A metamodel to integrate business processes time perspective in BPMN 2.0

C. Arevalo [a,*], M.J. Escalona [a], I. Ramos [a], M. Domínguez-Muñoz [b]

[a] Department of Computer Languages and Systems, University of Seville, Avda. Reina Mercedes s/n, 41012 Seville, Spain
[b] Web Engineering and Early Testing (IWT2), Avda. Reina Mercedes s/n, 41012 Seville, Spain

Keywords:

BPM

BPMN

Time perspective

Metamodel

OCL

MDE

Legacy system

A B S T R A C T

Context: Business Process Management (BPM) is becoming a strategic advantage for organizations to streamline their operations. Most business experts are betting for OMG Business Process Model and Notation (BPMN) as de-facto standard (ISO/IEC 19510:2013) and selected technology to model processes. The temporal dimension underlies in any kind of process however, technicians need to shape this perspective that must also coexist with task control flow aspects, as well as resource and case perspectives. BPMN poorly gathers temporary rules. This is why there are contributions that extend the standard to cover such dimension. BPMN is mainly an imperative language. There are research contributions showing time constraints in BPMN, such as (i) BPMN patterns to express each rule with a combination of artifacts, thus these approaches increase the use of imperative BPMN style, and (ii) new decorators to capture time rules semantics giving clearer and simpler comprehensible specifications. Nevertheless, these extensions cannot yet be found in the present standard.
Objective: To define a time rule taxonomy easily found in most business processes and look for an approach that applies each rule with current BPMN 2.0 standard in a declarative way.
Method: A model-driven approach is used to propose a BPMN metamodel extension to address time-perspective.
Results: We look at a declarative approach where new time specifications may overlie the main control flow of a BPMN process. This proposal is totally supported with current BPMN standard, giving a BPMN metamodel extension with OCL constraints. We also use AQUA-WS as a software project case study which is planned and managed with MS Project. We illustrate business process extraction from project plans.
Conclusion: This paper suggests to handle business temporal rules with current BPMN standard, along with other business perspectives like resources and cases. This approach can be applied to reverse engineering processes from legacy databases.

1. Introduction

Today, most organizations take into consideration the Business Process Management paradigm (BPM), defined by relevant researchers as a strategic advantage [11,32,36,40,42] to support their operations. Most of these processes involve internal tasks as well as collaborative activities concerning other organizations. The Process Mining Manifesto, defined by Van der Aalst et al. [34], identifies different perspectives in a BPM system: *control, organizational, case and time*. They are well defined in [33,34] as follows: a) *control-perspective* focuses on the organization's activity, thus it allows depicting the control flow of a process; b) *organizational-perspective* is focused on information about resources, that is, which actors (e.g., people, systems, roles, and departments) are involved in activities and how they relate; c) *case-perspective* shows properties of cases; a case follows a specific path in the process depending on the originators working on it, but it can also be characterized by the values of the corresponding data elements; and d) *time-perspective* is concerned with time events; it should be possible to discover bottlenecks, measure service levels, monitor the utilization of resources, and predict the remaining processing time of running cases. We may also contemplate a *data-perspective* that involves the transitional data flowing among activities and data stores which, in turn, represent the of business data persistence layer.

Time dimension is present in all kinds of business processes. It appears in the form of business rules related, among others, to: a) single tasks aspects, such as duration or start and end events, for example; b) dependencies among activities that constrain its start and end events; c) deadlines of all kinds of activities; and

* Corresponding author. Fax: (+34)95-455-7139.
E-mail address: carlosarevalo@us.es (C. Arevalo).

d) inter-organizational process constraints that are concerned with exchanging messages and data.

Researchers have defined business process time dimension with different approaches, such as Timed Automata [6,7,35], Time Petri Nets [19,23,24], Time workflow nets [10], Timed activity graphs [12,13], BPMN [8,15–18,35] and other different techniques [20,21,37].

OMG Business Process Model and Notation (BPMN) [24] is becoming the de-facto standard and selected technology among other approaches for most business experts to model processes, because it offers the advantages of a graphical language [58], as well as simplicity, standardization (*ISO/IEC 19510:2013* [26]) and support for execution processes. It is defined by Bonnet et al. [4] as the leading standard for modeling business processes. BPMN provides business professionals with a notation that not only allows internal communication of business procedures, but also business-IT alignment and collaboration among business partners. However, some works, such as [67–70], conclude that most real models only handle a reduced set of symbols, due to the large number of artifacts in BPMN 2.0 and training cost to the average non-expert users. Zur Muehlen and Recker define three BPMN conformance levels [68]: Descriptive for simple, flowchart-like diagrams; Analytic for more sophisticated models that include event handling and messaging; and Common Executable with a focus on the model attributes that a Business Process Management System would expect.

Control-flow orders activities (single tasks, processes or subprocesses) execution, despite BPMN 2.0 limited capabilities to cope with time dimension. They may be executed sequentially or in parallel, what implicitly interrelates start and end events. Nevertheless, BPMN supports activities with multiple instances, looping and timer events, even though some other time rules are not directly supported. In consequence, many researchers agree that this standard is weak to model time dimension with rules that appear in many business processes.

BPMN is mainly an *imperative* language, which means that it depicts how the process has to run exactly. In addition, *declarative languages* only propose essential characteristics that constrain the execution of activities in a business process. Thus, *imperative approaches* are closer to production side while languages of *specification approaches* are closer to users or business experts' rule performance. Reijers et al. study these two approaches [29]. The aforementioned reasons lead us to focus on BPMN 2.0 for time dimension specification, since it is better overlaying new time rules as a complementary specification than reconstructing complete imperative diagrams with BPMN artifacts that may derive to models excessively overloaded. In view of this, we will propose a *specification approach.*

The next sections of this paper are organized as follows: Section 2 provides a motivating example that models a process for events organization. Section 3 summarizes the literature related to *time-perspective* in business process modeling, especially to BPMN time modeling. Then, Section 4 presents taxonomy for the time rules we will work with. Section 5 analyzes BPMN capabilities and weaknesses for time modeling. Subsequently, Section 6 presents a BPMN metamodel extension with Object Constraint Language (OCL) [27] formulation of our rules taxonomy. Section 7 describes a case study with the aim to evaluate the proposal. It uses the legacy database of MS Project [54] and a model-based approach to obtain business processes in software organizations. To end up, Section 8 includes a discussion of results and Section 9 states conclusions and future lines of work.

2. A motivating example

Below, Fig. 1 depicts an example of an organization's events management process. The first task, *"Preference analysis"*, allows the customer to analyze the company offers to fix preferences before *"Signing a contract"*, although both parts still have opportunities to make *"Contract amendments"*. Once signed, the next step deals with carrying out a global activity to *"Resource procurement"*, but eventually the customer may decide to *"Cancel the contract"*. *"Resource procurement"* activity is identified as an ad-hoc process composed of three tasks that can be executed in parallel: *"Book place"*, *"Book catering"* and *"Recruitment"*. After that, *"Event celebration"* is carried out in the planned date.

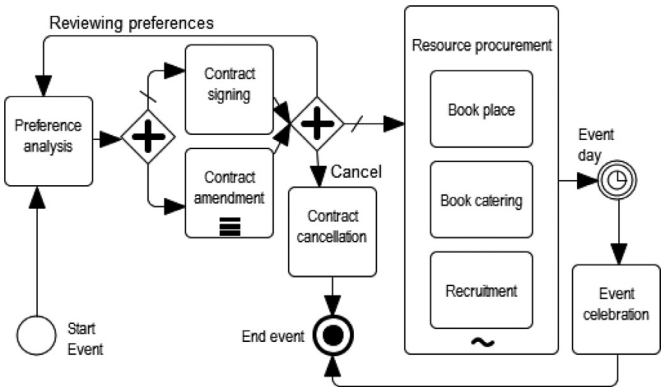


Fig. 1. A motivating example.

Table 1
Example rules.

#	Rule
r1	*Preference analysis lasts one or two days.*
r2	*Contract signing is carried out in one hour maximum.*
r3	*At most, three Contract amendments in a month are allowed.*
r4	*Event celebration will be celebrated on April 20th between 10:00am and 20:00pm*
r5	*Contract cancellation is not allowed in December.*
r6	*Book catering and Book place begin simultaneously.*
r7	*Recruitment and Book catering finish simultaneously.*
r8	*Resource procurement begins within thirty days from the Event celebration.*
r9	*Resource procurement should end within a week before Event celebration.*
r10	*If Book catering starts, then Contract amendment is not allowed.*
r11	*If Recruitment starts, then Contract cancellation is only possible 20 days before.*

Fig. 1 depicts this control flow, despite implicit time rules, because activity ordering use sequential or parallel modes. There is also an ad-hoc subprocess for *"Resource procurement"* that groups three activities that run concurrently, as well as time event rules and activities with multiple instances. Nevertheless, several business rules related to time are outlined in Table 1. These rules are not modeled in the previous BPMN diagram; they may be subject to certain changes (new time rules, deletions, modifications of values or types of rules, for instance) while the structure of the main control flow can be maintained.

Fig. 2 shows a new BPMN diagram that adds time rules of Table 1. Each one represents a specification (blue boxes) which experts would like BPMN may easily express.

At this point, some questions rise: If BPMN 2.0 [25,26,39] is mainly an imperative language, how can we define these rules in terms of this diagram? Must we modify and overload the control flow of the process to define these rules? Are there alternatives to specify these rules while maintaining the overall structure of the business process? Does BPMN have enough expressiveness to seriously address these issues? What ideas are researchers proposing to model time in BPMN? What advantages and disadvantages have their proposals?

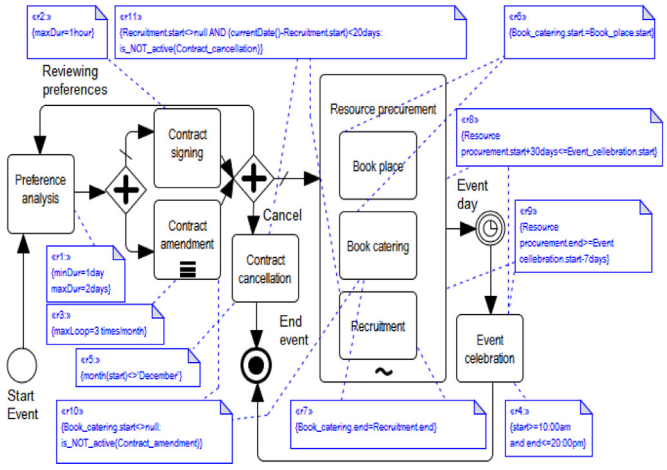


Fig. 2. Motivating example with time rules.

Next sections will analyze these issues and the related work for *time-perspective* specification in business processes. We will suggest an approach for overlaying time rules over imperative BPMN models that usually depict the *control flow perspective*. As future work concerns, we would like to integrate our approach into other BPM perspectives: *organizational, case or data*, among others.

3. Related work

The concept of business rule is well defined in Ross [63] and Baisley [64], although there are multiple rule classifications. Arevalo et al. [56] analyze two classifications: i) Wagner [66] provides a subdivision that is closer to technology, and ii) Jablonski and Bussler [65] include business time rules in their classification. Bibliography is full of research work dealing with business processes time dimension, so, it can be divided in two groups: authors who only work with time modeling or specification approaches and authors who propose verification methods of time rules in processes.

(i) Some of the former, use Time Petri Nets, for example: Huai et al. [19] and Makni et al. [23,24]. Some others prefer Timed Automata, such as Watahiki et al. [35] or Cheikhrouhou et al. [6,7]. Another subgroup utilize different techniques, for instance, Du et al. [10] with Time Workflow Nets, Kazhamiakin et al. [21] with Web Service Timed State Transition Systems, Kallel et al. [20] with XTUS-Automata, Wong and Gibbons [37] with Communicating Sequential Processes (CSP) and Eder and Tahamtan [12,13] with timed activity graphs. Lanz et al. [61] propose a set of time-patterns for process-aware information systems (PAIS) [33,41]. Finally, there is a subgroup that trust BPMN as a pillar for time extensions, such as Cheikhrouhou et al. [8], Flores and Sepúlveda [15], Gagné and Trudel [16–18] and Watahiki et al. [35].

(ii) Some of the latter, who verify time constraints in processes, use *UPPAAL* integrated tool environment for carrying out validation and verification of real-time systems modeled as timed automata networks, extended with data types (bounded integers and arrays, among others), such as Watahiki et al. [35], Du et al. [10] and Cheikhrouhou et al. [6,7]. Another subgroup of authors choose algorithms, like Eder and Tahamtan [12,13], and finally, some other authors like Huai et al. [19], Kazhamiakin et al. [21], Makni et al. [23,24] or Wong and Gibbons [37] prefer other different techniques. Researchers try to verify bottlenecks, deadlocks, infinite loops, dead tasks, maximal and minimal duration of processes, temporal consistency and conformance, and time compatibility, among other aspects. Cheikhrouhou et al. [5] propose a survey for identifying research challenges related to business process modeling *time- perspective*. They offer a suitable classification and analyze contributions of each work.

We will focus on time dimension specification on BPMN, so we will discuss the following authors' proposals.

Flores and Sepúlveda [15] work with time constraints included in PERT and GANTT diagrams for scheduling and control projects, thus they use MS Project as a case to model constructs in BPMN 1.2. They look at using BPMN patterns or constructs to specify every kind of time constraint for an activity or for precedence relationships between two activities. BPMN 1.2 does not yet support non-interrupting-intermediate events, in consequence they use signal artifacts to design their constructs. This approach respects the standard version capabilities and derives a BPMN diagram for a scheduled project. Nonetheless, the model seems to be overloaded with artifacts because of BPMN 1.2 limitations.

