The Temporal Perspective: Expressing Temporal Constraints and Dependencies in Process Models

Denis Gagné, Trisotech, Canada

André Trudel, Acadia University, Canada

Abstract

We characterize the temporal perspective of workflow specifications through a series of 31 temporal constructs that may occur when defining a process model. This characterization is independent of any specific modeling formalism or approach. We precisely define each temporal construct and when possible, provide a formal temporal account of these constructs based on Allen's interval algebra. Our characterization not only provides a basis and objective means to evaluate the temporal expressiveness of various formalisms and tools, but can also open the way to the integration of formal validation tools, such as constraint satisfaction or theorem proving systems, to verify temporal satisfiability of the workflow specification. As an example of the application of the proposed framework, we present a brief discussion of the suitability of BPMN and Gantt Charts for business process modeling from the temporal perspective.

1 Introduction

Business processes require the coordinated execution of individual activities to achieve a common business goal. A process model (or workflow specification) formally describes the structure of the workflow and how these atomic activities are coordinated and enacted by software systems named Workflow Management Systems (WfMS) [1] or by software suites named Business Process Management Suites (BPMS). These generic software systems normally support both the definition and the enactment of process models.

Process modeling is the task of creating process specifications. Several formalisms (e.g. BPMN [2], BPEL [3], UML [4], YAWL [5]) have been suggested for modeling business processes from various perspectives and for various purposes. The notion of perspectives in process modeling was addressed by Jablonski and Bussler [6]. More recently, the Workflow Pattern Initiative [7] reframed the notion of process modeling perspectives via a series of workflow pattern frameworks.

The various pattern frameworks from the Workflow Pattern Initiative provide a collection of generic recurring process constructs. Each pattern framework captures a different perspective such as control [8, 23], data [9], resources [10], and exceptions [11]. One important perspective missing from this list is the temporal perspective.

Time is a critical dimension of process modeling as it is directly related to customer satisfaction and cost reduction. The speedy delivery of goods or services has a direct impact on customer satisfaction. Furthermore, time optimization is often a very effective cost reduction strategy for an organization.

The temporal perspective contributes to both the definition and the enactment of a workflow specification. When defining a workflow, the temporal perspective allows the modeler to explicitly specify temporal constraints and dependencies to ensure that all temporal requirements of the process are met. At enactment time, the temporal perspective of the workflow specification leads to the ability to precisely schedule a process and its resources.

We characterize the temporal perspective of process modeling by providing a series of generic recurring temporal constructs. This characterization is independent of any specific modeling formalism or approach. We precisely define each temporal construct and when possible provide a formal temporal account of these constructs based on Allen's interval algebra [12]. Allen's interval algebra is the most popular in Artificial Intelligence. Although the management of time in the context of business processes has been studied (e.g. [13, 14, 15, 16, 17, 18, 19, 20, 21, 22]), previous work does not address all the temporal constructs studied in this paper. Furthermore, although Allen's interval algebra has been applied to many application areas, it has not received much attention in the business process community. To the best of our knowledge, only Lu et. al. [22] use Allen's algebra as the basis for flexible business process execution via constraint satisfaction.

Our characterization of the temporal perspective for workflow specifications provides a basis and objective means to evaluate the temporal expressiveness of various formalisms and tools. Also it opens the way to the integration of formal validation tools, such as constraint satisfaction or theorem proving systems, to verify the temporal satisfiability of the workflow specification.

The remainder of this paper is organized as follows. After providing a brief overview of Allen's interval algebra, we present our characterisation of the temporal perspective via various temporal constructs. We first identify and define the temporal components that capture the elapsed time occurring during the enactment of a process or an activity and then introduce Temporal Dependencies and Constraints. As an example of the application of the proposed framework, we present a brief discussion of the suitability of BPMN and Gantt Charts (as portrayed by Microsoft Project) for business process modeling from the temporal perspective.

2 ALLEN'S INTERVAL ALGEBRA

The most popular temporal reasoning approach in Artificial Intelligence is due to Allen [12]. Allen's approach is based on intervals and the 13 possible binary relations between them. The first relation is "precedes" which is represented by the letter "p". Interval A precedes interval B if A ends before B starts, and is written as "A{p}B". If A precedes B, then it is also the case that B is preceded by A. This inverse relation for precedes, "preceded by", is represented by "pi". The precedes relation is drawn in the top left of figure 1. This precedes diagram represents both "A{p}B" and "B{pi}A".

The other relations are meets (m), overlaps (o), during (d), starts (s), finishes (f), and equals (e). As with precedes, each of these relations has an inverse which is represented by appending an "i" to the relation symbol: mi, oi, di, si, and fi. The inverse of equals is equals. We refer to the 13 relations by their basic labels and they are all shown in figure 1.

Allen's interval relations are mutually exhaustive. For example, given two intervals, exactly one of the 13 relations will hold between them. It is impossible to have none or, two or more relations true between two temporal intervals.

