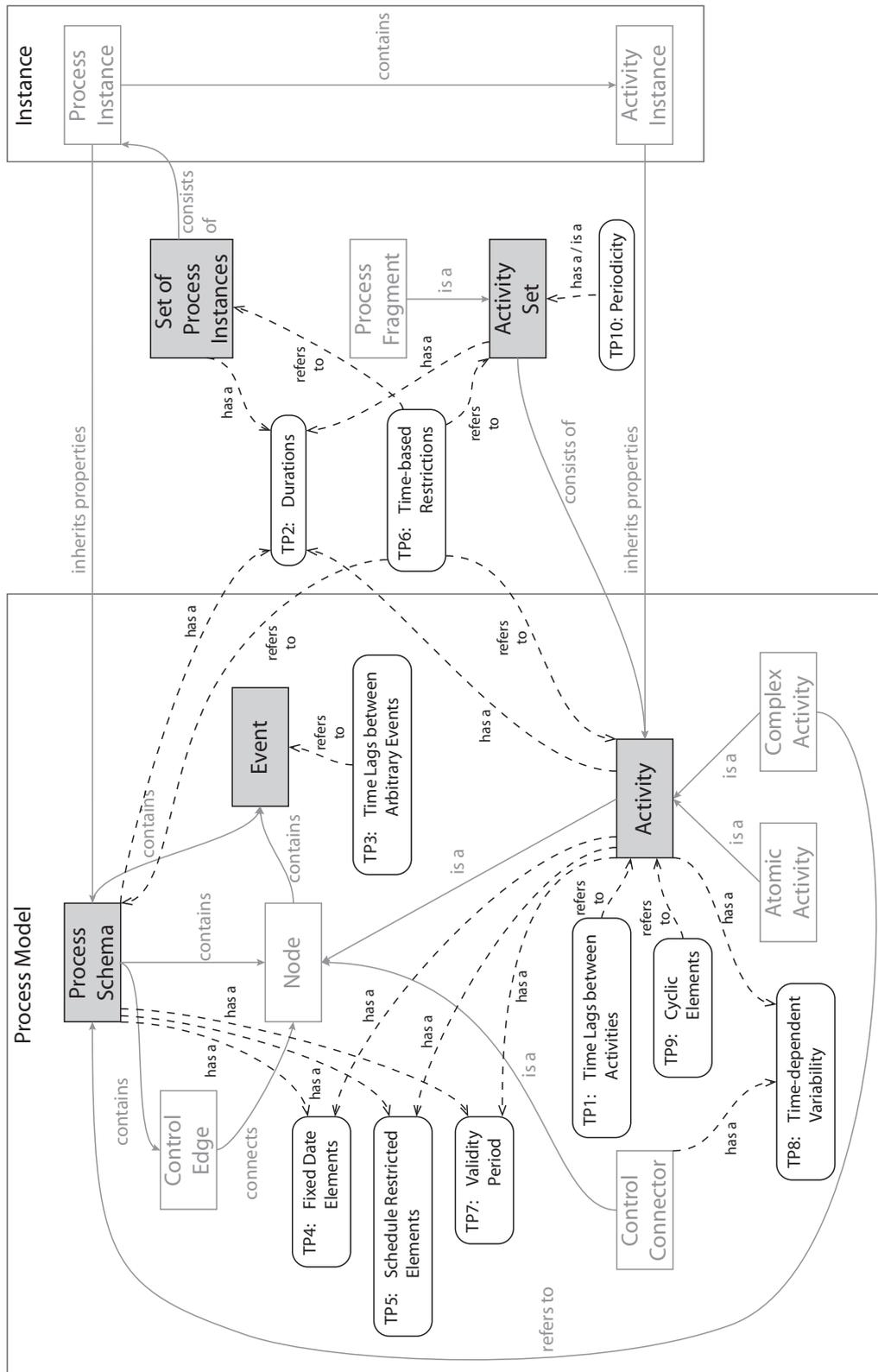


Fig. 30 Ontology of the basic constructs of a Process Meta-Model including Time Patterns

## References

1. van der Aalst, W.M.P.: Verification of workflow nets. In: P. Azéma, G. Balbo (eds.) *Application and Theory of Petri Nets (PETRINETS'97)*, *Lecture Notes in Computer Science*, vol. 1248, pp. 407–426. Springer Berlin / Heidelberg (1997)
2. van der Aalst, W.M.P., van Hee, K.M.: *Workflow Management: Models, Methods, and Systems*. MIT Press (2004)
3. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. *Distributed and Parallel Databases* **14**(1), 5–51 (2003)
4. van der Aalst, W.M.P., Pesic, M., Song, M.: Beyond process mining: From the past to present and future. In: B. Pernici (ed.) *Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE'10)*, *Lecture Notes in Computer Science*, vol. 6051, pp. 38–52. Springer Berlin / Heidelberg (2010)
5. van der Aalst, W.M.P., Rosemann, M., Dumas, M.: Deadline-based escalation in process-aware information systems. *Decision Support Systems* **43**(2), 492–511 (2007)
6. van der Aalst, W.M.P., Weijters, A.J.M.M.: Process mining. In: M. Dumas, W.M.P. van der Aalst, A.H.M. ter Hofstede (eds.) *Process-Aware Information Systems: Bridging People and Software through Process Technology*, chap. 10, pp. 235–255 (2005)
7. Alexander, C., Ishikawa, S., Silverstein, M.: *A Pattern Language*. Oxford University Press, New York (1977)
8. Allen, J.F.: Maintaining knowledge about temporal intervals. In: *Communications of the ACM*, vol. 26, pp. 832–843 (1983)
9. Anselma, L.: Recursive representation of periodicity and temporal reasoning. In: C. Combi, G. Ligozat (eds.) *11th International Symposium on Temporal Representation and Reasoning (TIME 2004)*, pp. 52–59. IEEE Computer Society Press (2004)
10. Barba, I., Lanz, A., Weber, B., Reichert, M., Valle, C.D.: Optimized time management for declarative workflows. In: *13th BPMDS'12 Working Conference, Lecture Notes in Business Information Processing*. Springer Berlin / Heidelberg (2012)
11. Bettini, C., Wang, X.S., Jajodia, S.: Solving multi-granularity temporal constraint networks. *Artificial Intelligence* **140**(1-2), 107–152 (2002)
12. Bettini, C., Wang, X.S., Jajodia, S.: Temporal reasoning in workflow systems. *Distributed and Parallel Databases* **11**(3), 269–306 (2002)
13. Bobrik, R.: *Konfigurierbare Visualisierung Komplexer Prozessmodelle*. Ph.D. thesis, University of Ulm (2008)
14. Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M.: Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software* **80**(4), 571–583 (2007)
15. Combi, C., Gozzi, M., Juarez, J.M., Oliboni, B., Pozzi, G.: Conceptual modeling of temporal clinical workflows. In: V. Goranko, X.S. Wang (eds.) *14th International Symposium on Temporal Representation and Reasoning*, pp. 70–81. IEEE Computer Society Press (2007)
16. Combi, C., Posenato, R.: Controllability in temporal conceptual workflow schemata. In: U. Dayal, J. Eder, J. Koehler, H.A. Reijers (eds.) *Proceedings 7th International Conference on Business Process Management (BPM'09)*, *Lecture Notes in Computer Science*, vol. 5701, pp. 64–79. Springer Berlin / Heidelberg (2009)