There is a main contribution in BPMN *time-perspective*, called *Time-BPMN* [18] proposed by Gagné and Trudel, who deserves complementary literature regarding time specification [16,17]. They work with Allen's interval algebra [1], identifying some weakness of BPMN 2.0, by taking control of start and end events and duration of activities, in addition to modeling time dependencies between pairs of activities. The authors propose extensions of BPMN 2.0 to include time constraints on an activity and model precedence dependencies between two activities based on Allen's algebra. They recommend new BPMN decorators attached to activities for capturing the semantics of durations and start and end events, so that they can model *fixed* and *flexible duration, fixed dates* such as *Must Start On (MFON)* and *Must Finish On (MFON)* and different kinds of *flexible start and end events* such as *As Soon As Possible (ASAP), As Last As Possible (ALAP), Not Earlier Than (NET)* and *Not Later Than (NLT)*. These date events change, only if the global schedule of the project is not affected (in PERT graphs: *critical-path* that sets the minimum project duration that must be reached). Besides, the authors suggest a new kind of association among these extended decorators, in order to model precedencies among activities to set time dependencies rules (constraints like *Start To Start (SS), Start To Finish (SF), Finish To Start (FS) and Finish To Finish (FF). Time-BPMN* is a reference proposal to model time in BPMN, because it facilitates the specification of business rules based on time and allows compact and clean business models, with less BPMN artifacts than other approaches focused on standard BPMN constructs. However, such models cannot be directly implemented for execution until the BPMN standard may support these extensions.

Cheikhrouhou et al. [8] extend the scope of time rules that can be modeled in BPMN 2.0. As in *Time-BPMN* [18], authors use decorator extensions to model time constraints and time dependencies among activities (tasks, processes and subprocesses). They include several rule categories: *i) Intra-activity temporal constraints, ii) Inter-activity temporal constraints, iii) Inter-process temporal constraints,* and *iv) Temporal constraints correlated with resource/data constraints.* These are further explained below:

(i) *Intra-activity temporal constraints.* They gather temporal constraints (TC) associated to one activity such as: a) duration, b) TC over cardinality, c) start/end, and d) intra-activity absence constraint.

(ii) *Inter-activity temporal constraints*. They group temporal constraints crossing the boundary of an activity within the Process Model such as: a) *Temporal dependencies*, and b) *Inter-activity absence constraint*.

(iii) *Inter-process temporal constraints.* This may occur in cases of collaborative or *Inter-Organizational Business Processes (IOBP)* between two organizations. There are temporal constraints

crossing the boundary of one process such as: *a) Exchanged temporal data* and *b) Deadline of message exchange.*

(iv) *Temporal constraints correlated with resource/data constraints.* The first kind of constraints concerns resources or group of resources that are granted to execute activities with time intervals where everyone performs its work. Correlation among time and data may define some activity durations depending on data functions that are relevant to each activity.

We are interested in expressing time rule taxonomy with BPMN, then we will take into account the contributions of the last group of authors with the aim to illustrate it in the next section.

Legacy information systems (also called legacy systems) are software systems that often last many years in an organization and that are the core of its business. These systems are highly complex and they include a lot of functionality and code; they usually mix different technologies and their maintenance is difficult and costly. The substitution of a legacy system by a modern system is not an easy task [53]. They are still alive, therefore managers delay making the decision of changing them.

The Model-Driven Engineering (MDE) paradigm is maturing in the software field. MDE is characterized by handling different levels of abstraction to model and metamodel describing platforms. Consequently, there are: i) Computer Independent Metamodel (CIM), Platform Independent Metamodel (PIM) and Platform Specific Metamodel (PSM), and ii) languages that allow mapping among models: model-to-model (M2M) or model-to-text (M2T). OMG Modern Driven Architecture (MDA) [48,49] is the best known MDE initiative, characterized by the Meta-Object-Facility (MOF) to describe models, metamodels and the recursive MOF Meta-metamodel.

To address the problem of legacy systems modernization OMG proposes Architecture-Driven Modernization (ADM) [38], providing cycles of reverse engineering and forward engineering focused on extracting knowledge from legacy systems using the Abstract Syntax Tree Metamodel (ASTM) [50] and Knowledge Discovery Metamodel (KDM). ASTM may be generic (GASTM) or specific (SASTM) depending on whether they describe artifacts on a group of platforms or just a case of platform, respectively. If the primary device used for modernizing a legacy system is its database, it is called legacy database modernization. There are a lot of studies in this field that are described in Arevalo et al. [2], which propose an approach to extract knowledge from relational databases, so that rules hardcoded as constraints and triggers on tables are defined as ECA-rules related to classes.

4. Time constraints

It is necessary to define the time rule scope we will consider in our proposal. We focus on *activities* in the business process field that run in the same organization. Therefore, we firstly exclude some time rules, like *IOBPs* [5,8], that could be included in future work. An *activity* may be a single task, a *process* or a *subprocess*, but, in general, it must have fixed or flexible duration.

Fig. 3 shows the rule taxonomy we will work with. We can distinguish: *i) Temporal Constraints* related to individual activities and *ii) Temporal Dependencies* that constrain relationships between two activities.

(i) *Temporal constraints* may express:
  (a) *Duration* of an activity that may be *fixed* or *variable* in a time interval (between minimum and maximum duration).
  (b) *Inflexible Time Constraints MSON* and *MFON* that specify a fixed start and finish date for the activity.

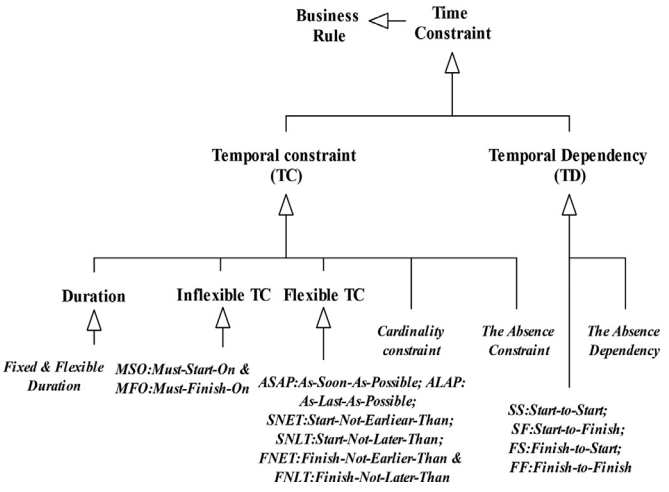


Fig. 3. Rule taxonomy.

  (c) *Flexible Time Constraints: ASAP (SASAP, FASAP), ALAP (SALAP, FALAP)*, and *NET(SNET, FNET), NLT (SNLT, FNLT)* that allow moving the start and finish date whenever all predecessors and successor activities are not affected in the global schedule of the process. These constraints are closely related to *Critical-Path-Method (CPM)*, by Kelley and Walker [22], which is a mathematical method for calculating the minimum duration of an activity graph, where each activity is a node and the arcs represent their precedence relationships. *CPM* defines a subgraph composed of *critical activities* called critical-*path*; these activities do not allow gaps in their start or finish dates without altering the minimum duration of the graph. Activities that do not belong to the *critical-path-subgraph* have *slack time* in their duration, so that they can be moved without changing the length or duration of the *critical-path-subgraph. CPM* scheduler must have the capability for automatically move these activity events, then CPM iterates in a sensibility analysis until the minimum duration of the process *P* is calculated. Thus, a BPMN process *P* may be depicted as a graph *G* and Eq. (1) is satisfied:

$$G \equiv \{Ai\}; G' \equiv \left\{A'i\right\}; G' \subseteq G;$$

$$dur(P) = \min\left(dur(G)\right) = dur\left(G'\right) = \sum_{A'i \in G'} dur\left(A'i\right) \quad (1)$$

  In the equation above *G* stands for the complete graph of activities A_i, each one with a duration $dur(A_i)$ and *G'* represents the subgraph of activities A'_i that is calculated with the *CPM* method. Users fix *upper* and *lower* bound dates in the case of *NET* and *NLT*, whereas in *ASAP* and *ALAP*, it is the scheduler that calculates such bounds by means of the *CPM* method iterations.
  If we used CPM method, then we could reach the minimum duration for the process, as in Eq. (1), otherwise the process would be longer than this minimum: $dur(P) \geq min(dur(G))$.
  (d) A *Cardinality Constraint*, which limits iterations of a loop activity related to its duration.
  (e) *An Absence Constraint*, which states that an activity cannot be executed; it may either be skipped permanently or activated only in a time interval.

(ii) *Temporal dependencies* apply to precedence relationships, thus:
  (a) *Absence Dependency* constrains that an activity cannot be executed, if the precedent is already carried out or it

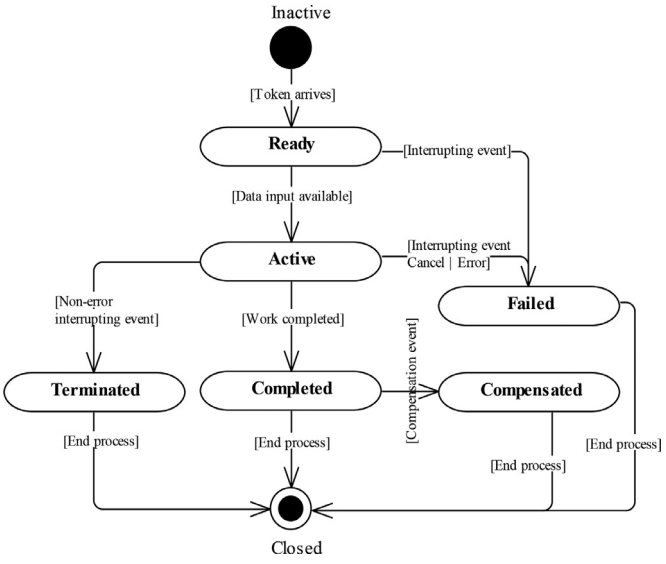


Fig. 4. BPMN activity lifecycle.

is active. Thus, like *Absence Constraint*, it may either be skipped or activated only in a time interval.

(b) *SS, SF, FS* and *FF* are typical dependencies among *start* and *end* events of an activity and its predecessor.

5. BPMN time constraint limitations

BPMN 2.0 [25,26,39] allows defining temporal dimension. In this sense, a process is composed of artifacts where *activity* is the main type of artifact, representing a state and a job to perform. An activity needs a finite time interval to be executed. In addition there are control flow, information flow and events that depend on a particular time space. However, many researchers agree that BPMN is weak to express temporal dimension (see Section 3). This is because there are complex time rules, either related to a single activity or among sets of activities. It is worth pointing out that BPMN does not support all kinds of time rules included in Section 4.

As long as the analysis of a BPMN activity lifecycle concerns, it must be pointed out that Fig. 4 shows a Unified Modeling Language (UML) [28] state-machine, that means a simplified version included in the BPMN standard [25,26,39]. Weske [36] goes deeper into the analysis of an activity lifecycle, adding more states and transitions, such as a) *disabled* and transitions among *ready* state, b) *suspended* and transitions among *active* state and c) *skipped* state to avoid execution without doing the work. Svatos [31] shows that contemporary process modeling languages, including BPMN, only cover the defined general lifecycle of an activity partially.

This could be one of the main reasons of BPMN weakness regarding time specification. Another reason is that precedence relationships considered by Allen's algebra [1] are not directly supported. Furthermore, there are no mechanisms to control the work that must be executed after triggering the start event of an activity. However, BPMN 2.0 introduces non-interrupting events attached to activities. That enables modeling new control flow from an activity while it is still active, helping design new BPMN patterns or constructs to express time rules. In BPMN, every *activity A* has a start *s(A)* and an end event *e(A)*, then the activity duration *dur(A)* verifies the Eq. (2) for time interval execution of an item:

$$dur(A) = e(A) - s(A) \tag{2}$$

If we aim to introduce *flexible duration* and *flexible start and finish* for an activity, we need to add more information to the BPMN metamodel, like minimum and maximum duration *minDur(A), maxDur(A),* and *due-dates* or *deadlines* of activities. Consequently, activities may be scheduled verifying the Eq. (3):

$$\min Dur(A) \leq (e(A) - s(A)) \leq \max Dur(A) \tag{3}$$

We could design BPMN 2.0 constructs (like Flores and Sepúlveda [15] with BPMN 1.2) to express time constraints and dependencies, with non-interrupting events and other capabilities of the new standard. Table 2 and Table 3 show possible models for each time rule of taxonomy in Fig. 3.

Table 2(a) shows constructs to model *fixed* and *flexible start* and *end* events together with *fixed* and *flexible duration* of an activity. Firstly, for *flexible duration*, the scheduler may manage *ASAP* (*SASAP, FASAP*), that is the default option, if not explicitly expressed (the diagram depicts these default control flows), or it may manage *ALAP* (*SALAP, FALAP*), that means alternative flows. In both cases, predecessor and successor activities must not be affected when they are scheduled. Later, regarding *duration constraints* it must be stated that, if *duration is flexible,* the previous Eq. (2) is valid, whereas if it is *fixed*, when *A* reaches its duration, the end fires out (with a timer event) and the Eq. (4) is satisfied:

$$\min Dur(A) = (e(A) - s(A)) = \max Dur(A) \tag{4}$$

Table 2(b) models the cardinality constraint to limit the loop times *(N)* that an activity $A\{A_i\}$ may execute; $\{A_i\}$ constitutes the set of runtime instances. It depicts a default flow, when the activity work is totally fulfilled. An alternative flow fires when the limit is reached, despite there is still work to do. The latter is drawn with an attached intermediate event, a business rule that fires the end of the activity.