Often, there is uncertainty as to exactly which relation holds between two intervals. For example, supper may be before or after going to the movies. We write this as: (supper p movies) xor (supper pi movies). Since exactly one of "p" or "pi" will be true, we use an exclusive-or (i.e., "xor"). A shorthand notation for the above formula is: supper{p,pi}movies. In general, the relation between two intervals is allowed to be any subset of I = {p,pi,m,mi,o,oi,d,di,s,si,f,fi,e} including I itself.

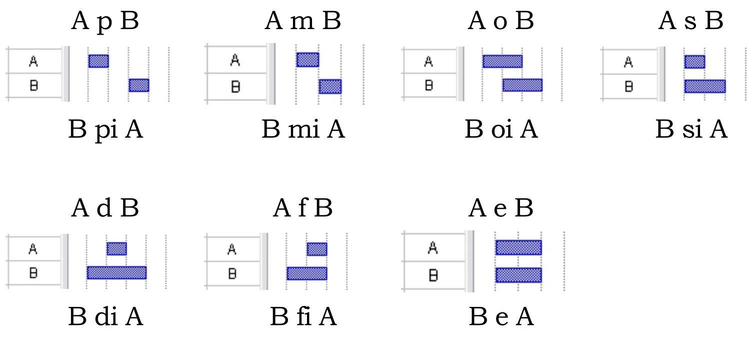


Fig. 1. Allen's 13 relations

We extend Allen's approach with points. We represent a temporal interval A by its endpoints written as (A-, A+). For example, the interval A = (1, 10) has a left endpoint A- = 1, and right endpoint A+ = 10. All intervals are convex (i.e., there are no gaps in the interval). We define the function *length(A)* to be the length of interval A.

The underlying temporal structure is assumed to be discrete, linear, and totally ordered. An example of such a structure is the integers.

3 THE TEMPORAL PERSPECTIVE

In this section, we introduce generic recurring temporal constructs from the temporal perspective and precisely define each temporal construct. When possible, we capture their temporal semantics with one or more logical formulas. This serves as a template for representing the general constructs.

In our formulas, we use the convention that capital letters, except for I, are constants and represent the specific temporal intervals during which activities take place. For example, the constant A represents the unique temporal interval over which the activity labeled A takes place. If activity A happens before activity B we write A{p}B.

Recall that I is not an activity, and represents the set of 13 possible interval relationships. For example, if a construct does not place any restrictions on activities C and D, we can write CID.

Variables range over temporal intervals and are specified with Greek letters (e.g., α, β).

We consider processes and subprocesses to be compound activities (i.e., they can be broken down into lower levels of detail through a set of sub activities [2]). Assume a process or subprocess A contains activities A_1, A_2, …, A_n. Each activity A_i occurs during A:

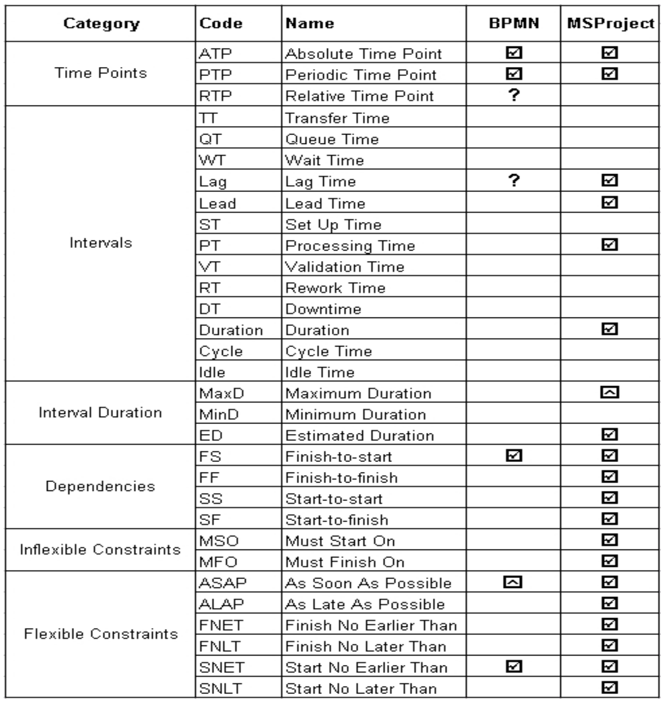
$$A^- \le Ai^- \le Ai^+ \le A^+ \qquad (1)$$

Non trivial processes may contain many activities which depend on one another. We will refer to the process or activity that depends on the other process or activity as the *Successor*, and the process or activity it depends on as the *Predecessor*.

During its enactment, a process or an activity will be instantiated and will take place over a certain period of time. We often refer to a process instance as a *Case* and to an activity instance as a *Task*. We represent the Case or Task instantiation, start and completion times with the constants *instantiation-time*, *start-time* and *finish-time* respectively. Note that these constants represent a time point and not an interval.

We now introduce a series of temporal components of interest from the temporal perspective. The 31 temporal constructs are shown in Table 1. The last two columns will be discussed at the end of the paper. In the following subsections, we precisely define each temporal construct and when possible, provide a logical specification using Allen's algebra.