17. Combi, C., Pozzi, G.: Towards temporal information in workflow systems. In: *Advanced Conceptual Modeling Techniques (ER 2002 Workshops)*, *Lecture Notes in Computer Science*, vol. 2784, pp. 13–25. Springer Berlin / Heidelberg (2003)
18. Combi, C., Pozzi, G.: Task scheduling for a temporal workflow management system. In: J. Pustajovsky, P. Revesz (eds.) *13th International Symposium on Temporal Representation and Reasoning*, pp. 61–68. IEEE Computer Society Press (2006)
19. Dadam, P., Reichert, M.: The ADEPT project: A decade of research and development for robust and flexible process support - challenges and achievements. *Computer Science - Research and Development* **22**(2), 81–97 (2009)
20. Dadam, P., Reichert, M., Kuhn, K.: Clinical workflows - the killer application for process-oriented information systems. In: *4th International Conference on Business Information Systems (BIS' 2000)*, pp. 36–59 (2000)
21. Eder, J., Panagos, E.: Managing time in workflow systems. In: L. Fischer (ed.) *Workflow Handbook 2001*, pp. 109–132 (2000)
22. Eder, J., Panagos, E., Rabinovich, M.: Time constraints in workflow systems. In: M. Jarke, A. Oberweis (eds.) *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE'99)*, *Lecture Notes in Computer Science*, vol. 1626, pp. 286–300. Springer Berlin / Heidelberg (1999)
23. Eder, J., Pichler, H., Gruber, W., Ninaus, M.: Personal schedules for workflow systems. In: W.M.P. van der Aalst, A.H.M. ter Hofstede, M. Weske (eds.) *Proceedings 1st International Conference on Business Process Management (BPM'03)*, *Lecture Notes in Computer Science*, vol. 2678, pp. 216–231. Springer Berlin / Heidelberg (2003)
24. German Association of the Automotive Industry (VDA): *Engineering change management. part 1: Engineering change request (ECR)* (2005)
25. van Hee, K.M., Reijers, H.A.: Using formal analysis techniques in business process redesign. In: W.M.P. van der Aalst, J. Desel, A. Oberweis (eds.) *Business Process Management - Models, Techniques, and Empirical Studies*, *Lecture Notes in Computer Science*, vol. 1806, pp. 51–71 (2000)
26. Jablonski, S., Bussler, C.: *Workflow Management: Modeling Concepts, Architecture and Implementation*. International Thomson Computer Press (1996)
27. Käfer, R.: *Medical information processing in the hospital - current status, problems and perspectives by the example of the medical university clinic ulm*. Diploma thesis, Heidelberg University (1993)
28. Kuhn, K., Reichert, M., Käfer, R., Köhler, C.: *Situations- und Schwachstellenanalyse der Informationsverarbeitung in einer Universitätsklinik*. In: *Proceedings 39. Jahrestagung der GMDS* (1994)
29. Lanz, A., Kreher, U., Reichert, M., Dadam, P.: Enabling process support for advanced applications with the AristaFlow BPM Suite. In: *Proceedings of the Business Process Management 2010 Demonstration Track*, no. 615 in *CEUR Workshop Proceedings* (2010)
30. Lanz, A., Reichert, M., Dadam, P.: Making business process implementations flexible and robust: Error handling in the AristaFlow BPM Suite. In: *Proceedings CAiSE'10 Forum* (2010)