In both cases, Eqs. (5a) and (5b) must be satisfied; (5a) constrains that *A* duration is obtained by adding until *N* times) durations of runtime instances $\{A_i\}$, and (5b) constrains that this duration must appear among *A* duration limits:

$$e(A) - s(A) = dur(A) = \sum_{i=1}^{N} (e(Ai) - s(Ai)) \tag{5a}$$

$$\min Dur(A) \leq \left(\sum_{i=1}^{N} (e(Ai) - s(Ai)) \right) \leq \max Dur(A) \tag{5b}$$

Table 2(c) depicts inflexible start *MSON* and end *MFON* constraints. *MSON* is a due-date that fires the activity and *MFON* is a due-date with throws and *intermediate-interrupting-event* for the activity, even though its work is uncompleted.

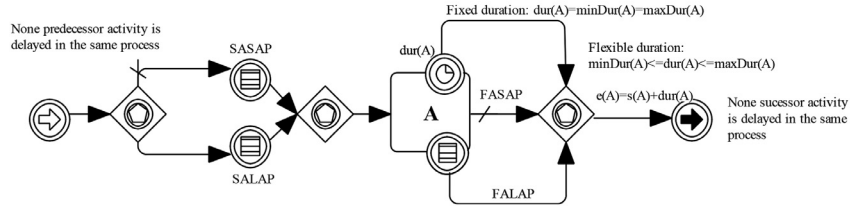
Table 2(d) shows flexible TC for activity start and end events. Furthermore, due-dates *SNET* and *SNLT* act as intermediate events that fire the *activity start.* The model includes the following elements for rules concerning finishing the activity: a) an attached *intermediate-interrupting-event* for due-date *FNLT* that finishes the activity, although the work has not concluded yet, and b) a due-date *FNET* depicted as an attached *intermediate-non-interrupting-event* that expresses that the *end of the activity* must be later than the reference date, when all the work has finished. *FNET* does not fire the activity end.

Table 2(e) models an absence unconditional constraint that expresses the *activity is skipped*, since the normal flow should be to execute the activity. Table 2(e) shows the situation taking place when absence is conditioned to time interval $[t_1, t_2]$, thus activity may be active in this time interval and in absence state, if running time is out of the same interval.

Table 3(a) depicts a *Finish to Start (FS)* dependency between activities A_i and A_j that are sequentially executed with a *lead or lag* between end event of A_i and firing of the start event of A_j.

Table 3(b) models a *Start to Start (SS)* dependency between activities A_i and A_j that run concurrently. The start event of A_j is fired when A_i starts and after a *lead or lag*.

Table 2
BPMN constructs for time constraints.

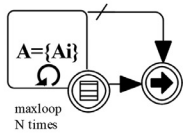


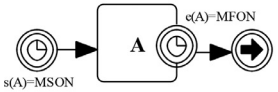
(a) Fixed and Flexible Start and End, Fixed and flexible duration

(b) Cardinality constraint

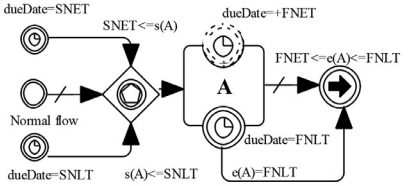
$$e(A)-s(A)=dur(A)=\sum_{i=1}^{N}\left(e(A_i)-s(A_i)\right)$$

$$\min Dur(A)\le\left(\sum_{i=1}^{N}\left(e(A_i)-s(A_i)\right)\right)\le\max Dur(A)$$



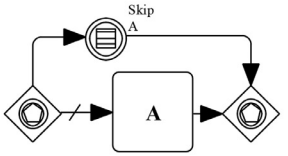


(c) Inflexible start and end



(d) Flexible TC for start and end

(e) Absence unconditional constraint *(ABS-U)*



(f) Absence constraint in a time interval *(ABS-C)*

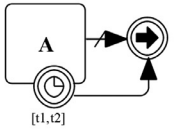
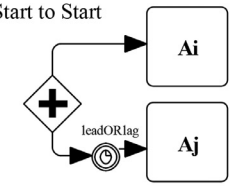


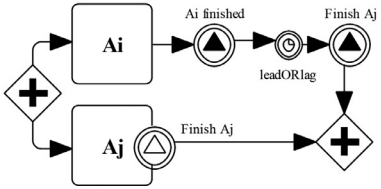
Table 3
BPMN constructs for time dependencies.

(a) FS: Finish to Start

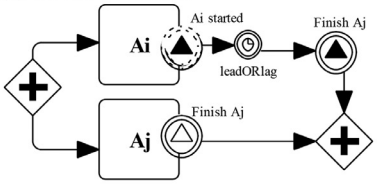
(b) SS: Start to Start



(c) FF: Finish to Finish



(d) SF: Start to Finish



(e) Absence dependency *(ABS-U, ABS-C)*

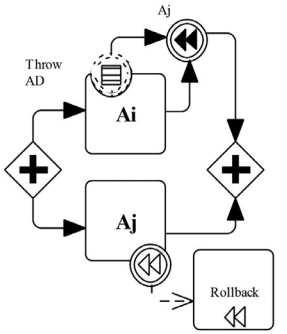


Table 3(c) shows a construct for *Finish to Finish (FF)* dependency between activities A_i and A_j. Even though they run in parallel, it must be known that A_i finishes first, and then, after a *lead or lag*, A_j finishes as well. This constructs uses signal events for capturing this semantics, where A_i *throws* a signal event and A_j *catches* it to finish.

Table 3(d) illustrates a *Start to Finish (SF)* dependency between activities A_i and A_j. Their instances run in parallel too, that means, an attached non-interrupting signal event takes place to highlight that A_i has started. Thus, A_i keeps active until its work has finished. Moreover, after a *lead or lag*, a new signal event is thrown. A_j catches this signal event to finish.

Finally, Table 3(e) consists in a construct for the *Absence dependency* between activities A_i preceding A_j that run in parallel. If A_i verifies a rule (it has usually started or finished), there exists an absence dependency from A_i to A_j. It **implies** that A_j cannot be active, so it must be skipped, either permanently interrupted or just interrupted during a specified time interval $[t_1, t_2]$. We introduce a construct that throws a *compensation event* when identifying the nature of the dependency. In consequence, A_j catches this compensation event, if necessary, *it* may be rolled back with an associated task. As it is well known, some conditions must be met for A_j compensation in relation to A_i may happen, since they belong to the same parent process that still remains active.

We conclude that models obtained will be overloaded with BPMN artifacts, even though we can use these patterns to model time constraints and the fact that BPMN 2.0 [25,26,39] is more powerful than BPMN 1.2. In contrast, works extending BPMN 2.0 with decorators to introduce time dimension, like *Time-BPMN* by Gagné and Trudel [18] and, among others, some extensions like those included in the work by Cheikhrouhou et al. [8]. Both of them give very good comprehensibility specifications, although we should wait until BPMN will be able to have this semantics. Table 4 shows a survey of rules soported by these researchers with some variations linked to the icons they use.

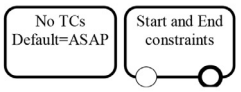
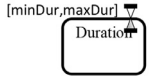
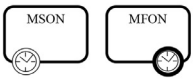
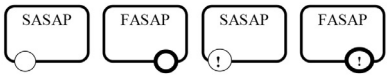
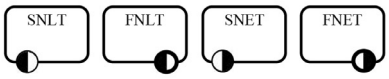
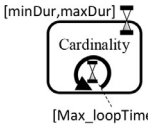
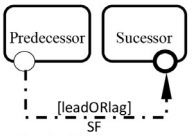
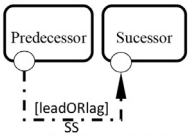
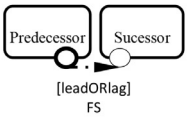
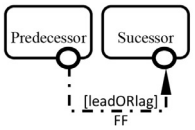
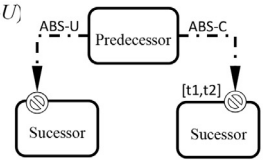
Then, the starting point is rule taxonomy (Section 4, Fig. 3) and the feasibility of actual BPMN 2.0 to define the semantics of each rule with a MDE-based approach [46,47] presented in the next section. After that, business experts can use custom facilities of modeling tools (i.e. Enterprise Architect from Sparx Systems or Activity open source BPMN modeler, among others) to adjust the graphical interfaces of modeled processes with figures, icons or stereotypes.

6. BPMN metamodel to integrate time dimension

BPMN 2.0 [25,26,39] has a metamodel for activities. This metamodel is very similar to the UML metamodel for activity diagrams [28]. There are authors who have worked with this metamodel and OCL [27] rules specification, such as: a) Awad et al. [3], who model BPM *resource* and *case perspective*. They work with *Workflow Resource Patterns (WRP)*, so that each resource allocation pattern is expressed as an OCL constraint over the metamodel proposed; b) Stroppi et al. [30], who enhance previous work; and c) Arevalo et al. [2], who propose a framework to derive BPMN models from legacy systems focusing on time and resource perspectives.

We are interested in these proposals because, as a future work, we would like to integrate more business perspectives into our approach to depict *time-perspective*. It seems to be easier since we share the same architectural aspects, therefore we will use a MDE-based approach [46,47] together with a BPMN metamodel (at CIM level) and OCL constraints to capture semantics of our time rules taxonomy (Fig. 3). We would also add a new package named *Extension*, which contains classes and enumerations to extend the standard. For this purpose, we will use a simplified view of this metamodel that is related to activities. Next subsections will describe our proposal in detail.

Table 4
Time decorators in BPMN 2.0.

#	Rule	(a)	(b)
TC, TD	 (a) TC and TD over start and end events	✓	✓
TC	 (b) Flexible duration		✓
TC	 (c) Inflexible start and end	✓	✓
	 (d) Flexible start and end	✓	✓
TC		✓	✓
TC	 (e) Cardinality constraint		✓
TC	(f) Absence unconditional (*ABS-U*)  (g) Absence conditional (*ABS-C*)		✓
TD	(h) Start to Finish  (i) Start to Start 	✓	✓
TD	(j) Finish to Start  (k) Finish to Finish 	✓	✓
TD	(l) Absence Unconditional (*ABS-U*)  (m) Absence Conditional (ABS-C)		✓

[a] Time-BPMN by Gagné & Trudel [18]
[b] Cheikhrouhou et al. [8]

6.1. BPMN metamodel extensions

We select classes from the BPMN metamodel, particularly *Activity, Diagram, Lane, Pool, Process, Subprocess and Task*. Fig. 5 shows the package structure, the standard *BPMN* package and the Extension package. This last includes *Temporal_Constraint* and *Temporal_Dependency* classes to specify each constraint, as well as *ETC_type* enumeration for each temporal constraint and *ETD_type*

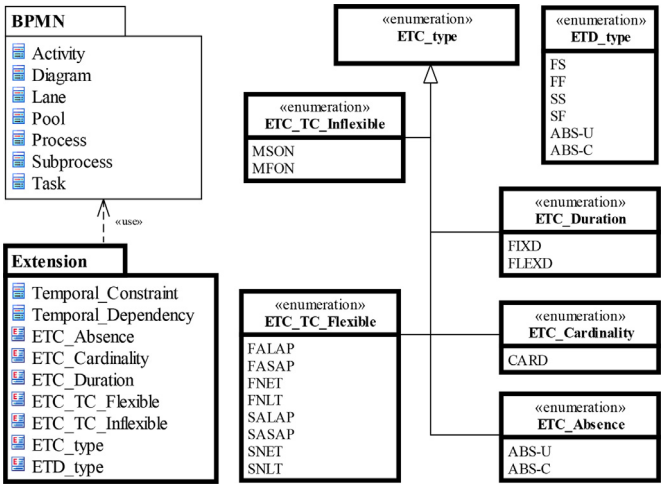


Fig. 5. BPMN packages and enumerations.

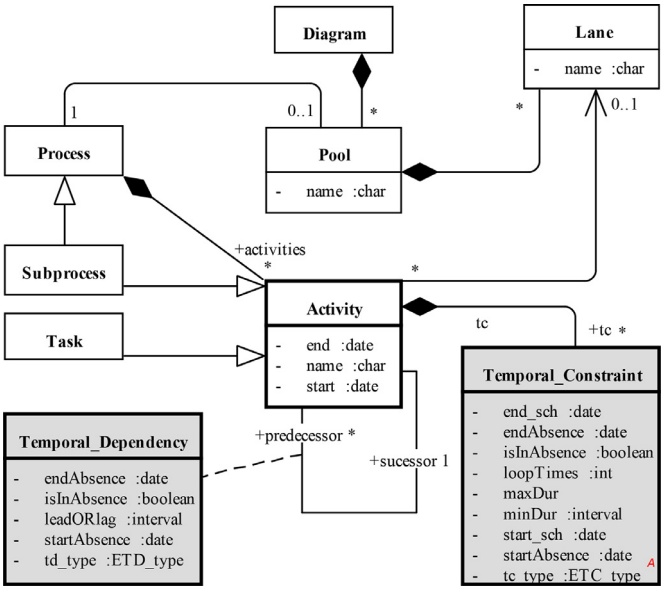


Fig. 6. BPMN metamodel extensions.

enumeration for each temporal dependency. *ETC_type* generalizes enumerations for specific time constraints: *ETC_TC_Inflexible* for fixed start and end events; *ETC_Duration* for fixed and flexible duration; *ETC_TC_Flexible* for *ASAP, ALAP, NET* and *NLT* rules and *ETC_Cardinality* and *ETC_Absence* for two kinds of absence constraints: unconditional *(ABS-U)* and conditional in an interval *(ABS-C)*. Nonetheless, *ETD_type* includes *FS, FF, SS, SF* and two kinds of absence dependency *(ABS-U, ABS-C)*. We have drawn thicker border for package and enumerations that are added to the standard metamodel.