Table 1. Expressing the temporal perspective in BPMN and MSProject. ☑ Directly expressible. ⧄ Indirectly expressible. ? Inconsistent documentation. A blank entry represents not expressible.



Category	Code	Name	BPMN	MSProject
Time Points	ATP	Absolute Time Point	☑	☑
	PTP	Periodic Time Point	☑	☑
	RTP	Relative Time Point	?	
Intervals	TT	Transfer Time		
	QT	Queue Time		
	WT	Wait Time		
	Lag	Lag Time	?	☑
	Lead	Lead Time		☑
	ST	Set Up Time		
	PT	Processing Time		☑
	VT	Validation Time		
	RT	Rework Time		
	DT	Downtime		
	Duration	Duration		☑
	Cycle	Cycle Time		
	Idle	Idle Time		
Interval Duration	MaxD	Maximum Duration		⧄
	MinD	Minimum Duration		
	ED	Estimated Duration		☑
Dependencies	FS	Finish-to-start	☑	☑
	FF	Finish-to-finish		☑
	SS	Start-to-start		☑
	SF	Start-to-finish		☑
Inflexible Constraints	MSO	Must Start On		☑
	MFO	Must Finish On		☑
Flexible Constraints	ASAP	As Soon As Possible	⧄	☑
	ALAP	As Late As Possible		☑
	FNET	Finish No Earlier Than		☑
	FNLT	Finish No Later Than		☑
	SNET	Start No Earlier Than	☑	☑
	SNLT	Start No Later Than		☑

3.1 TIME POINTS

When modeling a process, we usually need to specify time points or intervals of interest to the model. *Time Points* can be specified as *Absolute* (e.g., Tuesday April 11), *Periodic* (e.g., every Monday, 16:00hrs) [16] or *Relative* (e.g., 2 days after start).

3.2 INTERVALS

Interval size can either be explicitly specified as a duration, or implicitly derived from the interval's endpoints.In Figure 2, we depict generally accepted temporal components that capture the elapsed time occurring during the enactment of a sequence of activities or processes. Rectangles in Figure 2 represent temporal intervals. The other symbols (e.g., stars and arrows) represent time points. Note that the duration of the temporal components of a *Case* will be the sum of the durations of all the temporal components of its *Tasks*. In the remainder of this sub-section, we precisely define each temporal component in Figure 2 and present an axiomatization when possible.

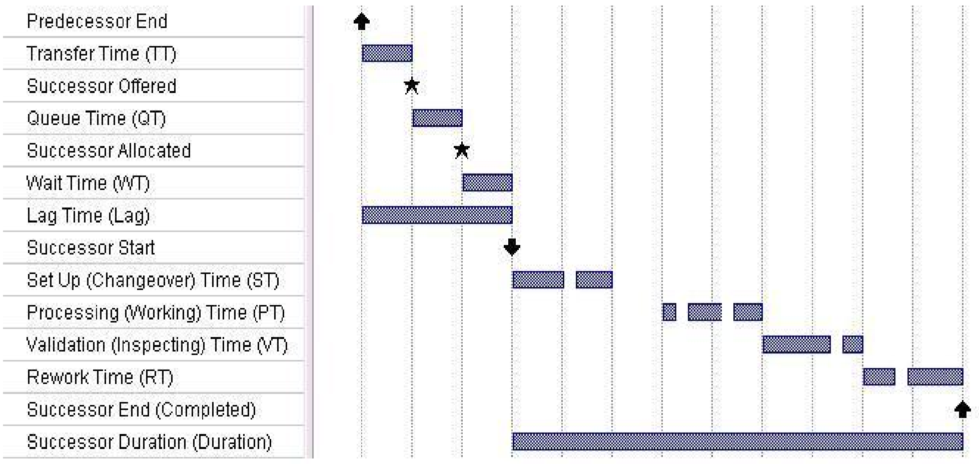


Fig. 2. Temporal components of a sequence of activities

At the point in time labeled *Successor Offered* in Figure 2, the Successor is offered to a single resource or to multiple resources [10]. We refer to the time between the end of the Predecessor and the offering of the Successor as the *Transfer Time*. The Transfer Time occurs over the interval TT. It may be the case that the Successor is offered at the same time as the Predecessor is completed. In such a case, there is no Transfer Time. If the Successor is offered to multiple resources, the Successor may spend time waiting for a resource to claim it. At which point in time, the Successor will become *Allocated*. The delay between the Successor being offered and the Successor being allocated is called the *Queue Time*. The Queue Time occurs over the interval QT. When the Successor is offered to a single resource, the offered time point and the allocated time point coincide and thus no Queue Time is incurred. The time between the Successor being allocated and it being actually started is called the *Wait Time*. The Wait Time occurs over the interval WT. It may be the case that the Successor is started at the same time that it is allocated, in which case no Wait Time is incurred. The *Lag Time* is the delay between the end of the Predecessor and the start of the Successor. It is the sum of the Transfer Time, Queue Time and Wait Time. The Lag Time occurs over the interval LAG. If there is a LAG, its length is equal to *Lag-Time:*

$$(\exists\ \alpha.\ \text{LAG}\{e\}\alpha) \rightarrow \text{length}(\text{LAG}) = \textit{Lag-Time} \qquad (2)$$