31. Lanz, A., Weber, B., Reichert, M.: Workflow time patterns for process-aware information systems. In: I. Bider, T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, R. Ukor (eds.) *11th International Workshop, BPMDS 2010, and 15th International Conference, EMMSAD 2010*, *Lecture Notes in Business Information Processing*, vol. 50, pp. 94–107. Springer Berlin / Heidelberg (2010)
32. Li, C.: *Mining process model variants: Challenges, techniques, examples*. Phd thesis, University of Twente, The Netherlands (2010)

33. Li, C., Reichert, M., Wombacher, A.: Discovering reference models by mining process variants using a heuristic approach. In: U. Dayal, J. Eder, J. Koehler, H.A. Reijers (eds.) Proceedings 7th International Conference on Business Process Management (BPM'09), *Lecture Notes in Computer Science*, vol. 5701, pp. 344–362. Springer Berlin / Heidelberg (2009)
34. Li, C., Reichert, M., Wombacher, A.: The MinAdept clustering approach for discovering reference process models out of process variants. *International Journal of Cooperative Information Systems* **19**(3 & 4), 159–203 (2010)
35. Li, C., Reichert, M., Wombacher, A.: Mining business process variants: Challenges, scenarios, algorithms. *Data & Knowledge Engineering* **70**(5), 409–434 (2011)
36. Li, H., Yang, Y.: Dynamic checking of temporal constraints for concurrent workflows. *Electronic Commerce Research and Applications* **4**(2), 124–142 (2005)
37. Li, H., Yang, Y., Chen, T.Y.: Resource constraints analysis of workflow specifications. *Journal of Systems and Software* **73**(2), 271–285 (2004)
38. Mans, R.S., Aalst, W.M.P., Russell, N.C., Bakker, P.J.M., Moleman, A.J.: Process-aware information system development for the healthcare domain-consistency, reliability, and effectiveness. In: S. Rinderle-Ma, S. Sadiq, F. Leymann (eds.) Business Process Management Workshops (BPM'05), *Lecture Notes in Business Information Processing*, vol. 43, pp. 635–646 (2010)
39. Mans, R.S., Russell, N.C., van der Aalst, W.M.P., Moleman, A.J., Bakker, P.J.M.: Schedule-aware workflow management systems. In: K. Jensen, S. Donatelli, M. Koutny (eds.) Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE09), *Lecture Notes in Computer Science*, vol. 6550, pp. 81–96. Springer Berlin / Heidelberg (2009)
40. Marjanovic, O.: Dynamic verification of temporal constraints in production workflows. In: M.E. Orłowska (ed.) Proceedings 11th Australasian Database Conference, *Australian Computer Science Communications*, vol. 22, pp. 74–81 (2000)
41. Marjanovic, O., Orłowska, M.E.: On modeling and verification of temporal constraints in production workflows. *Knowledge and Information Systems* **1**(2), 157–192 (1999)
42. Müller, D., Herbst, J., Hammori, M., Reichert, M.: IT support for release management processes in the automotive industry. In: S. Dustdar, J.L. Fiadeiro, A. Sheth (eds.) Proceedings 4th International Conference on Business Process Management (BPM'06), *Lecture Notes in Computer Science*, vol. 4102, pp. 368–377. Springer Berlin / Heidelberg (2006)
43. Müller, R., Greiner, U., Rahm, E.: AgentWork: a workflow system supporting rule-based workflow adaptation. *Data & Knowledge Engineering* **51**(2), 223–256 (2004)