Fig. 6 below shows the selected classes, associations among them and metamodeling extension details. *Temporal_Constraint* and *Temporal_Dependency* are represented with shaded background and thicker border, as follows:

(i) *Activity* is the core class of this metamodel view. It may be a single *Task* or a *Subprocess*, which in turn, is also a business *Process*. At the same time, *processes* are composed of a set of *+activities* and they *are* usually modeled in a *Pool*, where *Activities* fit a *Lane*. If the modeler uses pools, a *Diagram* comprises *Pools* and *Lanes*.

(ii) An *Activity* has a *name, start* and *end* dates.

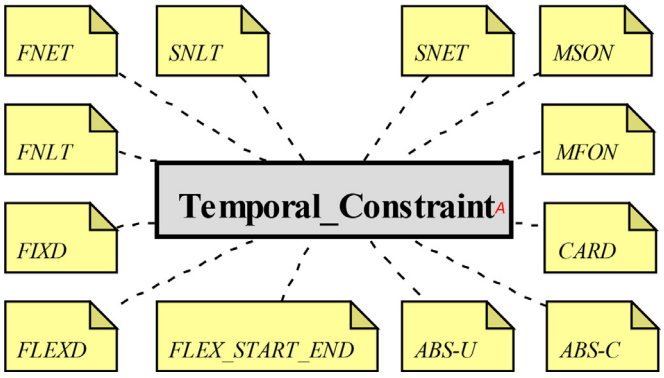


Fig. 7. OCL time constraints.

(iii) An *Activity* may have a set of temporal constraints, which are expressed with *Temporal_Constraint* class through *tc* composition association. This class contains scheduled dates, *start_sch* and *end_sch*; durations, *minDur* and *maxDur; loopTimes,* which is a maximum limit; attributes for absence constraints *isInAbsence;* and time interval dates, *startAbsence* and *endAbsence*.

(iv) Two activities with a precedence relation share one instance of association class *Temporal_Dependency*. This class lets us know the roles of *+predecessor* and *+successor* with a cardinality of *(1:n)*. The class has also properties: *td_type* that classifies each rule with respect to *ETD_Type* enumeration, and *leadORlag,* which refers to the time interval among the related events of both activities. We use attributes *isInAbsence* for *Absence dependencies* to identify this state, and *startAbsence* and *endAbsence* to determine the situation only in a time interval. However, if absence is unconditional, both dates are null.

6.2. OCL rules specification

OCL is a pure specification language without *side-effects* [27], so when an OCL expression is evaluated, it simply returns a value without changing anything in the model. OCL expressions are constraints over classes of our BPMN metamodel.

We code OCL constraints for time rules, but other BPM perspectives like resource allocation or case rules [3,30] may be easily integrated. We can also expand the metamodel scope to incorporate more complex time rules: for example, for collaborative processes (IOBP) [5,8].

There are OCL specifications with two contexts classes: *Temporal_Constraint* and *Temporal_Dependency,* which express both groups of rules. Fig. 7 depicts OCL invariants linked to the former. Each rule uses OCL navigation facilities from the context in order to apply them to other classes and write the constraint. Firstly, Fig. 7 includes temporal constraint rules for fixed events *MSON* and *MFON,* for constrained start and end events*: NET (SNET, FNET)* and *NLT (SNLT, FNLT),* and finally for flexible start and end events (*FLEX_START_END): ASAP (SASAP, FASAP)* and *ALAP (SALAP, FALAP),* where *NET* and *NLT* need to be grouped once again as we will later deeply study. Finally the figure includes duration constraints: *FLEXD* and *FIXD*, Rule *CARD* and also *ABS* invariants.

Fig. 8 shows a set of invariants for time dependencies linked to UML class Temporal_Dependency, rules SS, SF, FS, FF and Absence Dependency (unconditional and conditional).

Tables 5 and 6 describe each rule related to its context class, as follows:

(i) We have the definition of rules using Temporal Constraint in Table 5:

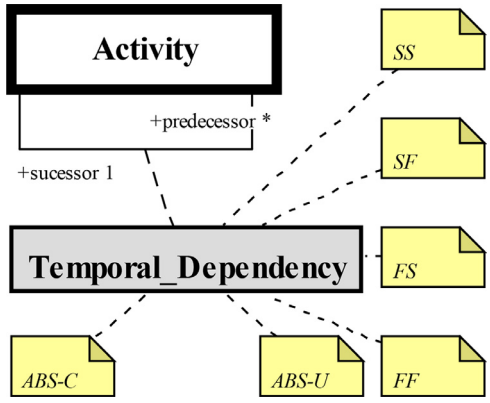


Fig. 8. OCL time dependencies.

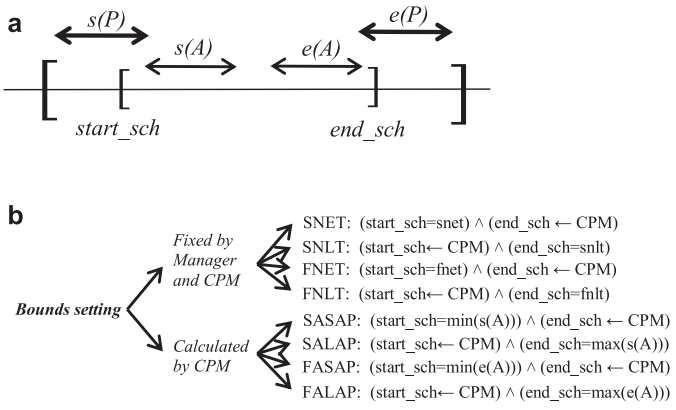


Fig. 9. (a) OCL lower and upper bounds for flexible start and end. (b) OCL lower & upper bounds for flexible start and end.

FIXD. It constrains the duration of the activity that is fixed for runtime and scheduled planning. It uses *minDur* and *maxDur* attributes to set *(minDur=maxDur)*.

FLEXD. It is similar to previous invariant, although runtime and scheduled durations must be in the interval [*minDur, maxDur*].

FLEX_Start_End. If we aim to reach the minimum duration of the process, as Eq. (1) shows, then, we must take into account the *CPM* Method [22] for flexible start and end constraints, otherwise the process will be longer than the minimum duration of its activity graph: $dur(P) \geq min(dur(G))$. If *CPM* is supported, then the scheduler can move activity events in a certain time interval; Fig. 9(a) shows that start *s(A)* and end events *e(A)* can be moved for an activity between *start_sch lower bound* and *end_sch upper bound*. Nonetheless, the duration of parent *P* process of activity *A* is establihed with *CPM* and Eq. (1) to minimize duration *dur(P)*, where $dur(P)=(e(P)-s(P))$ stands for the duration of the *P critical-path*. We must set these bounds to write OCL invariants in each case. Fig. 9(b) shows the rules to assing these bounds.

As SNET, SNLT, FNET and FNLT concern, it is the manager who may manually fixes one of the bounds that are assigned to *start_sch* or *end_sch* attributes in the BPMN metamodel. After that, at *CPM* runtime, the opposite bound acts as an independent variable in interations taking into account the *slack time* for the activity (we denote these offered calculations offered automatically with **←CPM** in Fig. 9(b)). With regard to *ASAP* and *ALAP*, both bounds are automatically calculated by the CPM scheduler (**←CPM**) to maintain the duration of *P critical-path* [$dur(P'_{CP})$], by considering *slack time* as well. We assume that

Table 5
OCL time constraints.

# tr	OCL rule specification
FIXD	*Context Temporal_constraint inv: (self.tc_type='FIXD') implies self.tc→select ((self.end_sch - self.start_sch) =self.minDur and self.minDur=self.maxDur)→notEmpty()*
	inv: (self.tc_type='FIXD' and not (self.start.OclIsUndefined() or self.start.OclIsUndefined())) implies self.tc→select ((self.end-self.start)=self.minDur)→notEmpty()
FLEXD	*Context Temporal_constraint inv:(self.tc_type='FLEXD') implies self.tc→select((self.end_sch-self.start_sch)>=self.minDur and (self.end_sch-self.start_sch)<=self.maxDur)→notEmpty()*
	inv: (self.tc_type='FLEXD' and not (self.start.OclIsUndefined() or self.start.OclIsUndefined())) implies self.tc→select((self.end-self.start)>=self.minDur and (self.end-self.start)<=self.maxDur)→notEmpty()
FLEX_ Start_ End	*Context Temporal_constraint: def: min_Dur_P:date = self→select ((self.tc.process.end - self.tc.process.start))→min() –min_Dur_P calculated by CPM scheduler when moving start —and end of activities between lower and upper bounds –start_sch and end_sch*
	inv: ((Set{'SASAP','SALAP','FASAP','FALAP', 'SNET','SNLT','FNET','FNLT'})→includes(self.tc_type) and self.tc.start>= self.start_sch and self.tc.end<= self.end_sch) implies self→select ((self.tc.process.end - self.tc.process.start))=min_Dur_P
CARD	*Context Temporal_constraint inv:(self.tc_type='CARD') implies not (self.start.OclIsUndefined() or self.start.OclIsUndefined()) and self.tc→select ((self.start_sch+self.loopTimes•self.minDur<=self.end_sch) and (self.end_sch<=self.start_sch+self.loopTimes•self.maxDur))→notEmpty()*
	inv: (self.tc_type='CARD' and not (self.start.OclIsUndefined() or self.start.OclIsUndefined())) implies self.tc→select ((self.start+self.loopTimes•self.minDur<=self.end) and (self.end<=self.start+self.loopTimes•self.maxDur))→notEmpty()
MSON	*Context Temporal_constraint inv:(self.tc_type='MSON') implies self.tc→select(self.start_sch.OclIsUndefined())→isEmpty()*
	inv: (self.tc_type='MSON' and not elf.start.OclIsUndefined()) implies self.tc→select(self.start=self.start_sch →notEmpty()
MFON	*Context Temporal_constraint inv:(self.tc_type='MFON') implies self.tc→select(self.end_sch.OclIsUndefined())→isEmpty()*
	inv: (self.tc_type='MFON') implies self.tc→select(end=self.end_sch)→notEmpty()
SNET	*Context Temporal_constraint inv:(self.tc_type='SNET') implies self.tc)→select(self.start_sch.OclIsUndefined()))→isEmpty()*
	inv: (self.tc_type='SNET') implies self.tc→select(self.start >= self.start_sch))→notEmpty()
SNLT	*Context Temporal_constraint inv:(self.tc_type='SNLT') implies self.tc→select(self.end_sch.OclIsUndefined())→isEmpty()*
	inv: (self.tc_type='SNLT') implies self.tc→select(self.start <= self.end_sch)→notEmpty()
FNET	*Context Temporal_constraint inv:inv: (self.tc_type='FNET') implies self.tc→select(self.start_sch.OclIsUndefined())→isEmpty()*
	inv: (self.tc_type='FNET') implies self.tc→select(self.end <= self.start_sch)→notEmpty()
FNLT	*Context Temporal_constraint inv:(self.tc_type='FNLT') implies self.tc→select(self.end_sch.OclIsUndefined())→isEmpty()*
	inv: (self.tc_type='FNLT') implies self.tc→select(self.end <= self.end_sch)→notEmpty()

(*continued on next page*)

Table 5 (*continued*)

# tr	OCL rule specification
ABS	*Context Temporal_constraint inv: inv: (self.tc_type='ABS-U' and self.startAbsence.OclIsUndefined() and self.endAbsence.OclIsUndefined()) implies self→select(self.isInAbsence=true)→notEmpty()*
	inv: (self.tc_type='ABS-C' and self.startAbsence.OclIsUndefined()) and (not (self.tc.start.OclIsUndefined()) or self.startAbsence<=self.tc.start)) and (not (self.tc.end.OclIsUndefined()) or (self.tc.end<=self.endAbsence)) implies self→select(self.isInAbsence=true)→notEmpty()

Table 6
OCL time dependencies.

# tr	OCL rule specification
FS	*Context Temporal_Dependency inv: self.td_type='FS' implies self.predecessor→select(P| not (P.end_sch.OclIsUndefined() or self.sucessor.start_sch.OclIsUndefined()) and self.sucessor.start_sch=P.end_sch+self.leadORlag)→notEmpty()*
	inv: (self.td_type='FS' and not (self.predecessor.end.OclIsUndefined() or self.sucessor.start.OclIsUndefined()) implies self.predecessor→select(P| self.sucessor.start=P.end+self.leadORlag)→notEmpty()
FF	*Context Temporal_Dependency inv: self.td_type='FF' implies self.predecessor→select(P| not (P.end_sch.OclIsUndefined() or self.sucessor.end_sch.OclIsUndefined()) and self.sucessor.end_sch=P.end_sch+self.leadORlag)→notEmpty()*
	inv: (self.td_type='FF' and not (self.predecessor.end.OclIsUndefined() or self.sucessor.end.OclIsUndefined()) implies self.predecessor→select(P| self.sucessor.end=P.end+self.leadORlag)→notEmpty()
SF	*Context Temporal_Dependency inv: self.td_type='SF' implies self.predecessor→select(P| not (P.start_sch.OclIsUndefined() or self.sucessor.end_sch.OclIsUndefined()) and self.sucessor.end_sch=P.start_sch+self.leadORlag)→notEmpty()*
	inv: (self.td_type='SF' and not (self.predecessor.start.OclIsUndefined() or self.sucessor.end.OclIsUndefined()) implies self.predecessor→select(P| self.sucessor.end=P.start+self.leadORlag)→notEmpty()
SS	*Context Temporal_Dependency inv: self.td_type='SS' implies self.predecessor→select(P| not (P.start_sch.OclIsUndefined() or self.sucessor.start_sch.OclIsUndefined()) and self.sucessor.start_sch=P.start_sch+self.leadORlag)→notEmpty()*
	inv: (self.td_type='SF' and not (self.predecessor.start.OclIsUndefined() or self.sucessor.start.OclIsUndefined()) implies self.predecessor→select(P| self.sucessor.start=P.start+self.leadORlag)→notEmpty()
ABS	*Context Temporal_Dependency inv: (self.td_type='ABS-U' and self.startAbsence.OclIsUndefined() and endAbsence.OclIsUndefined() and self.predecessor→select(P| (P.start.OclIsUndefined())→notEmpty()) implies self→select(self.isInAbsence=true) →notEmpty().*
	inv: (self.td_type='ABS-C' and self.startAbsence.OclIsUndefined() and self.predecessor→select(P | (not (P.start.OclIsUndefined() and self.startAbsence.OclIsUndefined()) or (P.start+self.leadORlag<=self.startAbsence)) and (not (P.end.OclIsUndefined() and self.endAbsence.OclIsUndefined()) or (P.end+self.leadORlag<=self.endAbsence)))→notEmpty ()) implies select(self.isInAbsence=true) →notEmpty()

CPM scheduler manages assingments in Fig. 9(b) and it also establishes *dur(P)* and *dur(P'$_{CP}$)* for *critical path.* As already mentioned, reference dates *start_sch* and *end_sch* can be moved in iterations, only if activities have *slack time* and they have not finished. Now, we can write OCL invariants that preserve *P duration* in each case with a compact formulation.