In formula (2) above, we use "LAG{e}α" in the antecedant to specify that the interval LAG is equal to an already existing interval α (i.e., LAG occurs). Technically, we cannot specify "LAG" by itself in the antecedent because it is a constant. If there is no LAG interval, then Lag-Time defaults to a value of zero:

$(\neg\exists\ \alpha.\ \text{LAG}\{e\}\alpha)\ \rightarrow \mathit{Lag\text{-}Time} = 0$ (3)

We have similar formulas for the intervals TT, QT, and WT:

$(\exists\ \alpha.\ \text{TT}\{e\}\alpha)\ \rightarrow \text{length(TT)} = \mathit{Transfer\text{-}Time}$ (4)

$(\exists\ \alpha.\ \text{QT}\{e\}\alpha)\ \rightarrow \text{length(QT)} = \mathit{Queue\text{-}Time}$ (5)

$(\exists\ \alpha.\ \text{WT}\{e\}\alpha)\ \rightarrow \text{length(WT)} = \mathit{Wait\text{-}Time}$ (6)

$(\neg\exists\ \alpha.\ \text{TT}\{e\}\alpha)\ \rightarrow \mathit{Transfer\text{-}Time} = 0$ (7)

$(\neg\exists\ \alpha.\ \text{QT}\{e\}\alpha)\ \rightarrow \mathit{Queue\text{-}Time} = 0$ (8)

$(\neg\exists\ \alpha.\ \text{WT}\{e\}\alpha)\ \rightarrow \mathit{Wait\text{-}Time} = 0$ (9)

The total lag time is the sum of the transfer, queue, and wait times:

$\text{Lag-Time} = \text{Transfer-Time} + \text{Queue-Time} + \text{Wait-Time}$ (10)

If there is a transfer time, it happens at the beginning of the Lag interval:

$(\exists\ \alpha.\ \text{TT}\{e\}\alpha)\ \rightarrow \text{TT}\{s,e\}\text{LAG}$ (11)

Since there may not be any transfer or wait time, the queue time can potentially occur anywhere within Lag:

$(\exists\ \alpha.\ \text{QT}\{e\}\alpha)\ \rightarrow \text{QT}\{s,d,f,e\}\text{LAG}$ (12)

The wait time can only happen at the end of the Lag:

$(\exists\ \alpha.\ \text{WT}\{e\}\alpha)\ \rightarrow \text{WT}\{f,e\}\text{LAG}$ (13)

Additionally, if we have transfer and queue times, they must follow each other:

$(\exists\ \alpha, \beta.\ [\text{TT}\{e\}\alpha\ \&\ \text{QT}\{e\}\beta]) \rightarrow\ \text{TT}\{m\}\text{QT}$ (14)

Similarly for queue and wait times:

$(\exists\ \alpha, \beta.\ [\text{QT}\{e\}\alpha\ \&\ \text{WT}\{e\}\beta]) \rightarrow\ \text{QT}\{m\}\text{WT}$ (15)

If there are transfer and wait times, and no queue time, then the transfer and wait times meet:

$(\exists\ \alpha, \beta, \neg\exists\ \gamma.\ [\text{TT}\{e\}\alpha\ \&\ \text{WT}\{e\}\beta\ \&\ \text{QT}\{e\}\gamma]) \rightarrow\ \text{TT}\{m\}\text{WT}$ (16)

The *Set Up Time* is the time expended prior to performing the actual work (e.g. lookup files, setup machines, etc.). In manufacturing this time is often referred to as *Changeover Time*. The Set Up Time occurs over the interval ST. The *Processing Time*, also referred to as the *Working Time*, is the time actually spent doing the work at hand. The Processing Time occurs over the interval PT. In the course of completing the Successor, a resource may spend time reviewing or inspecting the work done (*Validation Time*) and may even correct or redo some of the work (*Rework Time*). In manufacturing, Validation Time is often referred to as *Inspecting Time*. Note that we are referring here to validation/inspection and rework done by the same resource within the same Case or Task. As opposed to, for example, having another resource inspecting the work, this would constitute another process or activity. The Validation Time and the Rework Time take place over interval VT and RT respectively. In the course of completing the Successor, it may be *Suspended* and then later *Resumed*. The *Downtime* is the time elapsed between the Successor being Suspended and then Resumed. The Downtime occurs over interval DT. Downtime can occur between the intervals ST, PT, VT, and RT. For example in Figure 2, there is Downtime between ST and PT. There can also be Downtime within the ST, PT, VT, and RT intervals. For example in Figure 2, each has been suspended and resumed at least once. There can be arbitrarily many Downtime sub-intervals within ST, PT, VT and RT. Therefore, all the intervals ST, PT, VT, RT, and also DT can potentially be non-convex (i.e., a collection of intervals). Non-convex intervals cannot be directly represented in Allen's algebra. Future work will involve extending the algebra to deal with these intervals.