44. Müller, R., Rahm, E.: Dealing with logical failures for collaborating workflows. In: Proceedings 4th International Conference Cooperative Information Systems, *Lecture Notes in Computer Science*, vol. 1901, pp. 210–223 (2000)
45. Object Management Group: Business Process Model and Notation (BPMN) Version 2.0. <http://www.omg.org/spec/BPMN/2.0> (visited 15.12.2011) (2011)
46. Panagos, E., Rabinovich, M.: Predictive workflow management. In: A. Silberschatz, P. Shoval (eds.) 3rd International Workshop on Next Generation Information Technologies and Systems, pp. 193–197 (1997)
47. Pozewaunig, H., Eder, J., Liebhart, W.: ePert: Extending PERT for workflow management systems. In: Advances in Databases and Information Systems (ADBIS'97), pp. 217–224 (1997)
48. Reichert, M., Dadam, P.: ADEPTflex: Supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems* **10**(2), 93–129 (1998)
49. Reichert, M., Dadam, P.: Towards process-oriented hospital information systems: Some insights into requirements, technical challenges and possible solutions. In: Proc. 43. Jahrestagung der GMDS (1998)
50. Reichert, M., Dadam, P., Rinderle-Ma, S., Lanz, A., Pryss, R., Predeschly, M., Kolb, J., Ly, L.T., Jurisch, M., Kreher, U., Göser, K.: Enabling Poka-Yoke workflows with the AristaFlow BPM Suite. In: CEUR proceedings of the BPM'09 Demonstration Track, Business Process Management Conference 2009 (BPM'09) (2009)
51. Reichert, M., Kolb, J., Bobrik, R., Bauer, T.: Enabling personalized visualization of large business processes through parameterizable views. In: 27th ACM Symposium On Applied Computing, 9th Enterprise Engineering Track. ACM Press (2012)
52. Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies. Springer Berlin / Heidelberg (2012)
53. Rinderle, S., Reichert, M., Dadam, P.: Flexible support of team processes by adaptive workflow systems. *Distributed and Parallel Databases* **16**(1), 91–116 (2004)
54. Rinderle-Ma, S., Reichert, M.: Comprehensive life cycle support for access rules in information systems: The ceosis project. *Enterprise Information Systems* **3**(3), 219–251 (2009)
55. Rinderle-Ma, S., Reichert, M., Weber, B.: On the formal semantics of change patterns in process-aware information systems. In: Q. Li, S. Spaccapietra, E. Yu, A. Olivé (eds.) 27th International Conference on Conceptual Modeling, *Lecture Notes in Computer Science*, vol. 5231, pp. 279–293. Springer Berlin / Heidelberg (2008)
56. Robecke, A., Pryss, R., Reichert, M.: Dbisolar: An iphone application for performing citation analyses. In: S. Nurcan (ed.) CAiSE Forum 2011, *CEUR Workshop Proceedings*, vol. 734 (2011)
57. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Exception handling patterns in process-aware information systems. Tech. rep., BPMCenter.org (2006)
58. Russell, N.C., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow resource patterns. Tech. rep., Eindhoven University of Technology (2004)