CARD. The attribute *loopTimes* refers to the iteration limit between durations of each instance and duration of the global *Activity*. It may be expressed for runtime and scheduled dates.

MSON, MFON. These are invariants for fixed start and finish events of an activity. They must occur in scheduled dates *start_sch* and *end_sch*.

SNET, SNLT, FNET, FNLT. In this case, the invariants express constraints over *due-dates* that act as bounds for start and end events. These kinds of constraints are involved in previous FLEX_Start_End invariant because they imply flexible start or end of an activity.

ABS (ABS-U, ABS-C). The attribute *isInAbsence* refers to the absence state of an activity. If such state is permanent then (*isInAbsence=true)* unconditionally becomes (*ABS-U*), but if the activity is in this state only for a time interval (the absence state is conditioned (*ABS-C*) to timestamps or dates), we need to formulate the invariant with attributes *startAbsence* and *endAbsence duedates* to constrain start and end events.

(ii) Table 6 shows that each temporal dependency on its association class *Temporal Dependency* has two roles: predecessor and successor, for pairs of activities that share this kind of constraints. Thus, we have:

FS, FF, SF, SS. All of them refer to precedence among events. Invariants express that synchronization rules involve a possible *lead or lag* (attribute *leadORlag*). Navigation over *predecessor* and *successor* roles allows writing compact expressions for these rules.

ABS (ABS-U, ABS-C). These invariants describe dependencies between *predecessor* and *successor*, so that the execution of *predecessor* causes that *successor* becomes in *absence* (*ABS-U*: this is the unique condition to be met). The second version of invariant puts forward that the *successor* becomes absent, if its events are among *startAbsence* and *endAbsence.* In both cases, a *lead or lag (leadORlag)* are allowed among the events of the related activities.

6.3. Motivating example scenario

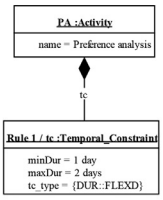
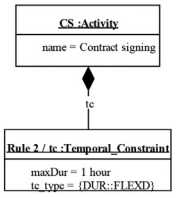
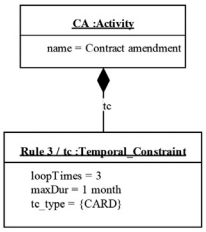
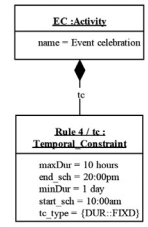
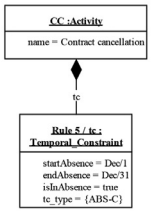
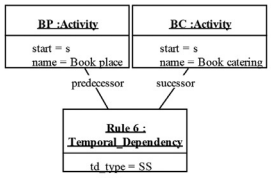
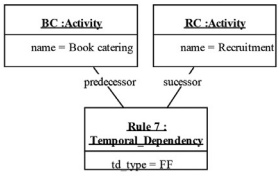
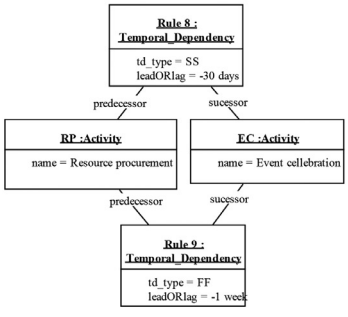
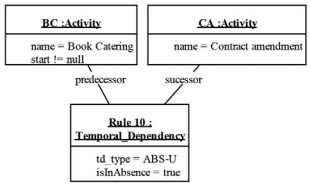
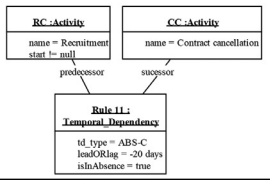
We have revised our proposal by means of a motivating example in Section 2. Table 7 describes the proposed rules. Each rule constitutes an instantiation of classes whose values will be constrained with OCL specification. It is modeled as a single UML scenario (Object diagram [28]) with objects and links among them, as follows:

(i) Rules ***r1, r2, r3, r4*** and ***r5*** are modeled with instances of *Temporal Constraint*, whose objects depend on *Activity owner*:

"**r1**: Preference analysis lasts one or two days" and "**r2**: Contract signing is carried out in one hour maximum" are flexible duration constraints.
"**r3**: At most, three Contract amendments in a month are allowed" is a cardinality constraint.
"**r4**: Event celebration will be celebrated on April 20th between 10:00am and 20:00pm" is a fixed duration rule.
"**r5**: Contract cancellation is not allowed in December" is an example of absence conditional rule.

(ii) Rules ***r6, r7, r8, r9, r10*** and ***r11*** have instances of *Temporal Dependency*, which is an association between two

Table 7
BPMN scenarios for example rules.

#	Rule	#	Rule
	r1: *Preference analysis* lasts one or two days 		**r2:** *Contract signing* is carried out in one hour maximum. 
	r3: *At most, three Contract amendments in a month are allowed.* 		***r4:*** *Event celebration will be celebrated on April 20th between 10:00am and 20:00pm.* 
	r5: *Contract cancellation* is not allowed in December. 		**r6:** *Book Catering* and *Book place* begin simultaneously. 
			**r7**: *Recruitment* and *Book catering* finish simultaneously.
			**r8:** *Resource procurement begins within thirty days from the Event celebration.* **r9:** *Resource procurement* should end within a week before than *Event celebration.*
	r10: If *Book catering* starts, then *Contract amendment* is not allowed. 		**r11:** If *Recruitment* is started, then *Contract cancellation* is only possible 20 days before. 

objects: *Activity successor* and *Activity predecessor*. Now, each rule represents a precedence relation with time semantics between both activities, as follows:

"**r6**: Book catering and Book place begin simultaneously" and "**r8**: Resource procurement begins within thirty days from the Event celebration" are Start To Start dependency rules.
"**r7**: Recruitment and Book catering finish simultaneously" and "**r9**: Resource procurement should end within a week before Event celebration" are Finish To Finish dependency rules.
"r10: If Book catering starts, then Contract amendment is not allowed" and "**r11**: If Recruitment starts, then Contract cancellation is only possible 20 days before" are Absence Unconditional and Conditional dependency rules.

6.4. Comparative analysis with other approaches

In this section we compare our proposal with other approaches that specify time dimension over processes (see Table 8). Firstly, we differentiate two groups of works: I) References *(i-m),* which use the BPMN language [25,26,39] where our approach *(l)* is included, and II) Works *(a-h),* which use other languages. The work *(i)* by Watahiki et al. [35] belongs to both groups.

(I) The first group is related to distinct languages or techniques, so that: *(a)* Time Petri Nets, with works by Huai et al. [19] and Makni et al. [23,24]; *(b, i)* Timed Automata, used by Cheikhrouhou et al. [6,7] and Watahiki et al. [35]; *(c)* Timed Workflow Nets, used by Du et al. [10]; *(d)* Web Service Timed State Transition Systems, with the work by Kazhamiakin et al. [21]; *(e)* XTUS-Automata, with Kallel et al. [20] reference; *(f)* Communicating Sequential Processes (CSP), used by Wong and Gibbons [37]; *(g)* Timed Activity Graphs, with works by Eder and Tahamtan [12,13]; and finally *(h)* the work by Lanz et al. [61], which proposes a set of time-patterns for process-aware information systems (PAIS) [33,41].
(II) The second group of works takes BPMN as a basis for time specification, so: *(j)* Flores and Sepúlveda [15] work with BPMN artifacts patterns; *(k)* Gagné and Trudel, in Time-BPMN [18], propose new decorators; and finally *(l),* Cheikhrouhou et al. [8] also use decorators for time and resource dimension of processes. For these works, the column "Control-flow overload" shows that the proposal is based on "BPMN artifact blocks" to solve the specification of each rule (this approach overloads the process control flow), and the column "BPMN 2.0 standard" shows whether the solution respects the standard √, or it proposes new extensions out of its scope ⊘.

The column *"Rules Taxonomy Support"* shows the coverage level of our rule taxonomy (Fig. 3) that may be: ● fully supported or ◉ partially supported.

Thus, our solution, that is BPMN-based, proposes a declarative rules-model (OCL formulas for each time rule) that overlay the BPMN Meta-model, so that we do not overload the main flow and we respect the standard BPMN 2.0. We can take advantage of decorator-based proposals, since attending to the semantics we have, it is easy to customize process modeling tools with stereotypes, new shapes or icons to provide processes with a time dimension graphical appearance. This meta-model based solution helps us in other of our goals: "The extraction of time dimension from legacy databases" as we will show with the case study in the next section. We have chosen this approach as it is easy to extend

Table 8
Comparative analysis with time specification approaches.

Research work	Rules Taxonomy Support	Language								BPMN				
		Time Petri Nets	Timed Automata	Time Workflow Nets	Web Service Timed State Transition Systems	XTUS-Automata	Communicating Sequential Processes (CSP)	Timed Activity Graphs	Time-patterns for PAIS	Patterns	Decorators	Meta-model extensions	Control-flow overload	PMN 2.0 standard
a) Huai et al. [19] and Makni et al. [23-24]	◉	✓												
b) Cheikhrouhou et al. [6-7]	◉		✓											
c) Du et al. [10]	◉			✓										
d) Kazhamiakin et al. [21]	◉				✓									
e) Kallel et al. [20] with XTUS-Automata	◉					✓								
f) Wong & Gibbons [37]	◉						✓							
g) Eder & Tahamtan [12-13]	◉							✓						
h) Lanz et al. [61]	●								✓					
i) Watahiki et al. [35]	◉		✓								✓		⊘	✓
j) Flores & Sepúlveda [15]	◉									✓			✓	✓
k) Time-BPMN: Gagné & Trudel [18]	◉										✓		⊘	⊘
l) Cheikhrouhou et al. [8]	●										✓		⊘	⊘
***m)* Our approach**	●											✓	⊘	✓

● Fully supported, ◉ Partially supported, ✓ It checks the feature, ⊘ It does not check the feature

it with resource and case dimensions, for example, by using with works by Awad et al. [3] and Stroppi et al. [30], which also use process meta-models.

7. A case study: AQUA-WS project

The case study illustrates a MDE-based approach [46,47] for reverse engineering [38,51] of a legacy database, where MS Project Server [54] is the source system and BPMN is the target system.

AQUA-WS [9] is a three-year project carried out in EMASESA (a local company in Seville responsible for water treatment and distribution to citizens) that consists in modernizing the software architecture, moving it from client-server to web-design. The system has several subsystems which are composed of applications. The Navigational Development Technique (NDT), created by Escalona and Aragón [14], was the reference methodology selected for the applications development lifecycle, and MS Project was intensively used for planning and controlling all kinds of activities.

Process mining [33,34] aims to discover, monitor and improve real processes by extracting knowledge from event logs readily available in today's PAIS [33,41]. However there are a lot of legacy systems that do not belong to PAIS category, although they hide some knowledge related to organization processes. Van der Aalst [43] gives a conceptualization for all changes in the database as events, proposing class, object and event models with the aim of generating event logs from legacy databases. The author concludes that there are no tools yet which develop this proposal using commercial DBMS.

MS Project Server [54] is a legacy system without event log, which is an artifact usually included in PAIS. Software organizations frequently use this system for planning and control projects. A project gathers knowledge about temporal dimension of business processes. Our goal is to take out a process from a project plan, involving activities, time rules and time dependencies. If the source is a well-defined plan then the target process should be a good approximation to the process in the organization. We will use our metamodel (Fig. 6) for the MDE-based reverse engineering [38,51] approach proposed in [2].

7.1. MS project server task model

MS Project Server is a collaborative environment for planning and control projects, based on MS SharePoint [57] and MS SQL∗Server [55]. Each installation has four databases instances: Drafts, Published, Archive and Reporting. The Published instance contains all the base tables regarding the definition of task models. Fig. 10 shows a relational diagram with the main tables and foreign keys.

7.2. NDT ASI phase

All AQUA-WS applications fit the NDT phases. We select a group of tasks concerning subsystem "Equipment and facilities" that are grouped as activity "64: (ASI) NDT Analysis phase- Equipment and Facilities" (Fig. 11), that is defined as a subproject in the global project.