The *Successor Duration* is the elapsed time between the *Successor Start* and *End* points. The *Successor Duration* occurs over the interval *Duration.* The length of *Duration* is the sum of the lengths of the intervals ST, PT, VT, RT, and DT. Since the latter intervals are all convex, we cannot at this time provide axioms for *Duration.*

We refer to the time between *Predecessor End* and *Successor End* as the *Cycle Time* (sometimes also called the *Elapsed Time*). The Cycle Time is the sum of the Lag Time and the Duration. The Cycle Time occurs over interval CYCLE:

$(\exists\, \alpha.\ \text{CYCLE}\{e\}\alpha)\ \ \&\ \text{length}(\text{CYCLE}) = \textit{Cycle-Time}$ (17)

The *Idle Time* is the sum of the time within the cycle that is not dedicated to processing the item (i.e., length(*Cycle*) – length(*PT*)). Once again, *Idle Time* can potentially be non-convex and we do not represent it in Allen's algebra.

Studies have shown that Processing Time -where actual work is performed- can be remarkably low in the total Cycle Time. In addition, only a small portion of that amount is actually value-added time. Hence, each of the temporal components enumerated above are often scrutinized, and the subject of temporal optimization. The Lean methodology is known for addressing problems of time reduction in various processes [24].

3.3 INTERVAL DURATION

Although interval duration can be inferred from the time points, temporal dependencies, and constraints of the workflow specification, it is sometimes simpler to directly specify the duration of an interval. Two aspects of interval duration are of interest as constraints, namely the Maximum and Minimum duration. For example, we may want to model statements like: the acceptance process must last (at least/at most) 6 days. A *Maximum Duration* (MaxD) or *Minimum Duration* (MinD) can be directly specified as a constraint for a process or an activity or any other temporal components (e.g., MaxD = 6 for the acceptance process).

Another aspect of interval duration that is often used for simulation or, as a threshold in process and activity monitoring is the *Estimated Duration* (ED). The Estimated Duration is not a temporal dependency or constraint that regulates the enactment of business processes but is nevertheless a ubiquitous aspect of the temporal perspective of workflow specification.

In the following subsections, we differentiate between *Temporal Dependencies*, which are temporalities implied by the interrelationship between processes or activities, and *Temporal Constraints* which are time values specified for a process or an activity. Temporal dependencies and constraints are in effect scheduling parameters that must be met by the enactment of the process specification.

Note that it is possible for a collection of processes or activities to have, or not have, interrelationships among them and thus no Temporal Dependencies. In either case, the processes or activities may or may not have Temporal Constraints specified.

3.4 DEPENDENCIES

A *Temporal Dependency* is a relationship between two processes or activities in which one process or activity depends on the start or finish of another process or activity in order to begin or end. There are four types of Temporal Dependencies between a Predecessor and Successor:

- Finish-to-start (FS),
- Start-to-start (SS),
- Finish-to-finish (FF), and
- Start-to-finish (SF).

Temporal Dependencies can be further constrained with delays called Lead and Lag Time. *Lag Time* which was introduced above (e.g., in Figure 2) can also be used as a constraint to specify a delay between the finish of the predecessor and the start of the successor. For example, we need a delay between the finish of the activity "Paint wall" and the start of the next activity "Hang pictures" to allow the paint to dry. A Lag time can be specified as a duration. For example, if “Paint wall” has a 4 day duration, specifying a Lag time of 1 day would result in a 1 day delay to allow the paint to dry before starting the activity “Hang pictures”.

Lead Time causes the overlap of the successor and predecessor. In this case, the successor starts before the predecessor finishes. Lead time is useful when the successor requires a head start. It is usual to specify Lead time as negative lag, such as “-1” day. For example, for the activities "Construct walls" and "Plaster walls," we can use lead time to begin "Plaster walls" when "Construct walls" is half done.

As a constraint, Lag can be specified either by its length or relative to two tasks. In the following formulas, we represent the lag as a binary function. The function *lagBT(A,B)* returns the lag between tasks A and B. For example, a lag of 5 between tasks A and B is written as lagBT(A,B)=5. If lagBT(A,B) is negative, it represents a lead time.