59. Russell, N.C., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow data patterns: Identification, representation and tool support. In: L. Delcambre, C. Kop, H.C. Mayr, J. Mylopoulos, O. Pastor (eds.) 24th International Conference on Conceptual Modeling, *Lecture Notes in Computer Science*, vol. 3716, pp. 353–368 (2005)
60. Sadiq, S.W., Marjanovic, O., Orłowska, M.E.: Managing change and time in dynamic workflow processes. *International Journal of Cooperative Information Systems* **9**(1-2), 93–116 (2000)
61. Sadiq, S.W., Orłowska, M.E.: Dynamic modification of workflows. Tech. rep., Department of Computer Science and Electrical Engineering, University of Queensland, Brisbane, Australia (1998)
62. Sayal, M., Casati, F., Dayal, U., Shan, M.C.: Business process cockpit. In: Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02), pp. 880–883 (2002)
63. Schultheiß, B., Meyer, J., Mangold, R., Zemmler, T., Reichert, M.: Designing the processes for chemotherapy treatment in a women's hospital (in german). Tech. Rep. DBIS-5, DBIS, Ulm University, Germany (1996)

64. Schultheiß, B., Meyer, J., Mangold, R., Zemmler, T., Reichert, M.: Designing the processes for ovarian cancer surgery (in german). Tech. Rep. DBIS-6, University of Ulm (1996)
65. Terenziani, P.: Integrating calendar dates and qualitative temporal constraints in the treatment of periodic events. *IEEE Transactions on Knowledge and Data Engineering* **9**(5), 763–783 (1997)
66. 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* **4**(2), 93–110 (2009)
67. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - enhancing flexibility in process-aware information systems. *Data & Knowledge Engineering* **66**(3), 438–466 (2008)
68. Weber, B., Reichert, M., Wild, W., Rinderle-Ma, S.: Providing integrated life cycle support in process-aware information systems. *International Journal of Cooperative Information Systems (IJCIS)* **18**(1), 115–165 (2009)
69. Weske, M.: *Business Process Management: Concepts, Languages, Architectures*. Springer Berlin / Heidelberg (2007)
70. Wombacher, A.: A-posteriori detection of sensor infrastructure errors in correlated sensor data and business workflows. In: S. Rinderle-Ma, F. Toumani, K. Wolf (eds.) *Business Process Management, Lecture Notes in Computer Science*, vol. 6896, pp. 329–344. Springer Berlin / Heidelberg (2011)
71. Xie, J., Tang, Y., He, Q., Tang, N.: Research of temporal workflow process and resource modeling. In: W. Shen, A. James, K.M. Chao, M. Younas, Z. Lin, J.P. Barthes (eds.) *9th International Conference on Computer Supported Cooperative Work in Design*, vol. 1, pp. 530–534 (2005)
72. Yu, Y., Tang, Y., Liang, L., Feng, Z.s.: Temporal extension of workflow meta-model and its application. In: W. Shen, T. Li, Z. Lin, J.P. Barthes, W. Zeng, S. Li, C. Yang (eds.) *8th International Conference on Computer Supported Cooperative Work in Design*, vol. 2, pp. 293–297 (2004)
73. Zhuge, H., Cheung, T.Y., Pung, H.K.: A timed workflow process model. *Journal of Systems and Software* **55**(3), 231–243 (2001)
74. Zhuge, H., Pung, H.K., Cheung, T.Y.: Timed workflow: Concept, model, and method. In: X. Zhou, J. Fong, X. Jia, Y. Kambayashi, Y. Zhang (eds.) *1st International Conference on Web Information Systems Engineering*, vol. 1, pp. 183–189 (2000)