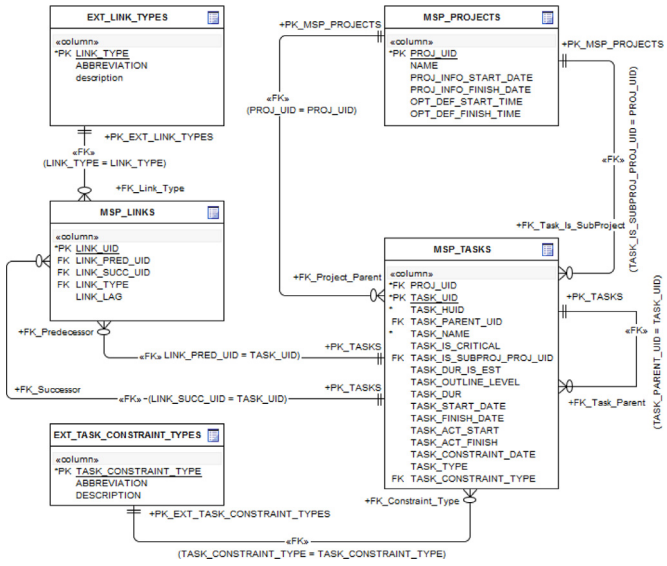
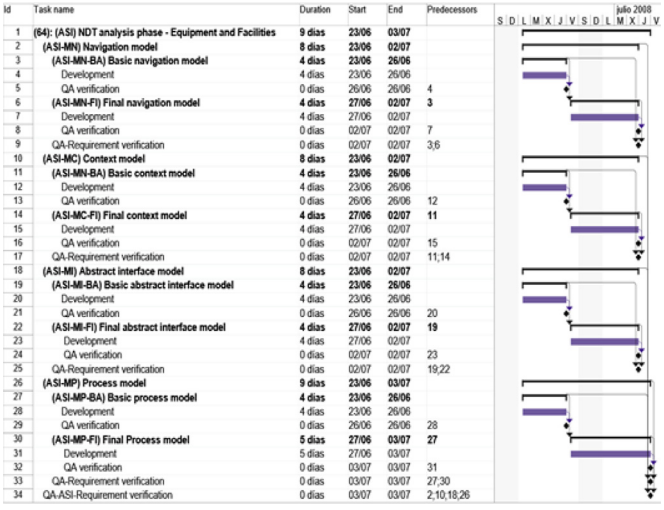


Fig. 10. MS project server task model (published instance).



Id	Task name	Duration	Start	End	Predecessors
1	(64): (ASI) NDT analysis phase - Equipment and Facilities	9 días	23/06	03/07	
2	(ASI-MN) Navigation model	8 días	23/06	02/07	
3	(ASI-MN-BA) Basic navigation model	4 días	23/06	26/06	
4	Development	4 días	23/06	26/06	
5	QA verification	0 días	26/06	26/06	4
6	(ASI-MN-FI) Final navigation model	4 días	27/06	02/07	3
7	Development	4 días	27/06	02/07	
8	QA verification	0 días	02/07	02/07	7
9	QA-Requirement verification	0 días	02/07	02/07	3;6
10	(ASI-MC) Context model	8 días	23/06	02/07	
11	(ASI-MN-BA) Basic context model	4 días	23/06	26/06	
12	Development	4 días	23/06	26/06	
13	QA verification	0 días	26/06	26/06	12
14	(ASI-MC-FI) Final context model	4 días	27/06	02/07	11
15	Development	4 días	27/06	02/07	
16	QA verification	0 días	02/07	02/07	15
17	QA-Requirement verification	0 días	02/07	02/07	11;14
18	(ASI-MI) Abstract interface model	8 días	23/06	02/07	
19	(ASI-MI-BA) Basic abstract interface model	4 días	23/06	26/06	
20	Development	4 días	23/06	26/06	
21	QA verification	0 días	26/06	26/06	20
22	(ASI-MI-FI) Final abstract interface model	4 días	27/06	02/07	19
23	Development	4 días	27/06	02/07	
24	QA verification	0 días	02/07	02/07	23
25	QA-Requirement verification	0 días	02/07	02/07	19;22
26	(ASI-MP) Process model	9 días	23/06	03/07	
27	(ASI-MP-BA) Basic process model	4 días	23/06	26/06	
28	Development	4 días	23/06	26/06	
29	QA verification	0 días	26/06	26/06	28
30	(ASI-MP-FI) Final Process model	5 días	27/06	03/07	27
31	Development	5 días	27/06	03/07	
32	QA verification	0 días	03/07	03/07	31
33	QA-Requirement verification	0 días	03/07	03/07	27;30
34	QA-ASI-Requirement verification	0 días	03/07	03/07	2;10;18;26

Fig. 11. Instance of NDT analysis phase (ASI).

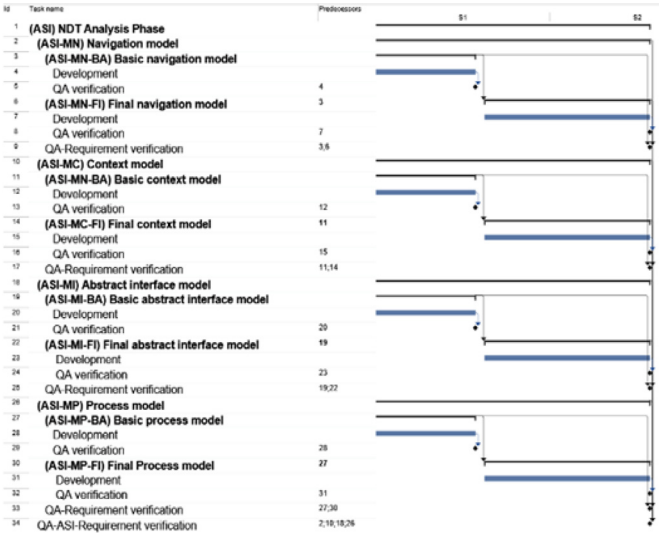
The project manager provides a subproject pattern for ASI phase (Fig. 12) that is parameterized and replicated for each ASI instance in the project, such as previous activity with id=64 (Fig. 11). Both plans (Figs. 11 and 12) are defined in the legacy database as instances of task model of Fig. 10.

Fig. 13 depicts a process model for NDT requirement and analysis phases that are defined by a business expert. The model uses artifacts as transactions, ad-hoc processes, error events and recursive logic to express iterations among activities. We try to extract an approximation to processes through source MS Project plans. The next section describes a heuristic approach to carry out this idea.

7.3. Heuristics to extract BPMN processes from project plans

In this case the domain expert is the same as the expert in IT, who is the project manager, but we want to offer simple models that are consistent with the descriptive level of BPMN [69], minimizing the number of symbols used. Thus, the generated processes will be better understood by the average expert in other domains.

We intend to realize correspondences between a MS Project plan and a BPMN business process, so that: i) a project as a process; ii) activities: single tasks or subprocesses; and iii) gateways to express the control flow according to time constraints and time dependencies.



Id	Task name	Predecessors
1	(ASI) NDT Analysis Phase	
2	(ASI-MN) Navigation model	
3	(ASI-MN-BA) Basic navigation model	
4	Development	
5	QA verification	4
6	(ASI-MN-FI) Final navigation model	3
7	Development	
8	QA verification	7
9	QA-Requirement verification	3;6
10	(ASI-MC) Context model	
11	(ASI-MN-BA) Basic context model	
12	Development	
13	QA verification	12
14	(ASI-MC-FI) Final context model	11
15	Development	
16	QA verification	15
17	QA-Requirement verification	11;14
18	(ASI-MI) Abstract interface model	
19	(ASI-MI-BA) Basic abstract interface model	
20	Development	
21	QA verification	20
22	(ASI-MI-FI) Final abstract interface model	19
23	Development	
24	QA verification	23
25	QA-Requirement verification	19;22
26	(ASI-MP) Process model	
27	(ASI-MP-BA) Basic process model	
28	Development	
29	QA verification	28
30	(ASI-MP-FI) Final Process model	27
31	Development	
32	QA verification	31
33	QA-Requirement verification	27;30
34	QA-ASI-Requirement verification	2;10;18;26

Fig. 12. NDT analysis phase (ASI) subproject pattern.

The heuristics is a MDE-based approach [2], depicted in Fig. 14. Hence, we need:

(i) **Source system**. It is the legacy database that stores instances (Figs. 11 and 12) of the task model (Fig. 10). MS Project Server runs on SQL∗Server relational DBMS [55]. We need PSM [48,49] metamodels for SQL∗Server that are based on: GASTM [50] from OMG Information Management Metamodel (IMM) initiative [52] and SASTM [50] from Arevalo et al. [56].

(ii) **Target system**. It is BPMN in this case, where its metamodel has time dimension extensions (Fig. 6), but it could be other standard commonly used in the software field as SPEM 2.0 or ISO/IEC 24744:2007, as well as NDTQ-framework [62], which is a framework that implements the NDT methodology [14].

(iii) **Heuristics.** Since our most relevant decision is to select a MDE-based approach to extract processes from legacy databases, we will use our Meta-model (Section 6) that extends the BPMN Meta-model with time-dimension. The heuristics is a M2M transformation that means an algorithm that maps artifacts from instances of legacy database to a BPMN model, which shows control and temporal dimension. We have chosen some criteria to transform artifacts among the two levels of abstraction (*PSM: the legacy database* and *CIM: the extended BPMN System with new time rules*):

(a) **Existing rule types within the source system**. Database is the more stable artifact of a legacy system that stores states related to traces of processes involving time dimension. We could consider these traces as *"Hidden knowledge of processes"* [43] within each legacy database (Fig. 14). Focusing on the time rules that are supported in a selected legacy system, we must extract specific views of the database model in correlation with our taxonomy of rules (Fig. 3). In our case study the source system is MS Project and we have taken out the task model for the Published instance of MS SQL∗Server (Fig. 10). This task model allows us to point to all time rules stored in every project without analyzing the hardcoded rules within the MS Project System.

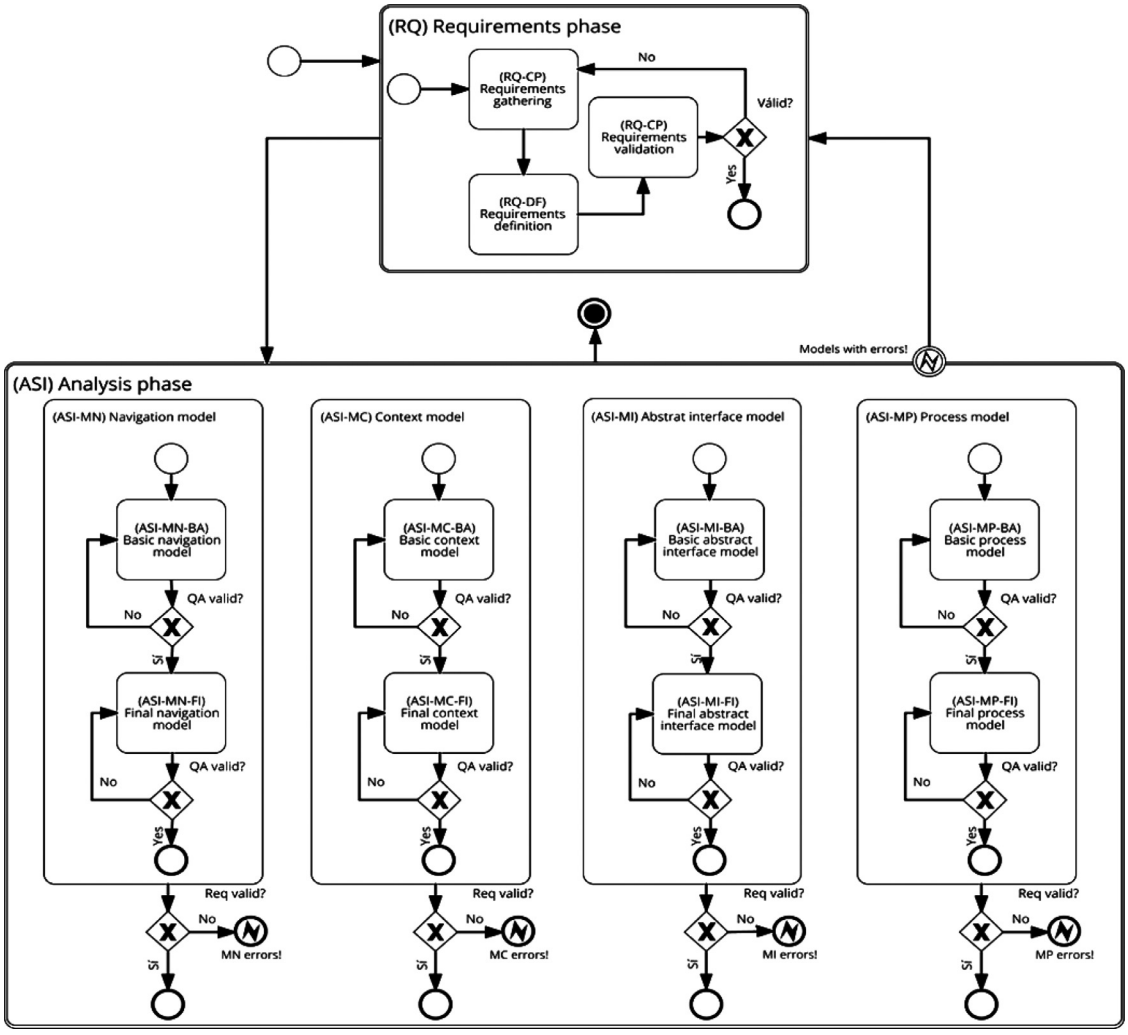


Fig. 13. NDT requirement and analysis process.