Finish-to-start (FS)

A Finish-to-start temporal dependency between predecessor A and successor B means that B cannot start until A finishes. For example, the activity "Submit Paper" cannot start until "Write Paper" finishes. This is the most common type of temporal dependency.

$$\mathrm{lagBT}(A,B)=0 \rightarrow A\{p,m\}B \quad (18)$$

$$\mathrm{lagBT}(A,B)>0 \rightarrow [[B^- - A^+ \geq \mathrm{lagBT}(A,B)] \;\&\; A\{p\}B] \quad (19)$$

$$\mathrm{lagBT}(A,B)<0 \rightarrow [[B^- \geq (A^+ - |\,\mathrm{lagBT}(A,B)\,|)] \;\&\; AIB] \quad (20)$$

Start-to-start (SS)

A Start-to-start temporal dependency between predecessor A and successor B means that B cannot start until A starts. For example, if A is "Pour foundation" and B is "Level concrete," "Level concrete" cannot begin until "Pour foundation" begins.

$$\mathrm{lagBT}(A,B)=0 \rightarrow A\{p,m,o,fi,di,s,e,si\}B \quad (21)$$

$$\mathrm{lagBT}(A,B)>0 \rightarrow [[B^- - A^- \geq \mathrm{lagBT}(A,B)] \;\&\; A\{p,m,o,fi,di\}B] \quad (22)$$

$$\mathrm{lagBT}(A,B)<0 \rightarrow [[B^- \geq (A^- - |\,\mathrm{lagBT}(A,B)\,|)] \;\&\; AIB] \quad (23)$$

Finish-to-finish (FF)

A Finish-to-finish temporal dependency between predecessor A and successor B means that B cannot finish until A finishes. For example, if A is "Add wiring" and B is "Inspect electrical," "Inspect electrical" cannot finish until "Add wiring" finishes. Note that with a Finish-to-finish dependency, B can finish later than A's finish time.

$\mathrm{lagBT}(A,B)=0 \rightarrow A\{p,m,o,fi,s,e,d,f\}B$ (24)

$\mathrm{lagBT}(A,B)>0 \rightarrow [[B^{+} - A^{+} \geq \mathrm{lagBT}(A,B)] \,\&\, A\{p,m,o,s,d\}B]$ (25)

$\mathrm{lagBT}(A,B)<0 \rightarrow [[B^{+} \geq (A^{+} - |\mathrm{lagBT}(A,B)|)] \,\&\, AIB]$ (26)

Start-to-finish (SF)

A Start-to-finish temporal dependency between predecessor A and successor B means that B cannot finish until A starts. This temporal dependency type can be used for just-in-time scheduling up to a milestone or the project finish date to minimize the risk of an activity finishing late if its dependent activities slip.

$\mathrm{lagBT}(A,B)=0 \rightarrow [A\ I\text{-}\{pi\}\ B]$ (27)

$\mathrm{lagBT}(A,B)>0 \rightarrow [[B^{+} - A^{-} \geq \mathrm{lagBT}(A,B)] \,\&\, [A\ I\text{–}\{pi,mi\}\ B]]$ (28)

$\mathrm{lagBT}(A,B)<0 \rightarrow [[B^{+} \geq (A^{-} - |\mathrm{lagBT}(A,B)|)] \,\&\, AIB]$ (29)

3.5 CONSTRAINTS

Temporal Constraints can be specified to control the start or finish time of a Process or an Activity. These temporal constraints can be inflexible (*i.e.* tied to a specific value) or flexible (*i.e.* not tied to a specific value).

3.5.1 INFLEXIBLE TEMPORAL CONSTRAINTS

An *Inflexible Temporal Constraint* ties a process or an activity to a specific time point. The inflexible temporal constraints are:

- Must Start On (MSO) and
- Must Finish On (MFO).

Inflexible Temporal Constraints override any process or activity dependencies and restrict a process or an activity to a specified time. For example, an activity with a Must Start On (MSO) constraint for April 16 and a temporal dependency to another activity will always be scheduled for April 16 whether its predecessor finishes early or late. The inflexible temporal constraints are described below.

Must Start On (MSO)

This temporal constraint indicates the exact time (i.e., *Time-Point*) at which a process or activity A must be scheduled to begin. Other scheduling parameters such as temporal dependencies, lead or lag time and delay cannot affect scheduling the process or activity unless this requirement is met.

$A^{-} = \text{Time-Point}$ (30)

Must Finish On (MFO)

This temporal constraint indicates the exact time (i.e., *Time-Point*) at which a process or activity A must be scheduled to be completed. Other scheduling parameters such as temporal dependencies, lead or lag time, and delay cannot affect scheduling the process or activity unless this requirement is met.

$A^{+} = \text{Time-Point}$ (31)

3.5.2 FLEXIBLE TEMPORAL CONSTRAINTS

A *Flexible Temporal Constraint* does not specify a specific time point for a process or an activity, but rather imposes scheduling upper and lower bounds. The flexible temporal constraints are:

- As Soon As Possible (ASAP),
- As Late As Possible (ALAP),
- Finish No Later Than (FNLT),
- Start No Later Than (SNLT),
- Finish No Earlier Than (FNET), and
- Start No Earlier Than (SNET).

Flexible Temporal Constraints work in conjunction with temporal dependencies to make a process or activity occur as soon or as late as the process or activity dependency will allow. For example, a successor activity with an As Soon As Possible (ASAP) constraint and a finish-to-start dependency will be scheduled as soon as the predecessor activity finishes.

Constraints with moderate scheduling flexibility restrict an activity from starting or finishing before or after a specified time point. For example, a successor activity with a Start No Later Than (SNLT) constraint of June 15 and a finish-to-start dependency can begin any time its predecessor is finished up until June 15, but cannot be scheduled after June 15.

The flexible constraints are described in detail below.

As Soon As Possible (ASAP)

If an activity A is assigned an As Soon As Possible constraint, the enactment scheduler instantiates the activity as early as it consistently can. No additional time restrictions are put on the activity. This is the default constraint used (or assumed) by most process modelers when specifying workflows. Note that an activity will not be instantiated prior to the process (Case) being instantiated (i.e., *instantiation-time).*

$\text{minimize}(A^-)\ \ \&\ \ A^- \geq \textit{instantiation-time}$ (32)

As Late As Possible (ALAP)

This constraint is analogous to the ASAP constraint, but relative to the end of the Case. If an activity A is assigned an As Late As Possible constraint, the activity is scheduled as late as it consistently can, given other scheduling parameters. No additional time restrictions are put on the activity. If activities are scheduled from the Case *finish-time*, the enactment scheduler will determine how late the Case can start and still finish by the specified Case *finish-time*.

$\text{maximize}(A^+)\ \ \&\ \ A^+ \leq \textit{finish-time}$ (33)

Finish No Later Than (FNLT)

This constraint indicates the latest possible time point (i.e., *Time-Point*) that activity A is to be completed. It can be scheduled to finish on or before the specified time point. A predecessor activity cannot push a successor activity with an FNLT constraint past the constraint time point.

$A^+ \leq \text{Time-Point}$ (34)

Start No Later Than (SNLT)

This constraint indicates the latest possible time point (i.e., *Time-Point*) that activity A can begin. The activity can be scheduled to start on or before the specified time point. A predecessor activity cannot push a successor activity with an SNLT constraint past the constraint time point.

$A^- \leq \text{Time-Point}$ (35)

Finish No Earlier Than (FNET)

This constraint indicates the earliest possible time point (i.e., *Time-Point*) that activity A can be completed. The activity cannot be scheduled to finish any time before the specified time point.

$A^+ \geq \text{Time-Point}$ (36)

Start No Earlier Than (SNET)

This constraint indicates the earliest possible time point (i.e., *Time-Point*) that activity A can begin. The activity cannot be scheduled to start any time before the specified time point.

$A^- \geq \text{Time-Point}$ (37)

4 EXPRESSING TEMPORALITIES IN BPMN AND GANTT CHARTS

In this section, we present a brief discussion of the expressibility of BPMN and Gantt Charts (as portrayed by Microsoft Project) for business process modeling from the temporal perspective.

4.1 BPMN

The Business Modeling Notation (BPMN) [2] is an OMG standard which allows workflow modellers to depict workflow specifications in an easily understandable graphical representation. Although BPMN has been shown to be quite expressive with respect to other perspectives [7, 25], it is limited in its ability to express temporal dependencies and constraints.

In a BPMN process diagram, a sequence flow depicts a Finish-to-start (FS) temporal dependency between a predecessor and a successor due to the fact that in BPMN an activity is instantiated and assumed started as soon as it receives the token from a sequence path. Although not explicit, the sequence flow implies an As Soon As Possible (ASAP) constraint on the successor.

Absolute and Periodic Time Points (ATP, PTP) are directly expressible in BPMN using the Timedate and TimeCycle attributes of Events with a Timer trigger. Although no BPMN attribute allows the expression of Relative Time Points (RTP), the BPMN specification provides example diagrams annotated with relative time points (e.g., Figure 3). Note that figure 3 does not capture the notion of a Finish No Later Than (FNLT) constraint as such, since a BPMN boundary Intermediate Event can only lead to an Alternate flow.

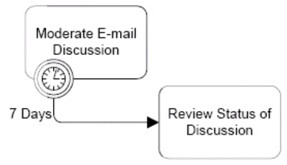


Fig. 3. BPMN example from p.44 of [2]

It is possible to depict a Start No Earlier Than (SNET) flexible constraint using an Intermediate Timer Event using the TimeDate attribute (see figure 4). Similarly, (as per above specification inconsistency) an expected delay, which corresponds to a Lag Time, can be depicted in BPMN using an Intermediate Event with a Timer Trigger using a duration or relative time point.

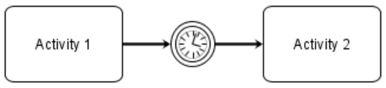


Fig. 4. A SNET and Potentially Lag Time in BPMN

4.2 GANTT CHARTS

Microsoft Project (MSProject) [26] is a product within the Microsoft Office Suite that allows the user to specify, plan and schedule various types of projects. MSProject is not a process modeling tool *per se*, as it cannot capture some of the basic constructs of process models such as decision (or-split) and loops. Nevertheless, MSProject can still be used to model certain simple classes of processes. For example, we can capture in MSProject all activities related to the opening of a new store. We can then use this MSProject file as a franchise template by instantiating the series of coordinated activities captured in the MSProject file for all future store openings.