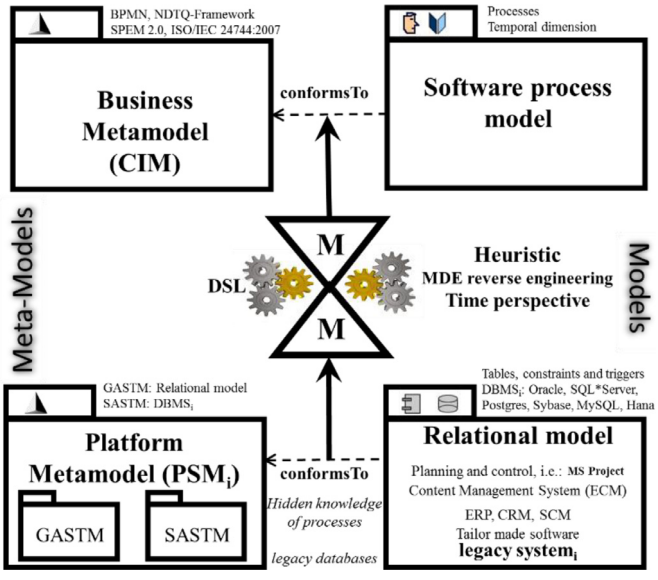


Fig. 14. MDE-based reverse engineering of legacy databases.

(b) **Temporal dimension formalization at expert abstraction level.** Our extended BPMN Meta-model (Section 6) includes the same taxonomy of rules (Fig. 3) that covers the task model of Fig. 10 as well as comprises more kinds of time rules that MS Project allows. Our choice is the formalization of time dimension with the OCL specification language. We offer a body of rules that stands over the BPMN Meta-model. This rule model avoids that experts have to reformulate time rules within each business model. They only need to specify the kind of rule and the attributes associated with them. In our case study, these attributes are automatically extracted from the source: MS Project legacy database.

(c) **Mapping criteria.** Once we have fixed the source and the target system, we may specify mapping algorithms to translate artifacts from legacy databases into BPMN models, where each one is based on its corresponding Meta-model. In this sense, our main mapping choices are:

- *Business Process*. A *Project* in the source system is mapped as a *BPMN Process*. All subsequent artifacts are subordinated to this project.

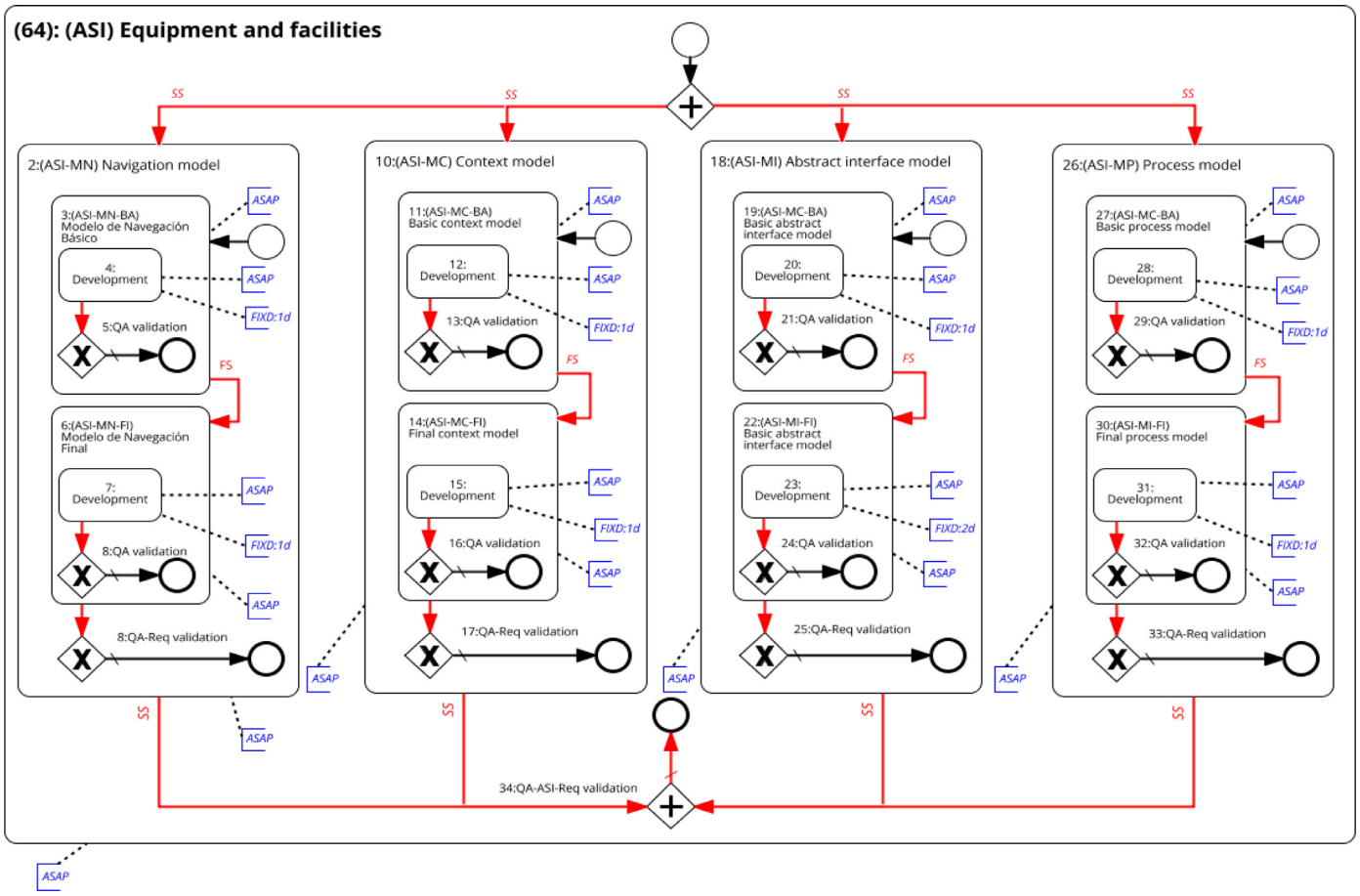


Fig. 15. NDT analysis process extracted from MS project server legacy database. For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.

- *Activities. Tasks* are the division unit of *Projects* that are mapped as *BPMN activities.* A *Project* may include *Task Groups (or Task Hierarchy);* in this case we also map them as *AdHoc BPMN Subprocesses* because they run concurrently. Besides, a *MS Project Task* may be a *complex one*, then it may be externally planned and controlled as a *Subproject*. We map the *Subproject* as a *BPMN Subprocess,* which is also a *Project* and it may have its own structure.
- *Time Constraint and Time Dependencies. MS Project* allows fixed and flexible duration and constraining *start* and *end events* with a due-date to specify *fixed events (MSON, MFON)* and *flexible events (NET, NLT)*. Allen's Algebra constraints are also allowed (*ASAP, ALAP* and *SS, SF, FS and FF*).

We translate these constraints into *BPMN models.* We may remember that this model includes *OCL formulas for each rule* that are all applied without manual management of business experts.

Although BPMN is a language that allows experts to model business from scratch, our approach automatically extracts BPMN proposed-models with a minimum set of processes artifacts (basic or declarative BPMN compliance [25,26,39]) and time dimension semantics.

7.4. Results

The execution of the heuristics (Fig. 14) with the instance of the project plan (Fig. 11) generates the result that is shown in Fig. 15 as an approximation to the business process that describes "64: (ASI) NDT Analysis phase - Equipment and Facilities". This BPMN diagram is an approximation to the real process of NDT ASI phase applied to this subsystem (Equipment and facilities). We may compare the process obtained (Fig. 15) with the process model depicted by the NDT expert (Fig. 13):

(i) Firstly, concerning activities that are detected, the running example give: a) a process for the plan with four subprocesses, each one for each embedded model: 2:(ASI-MN) Navigation model, 10: (ASI-MC) Context model, 18:(ASI-MI) Abstract interface model and 26:(ASI-MP) Process Model. In the same way, they are detected Basic (activities 3, 11, 19 and 27) and Final (activities 6, 14, 22 and 30) versions of models that are subordinated in a correct way are identified. Besides, these ones are composed of a single task: Development (activities 4, 7, 12, 15, 20, 23, 28 and 31) and gateways that depict QA validation, QA-Req validation in all cases, and finally the validation of the ASI phase regarding the requirements phase (34:QA-ASI-Req validation).

We conclude that the main structures are drawn with a good level of alignment.

(ii) Secondly, regarding the recursive logic that involves loops for reviewing models, a tool such as MS Project only allows to set precedence relationships, but not a more powerful logic to derive this recursiveness.

(iii) The extraction of *time-perspective* gives new BPMN artifacts (blue and red colors in Fig. 15) to show *control-perspective* in a standard way, also adding a set of overlaid constraints that conforms to the extended BPMN metamodel, thus: a) first level models run in parallel (SS time dependencies), and

final submodels always run sequentially after Basic models (FS dependencies) and b) time constraints are represented with blue annotations (durations as FIXD and FLEXD, and ASAP flexible starts and ends).
Again, the derived control flow is in good accordance with the real model.

(iv) Finally, we notice that the real model uses error events and transactions. In this case, it is not possible to derive this type of knowledge from a project plan.

The running example is a case or instance of a process, derived from a project plan (level M_0 [48,49]), if we use an instance as the pattern for NDT ASI (Fig. 12), then we will reach a process model (level M_1 [48,49]) for this pattern. The main aspect is to classify activities (at every level: a single task, a group of tasks and subprojects) in activity classes.

Thus, it is worth pointing out that our approach generates processes that are close to real models, and are a good start point for the business expert. Our approach is also useful for generating an event log as a set of processes traces, where each trace is an instance of a software process composed of activities, where each one is classified according to NDT methodology (although it could be any other). XES [44] is a standard format for event logs that can be used with automated tools for process mining [33,34], such as ProM [44,45]. In light of this, we would be able to create process models in a different way.

8. Discussion

We have gathered the semantics to depict time rules with a MDE-based approach [46,47], thus we propose a solution with BPMN extensions. It is considered a very technical specification, nearer to IT experts than users and business experts who usually check results graphically [58] through their favorite modeling tool. However, we propose to limit the number of symbols to facilitate the comprehensibility [67] by the average domain experts in any industry.

Other approaches, based on standard BPMN constructs like [15] or alternatives, as Tables 2 and 3 shows, could generate models overloaded with a lot of artifacts. In addition, proposals focused on BPMN decorators like [8,18] may produce more clear and comprehensible business models, although we cannot use them in modeling tools until BPMN standard level gathers this semantics for time. Our proposal includes the metamodel (Section 6), OCL [27] constraints for time rules, the instantiated model for each case and all the information that is needed to draw the aspect we have managed. Normally, modeling tools allow customizing graphical interfaces with images, icons, text and associations or by means of stereotyping classes and linking artifacts. If we support our metamodel with the repository of a modeling tool, we can use its capacities to display it with graphical notations of authors such as [8,18] (see Table 4), as we share the same time rules. To summarize, we propose a solution that strictly fits BPMN 2.0 standard, specifying the temporal dimension declaratively with OCL formulas, for avoiding the overhead of models with more imperatives artifacts. Then we can get the benefits of the solutions as [8,15,18] and avoid their weaknesses.

Besides, this proposal may be the key to automatically derive executable processes, either in standard languages like *Web Services Business Process Execution Language (WS-BPEL)* or proprietary code to interact with the application level.

9. Conclusions and future work

BPM is a strategic advantage for organizations to support their operations. Besides, BPM also enables them to model different aspects and perspectives: *control-flow, organizational, case* and *time*. We firstly focus on time perspective because time dimension is present in all kind of business processes.

BPMN is a recent ISO standard, but is cataloged as a de-facto standard among other languages. It is also the preferred technique for experts to model business processes, since it is mainly an imperative language with some weaknesses for time rules modeling. We use the basic or declarative BPMN compliance level, consisting of a minimum set of symbols to facilitate the comprehensibility of the models. We work with time rule taxonomy and BPMN 2.0 metamodel extensions to write formalization of each rule with OCL constraints. For this reason, we use a declarative approach where new time specifications may be overlaid over the main *control flow* of a BPMN process that is modeled with imperative style. This allows us to avoid models overloaded with a lot of artifacts that generate non-understandable views on the side of users and experts.

Our solution is a well-known model-based approach by IT experts. It not only guarantees that executable business processes specification be easily generated with these enhanced time rules, but also the fact that business experts aim to draw graphic models that depict these new time extensions. Sometimes, modeling tools graphical interface could be customized. Besides, all extensions are based on the current level of BPMN 2.0, therefore we do not need to wait until the standard supports these new rules.

Our proposal allows extensions such as a) more kinds of time rules, for example, in collaborative processes and b) integration of *time* with other BPM perspectives, like *organizational, data and case*.

We can use this proposal with legacy databases which hide the time-dimension (see our previous work in [2,56] for further details), because general approaches for the modernization of legacy systems are based on OMG ADM [38]. These approaches deal with mining application code and database schemas, but the models obtained lack of knowledge since it is too difficult to derive complex business rules from source systems that are coded at low level. We can enrich target models whenever we strengthen these general approaches with semantic perspectives that are always scattered into legacy systems. Therefore, we can work with different systems including *time-perspective*, for example: other planning and control project tools, such as Redmine; Content Management Systems (CMS), such as Alfresco and MS SharePoint; Enterprise Resource Planning (ERP), such as SAP, Microsoft Dynamics, Oracle Business Solutions, and also, tailor-made software. We will find time rules to incorporate them into BPMN models that expert can understand and manage. In addition, it is possible to derive more rules from these systems, by means of research works such as [3,30] based on the BPMN metamodel with *organizational perspective*, and [3] that also works with case perspective. After that, it is possible to utilize other BPMN perspectives to extend these approaches for reverse engineering.

We could also use our approach for generating event logs for no process-aware legacy systems, as they hide states that are the result of process events executed in an organization. So, we can build log traces, for example, in a XES format [44] that may be used for process mining approaches [33,34].

We have collaborated in many industrial projects like AQUA-WS [9], such as CALIPSOneo [59,60] with Airbus[1], introducing methodological contributions based on the Navigational Development Technique (NDT), proposed by Escalona and Aragón [14]. We have also gathered data, code and database schemas to compare and verify our approaches that point to reverse engineering business process within these projects.

[1] http://www.airbus.com/

Acknowledgments

This research has been supported by the MeGUS project (TIN2013-46928-C3-3-R) and by the SoftPLM Network (TIN2015-71938-REDT) of the Spanish Ministry of Economy and Competitiveness.

References

[1] J.F. Allen, Maintaining knowledge about temporal intervals, Commun. ACM 26 (1983) 832–843.
[2] C. Arevalo, I. Ramos, M.J. Escalona, Discovering Business Models for Software Process Management: An Approach for Integrating Time and Resource Perspectives from Legacy Information Systems, ICEIS, 2015.
[3] A. Awad, A. Grosskopf, A. Meyer, M. Weske, Enabling Resource Assignment Constraints in BPMN, Hasso Plattner Institute, Potsdam, 2009.
[4] F. Bonnet, G. Decker, L. Dugan, M Kurz, Z. Misiak, S. Ringuette, Making BPMN a true lingua franca, BPM Trends, 2014.
[5] S. Cheikhrouhou, S. Kallel, N. Guermouche, M. Jmaiel, A survey on time-aware business process modeling, in: ICEIS, 2013, pp. 236–242.
[6] S. Cheikhrouhou, S. Kallel, N. Guermouche, M. Jmaiel, Enhancing formal specification and verification of temporal constraints in business processes, in: IEEE SCC, 2014, pp. 701–708.
[7] S. Cheikhrouhou, S. Kallel, N. Guermouche, M. Jmaiel, Toward a verification of time-centric business process models, in: WETICE, 2014, pp. 326–331.
[8] S. Cheikhrouhou, S. Kallel, N. Guermouche, M. Jmaiel, Toward a time-centric modeling of business processes in BPMN 2.0, in: iiWAS, 154, 2013, p. 2013.
[9] C.R. Cutilla, J.A. García-García, M. Alba, M.J. Escalona, J. Ponce, L. Rodríguez, Aplicación del paradigma MDE para la generación de pruebas funcionales; Exp. AQUA-WS, 6ª Conf. Iberic. S.T.I. ISBN: 978-989-96247-4-0, 2011.
[10] Y. Du, P. Xiong, Y. Fan, X. Li, Dynamic checking and solution to temporal violations in concurrent workflow processes, Syst. Man Cybernetics, Part A 41 (6) (2011) 1166–1181.
[11] M. Dumas, M. La Rosa, J. Mendling, H.A. Reijers, Fundamentals of Business Process Management, Springer, Heidelberg, 2013.
[12] J. Eder, A. Tahamtan, Temporal conformance of federated choreographies, in: Database and Expert Systems Applications, Springer, Berlin Heidelberg, 2008, pp. 668–675.
[13] J. Eder, A. Tahamtan, in: Temporal Consistency of View Based Interorganizational Workflows, Springer, Berlin Heidelberg, 2008, pp. 96–107.
[14] M.J. Escalona, G. Aragón, NDT. A model-driven approach for web requirements, IEEE TSE 34 (3) (2008) 377–390.
[15] C. Flores, M. Sepúlveda, Temporal specification of business processes through project planning tools, in: BPMW, Springer, Berlin Heidelberg, 2011, pp. 85–96.
[16] D. Gagné, A. Trudel, The temporal perspective: expressing temporal constraints and dependencies in process models, Process Models in BPM and Workflow Handbook, 2008.
[17] D. Gagné, A. Trudel, A formal temporal Semantics for microsoft project based on Allen's interval algebra, in: BPSC, 2009, pp. 32–45.
[18] D. Gagné, A. Trudel, Time-bpmn, in: Commerce and Enterprise Computing. CEC'09. IEEE Conference on, IEEE, 2009, pp. 361–367.
[19] W. Huai, X. Liu, H. Sun, Towards trustworthy composite service through business process model verification, in: 7th International Conference on UIC/ATC, 2010, pp. 422–427.
[20] S. Kallel, A. Charfi, T. Dinkelaker, M. Mezini, M. Jmaiel, Specifying and monitoring temporal properties in web services compositions, in: Web Services, ECOWS'09. Seventh IEEE European Conference on, 2009, pp. 148–157.
[21] R. Kazhamiakin, P. Pandya, M. Pistore, Representation, verification, and computation of timed properties in web, in: Web Services. ICWS'06. International Conference on, IEEE, 2006, pp. 497–504.
[22] J. Kelley, M. Walker, Critical-path planning and scheduling, in: Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference, ACM, 1959, pp. 160–173.
[23] Makni, M., Tata, S., Yeddes, M.M., Hadj-Alouane, N.B. Satisfaction and coherence of deadline constraints in inter-organizational workflows. In Proceedings of the CoopIS, On the Move to Meaningful Internet Systems: OTM, 2010, volume 6426 of LNCS, pp. 523–539. Springer.
[24] Makni, M., Hadj-Alouane, N.B., Tata, S., Yeddes, M.M. Negotiating deadline constraints in interorganizational logistic systems: a healthcare case study. In Proceedings of the International Workshops and Education Track on BPM, 2011, volume 100 of LNBIP, pp. 108–118. Springer.
[25] OMG. *Business Process Model and Notation (BPMN) v2.0*, 2011.
[26] OMG. *Business Process Model and Notation (BPMN)* | *ISO/IEC 19510:2013*, 2013.
[27] OMG. *Object Constraint Language (OCL) v2.4*, 2014.
[28] OMG. *Unified Modeling Language (UML) v2.0*, 2011.
[29] H.A. Reijers, T. Slaats, C. Stahl, in: Declarative Modeling–An Academic Dream or the Future for BPM, Springer, Berlin, Heidelberg, 2013, pp. 307–322.
[30] L.J.R. Stroppi, O. Chiotti, P.D. Villarreal, Extended resource perspective support for BPMN and BPEL, in: CIbSE, 2012, pp. 56–69.
[31] O. Svatos, Modeling suspension and continuation of a process, J. Syst. Integration 3 (2) (2012) 62–73 ISSN:1804-2724.
[32] W. Van der Aalst, Business process management: a personal view, BPM J. 10 (2) (2004) 5.
[33] W. Van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer, Heidelberg, 2011.
[34] W Van Der Aalst, et al., Process mining manifesto, in: BPM workshops, Berlin Heidelberg, Springer, 2012, pp. 169–194.
[35] K. Watahiki, F. Ishikawa, K. Hiraishi, Formal verification of business processes with temporal and resource constraints, in: Systems, Man, and Cybernetics (SMC), IEEE International Conference, 2011, pp. 1173–1180.
[36] M. Weske, Business Process Management. Concepts, Languages, Architectures, 2nd Edition, Springer, 2012.
[37] P. Wong, J. Gibbons, A relative timed semantics for BPMN, Electron. Notes Theor. Comput. Sci. 229 (2) (2009) 59–75.
[38] W.M. Ulrich, P. Newcomb, Information Systems Transformation: Architecture–Driven Modernization Case Studies, Morgan Kaufmann, 2010.
[39] M. von Rosing, H. von Scheel, A. Scheer, The Complete Business Process Handbook, Body of Knowledge from Process Modeling to BPM, Volume I, 1s t Edition, Morgan Kaufmann, 2014 http://store.elsevier.com/product.jsp?isbn=9780127999593.
[40] J. Mendling, H.A. Reijers, W.M.P. van der Aalst, Seven process modeling guidelines (7PMG), Inform. Softw. Technol. 52 (2) (2010) ISSN 0950-5849, http://dx.doi.org/10.1016/j.infsof.2009.08.004127-136.
[41] C. Ayora, V. Torres, B. Weber, M. Reichert, V. Pelechano, VIVACE: a framework for the systematic evaluation of variability support in process-aware information systems, Inform. Softw. Technol. 57 (2015) 248–276 ISSN 0950-5849, http://dx.doi.org/10.1016/j.infsof.2014.05.009.
[42] I. Moreno-Montes de Oca, M. Snoeck, H.A. Reijers, A. Rodríguez-Morffi, A systematic literature review of studies on business process modeling quality, Inform. Softw. Technol. 58 (2015) 187–205 ISSN 0950-5849, http://dx.doi.org/10.1016/j.infsof.2014.07.011.
[43] W.M.P. Van der Aalst, in: Extracting Event Data From Databases to Unleash Process Mining, Springer International Publishing, 2015, pp. 105–128.
[44] H.M.W. Verbeek, J.C. Buijs, B.F. Van Dongen, W.M. van der Aalst, Xes, xesame, and prom 6, in: Information Systems Evolution, Springer, Berlin, Heidelberg, 2010, pp. 60–75.
[45] A. Kalenkova, M. De Leoni, W.M.P. van der Aalst, Discovering, analyzing and enhancing BPMN models using ProM★, in: BPM 12th International Conference, 2014, pp. 7–11.
[46] S. Kent, Model driven engineering, in: Integrated Formal Methods, Springer, Berlin, Heidelberg, 2002, pp. 286–298.
[47] D.C. Schmidt, Model-driven engineering, Computer 39 (2) (2006) 25–31.
[48] OMG. Modern Driven Architecture (MDA), 2011.
[49] A.G. Kleppe, J.B. Warmer, W. Bast, MDA Explained: The Model Driven Architecture: Practice and Promise, Addison-Wesley Professional, 2003.
[50] OMG, Syntax Tree Metamodel. v. 1.0, ASTM, 2011.
[51] E. Chikofsky, J.H. Cross, Reverse engineering and design recovery: a taxonomy, IEEE Softw 7 (1) (1990) 13–17.
[52] OMG. *Information management metamodel (IMM)*, 2006.
[53] J. Bisbal, D. Lawless, B. Wu, J. Grimson, V. Wade, R. Richardson, D.O. Sullivan, An overview of legacy information system migration, in: Software Engineering Conference APSEC'97 and ICSC'97. Proceedings, IEEE, 1997, pp. 529–530.
[54] T. Stover, Microsoft® office project 2007 inside out, Microsoft Press, 2007.
[55] R. Colledge, SQL*Server, Administration in action, Manning, 2008.
[56] C. Arevalo, M.T. Gómez-López, A.M. Reina Quintero, I. Ramos, An architecture to infer business rules from event condition action rules implemented in the persistence layer, in: R. Perez-Castillo, M. Piattini (Eds.), Uncovering Essential Software Artifacts through Business Process Archeology, Business Science Reference, Hershey, PA, 2013, pp. 201–221, doi:10.4018/978-1-4666-4667-4.ch008.
[57] T. Smith, S. Bates, SharePoint 2007 User's Guide: Learning Microsoft's Collaboration and Productivity Platform, Apress, 2007.
[58] R. Lu, S. Sadiq, A survey of comparative business process modeling approaches, in: Business Information Systems, Springer, Berlin Heidelberg, 2007, pp. 82–94.
[59] M.J. Escalona, J.A. García-García, F. Más, M. Oliva, C. Del Valle, Applying model-driven paradigm: CALIPSOneo experience, in: Proceedings of the Industrial CAiSE'13, vol. 1017, 2013, pp. 25–32.
[60] A. Salido, J.A. García-García, J. Gutiérrez, J. Ponce, Tests management in CALIPSOneo: a MDE solution, J. Softw. Eng. Appl. (2014).
[61] A. Lanz, B. Weber, M. Reichert, Time patterns for process-aware information systems, Requirements Eng. 19 (2) (2014) 113–141.
[62] J. Ponce, L. García-Borgoñon, J.A. García-García, M.J. Escalona, F.J. Domínguez–Mayo, M. Alba, G. Aragon, A model-driven approach for business process management, (CJICT) 1 (2) (2013) 32–52.
[63] R.G. Ross, BUSINESS RULE CONCEPTS: Getting to the Point of Knowledge, 4th Edition, Business Rule Solutions, LLC, 2013 ISBN: 0-941049-14-0.
[64] Baisley, D. OMG and business rules. Presentation available at http://www.omg.org/docs/omg/05-04-09.pdf, 2005.
[65] S. Jablonski, C. Bussler, Work flow Management. Modeling Concepts, Architecture and Implementation, Int. Thomson Computer Press, London, 1996.
[66] G. Wagner, Rule modeling and markup, in: Proceedings of the First international conference on Reasoning Web 2005, Springer-Verlag, 2005, pp. 251–274.
[67] M. Zur Muehlen, J. Recker, How much language is enough? Theoretical and practical use of the business process modeling notation, in: Advanced Information Systems Engineering, Springer, Berlin, Heidelberg, 2008, pp. 465–479.
[68] M. Zur Muehlen, J. Recker, in: We Still Don't Know How Much BPMN Is Enough, But We Are Getting Closer, Springer, Berlin, Heidelberg, 2013, pp. 445–451.
[69] J. Recker, Opportunities and constraints: the current struggle with BPMN, Bus. Process Manage. J. 16 (1) (2010) 181–201.
[70] M. Chinosi, A. Trombetta, BPMN: an introduction to the standard, Comput. Standards Interfaces 34 (1) (2012) 124–134.