A Gantt chart (as exemplified by MSProject) is an appropriate graphical representation for the temporal perspective. Many of the temporal dependencies and constraints can directly be depicted with a Gantt chart.

A summary of the temporal expressiveness of BPMN and MSProject, and a comparison between them appears in Table 1.

5. CONCLUSION & FUTURE WORK

Time is a critical dimension of business processes as it is directly related to customer satisfaction and cost reduction. The speed at which an organization delivers goods or services has a direct impact on customer satisfaction. Furthermore, time optimization is often an effective cost reduction strategy for many organizations. This important perspective is nevertheless lacking attention from current process standards initiatives.

In this paper, we provided a characterization of the temporal perspective of process modeling by providing a reference series of 31 generic recurring temporal constructs. For our temporal formalism, we chose Allen's which is the most popular in Artificial Intelligence. This characterization is independent of any specific modeling formalism or approach, and allows process models to be compared in a more meaningful manner on the basis of formal semantics.

Thus our characterization of the temporal perspective for workflow specifications provides a basis and objective means to evaluate the temporal expressiveness of various formalisms and tools. Also it opens the way to the integration of formal validation tools, such as constraint satisfaction or theorem proving systems, to verify the temporal satisfiability of the workflow specification.

As an example of the application of the proposed framework, we presented a brief discussion of the suitability of BPMN (an OMG specification) and Gantt Charts (as portrayed by Microsoft Project) for business process modeling from the temporal perspective. Gantt Charts have been found to be a suitable graphical representation for depicting the temporal perspective of process models.

These results have been used in a workflow management system which is currently under development. As a direct result of our deeper understanding of BPMN and MSProject's temporal semantics, we were able to semantically parse MSProject files as process model inputs and then visualise the process model via a BPMN depiction.

Allen's algebra only applies to convex time intervals (i.e., intervals containing no gaps). Future work will extend the algebra so that simple relationships can be specified between non-convex intervals. Specifically, we need to represent the relationship between two non-convex intervals. For example, one is before the other. Also, given two non-convex intervals, we need to represent the relationship between two sub-intervals where each is taken from one of the non-convex intervals. This will then allow us to represent the non-convex intervals Set Up, Processing, Validation, Rework, Down, and Idle times from Figure 2.

Future work will also involve using a constraint satisfaction system to automatically check the temporal consistency of a process. Given a process, we first capture its temporal attributes and constraints using the axioms presented in this paper. The resulting formulas are then input to the constraint satisfaction system which can detect inconsistencies. This exercise will also validate the soundness, feasibility, and usefulness of our proposed axiomatization.

REFERENCES

1. WFMC.: The Workflow Reference Model. WFMC-TC-1003, 19-Jan-95, 1.1, WfMC, (1995)

2. OMG.: Business Process Modeling Notation (BPMN) Specification. Version 1.0, OMG report: dtc/06-02-01, OMG, (2006)

3. OASIS.: Web Services Business Process Execution Language Version 2.0. OASIS report: wsbpel-specification-cd_Jan_25_2007, OASIS, (2006)

4. OMG.: Unified Modeling Language: Superstructure Specification. Version 2.1.1,. OMG report: formal/2007-02-03, OMG, (2007)

5. W.M.P. van der Aalst and A.H.M. ter Hofstede.: YAWL: Yet another workflow language. Information Systems, 30(4):245–275, (2005)

6. S. Jablonski and C. Bussler. Workflow Management: Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press, London, UK, (1996).

7. Workflow Patterns home page, retrieved March 24, 2007: http://www.workflowpatterns.com/

8. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros. A.P.: Workflow Patterns. Distributed and Parallel Databases 14(3) (2003) 5-51

9. Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow Data Patterns. QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane (2004)

10. Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow Resource Patterns. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, (2004)

11. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Workflow Exception Patterns. In: Dubois, E., Pohl, K. (eds.): Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE'06). Lecture

Notes in Computer Science, Vol. 4001. Springer-Verlag, Berlin Heidelberg New York (2006) 288-302

12. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. Communications of the ACM 26 (1983) 832-843

13. C. Combi and G. Pozzi.: Architectures for a Temporal Workflow Management System. In Proceedings of the 2004 ACM Symposium on Applied Computing (SAC'04), Nicosias, Cyprus, (2004)

14. C. Combi and G. Pozzi.: Task Scheduling for Temporal Workflow Management System. In Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning (TIME'06), IEEE, (2006)

15. C. Combi and G. Pozzi.: Towards Temporal Information Workflow Systems. In A. Olivé et al. (Eds.): ER 2002 Ws, LNCS 2784, pp. 13-25, Springer-Verlag, Berlin Heidelberg New York 2003.

16. C. Combi and G. Pozzi.: Temporal Conceptual Modelling of Workflows. In I.-Y Song et al. et al. (Eds.): ER 2003, LNCS 2813, pp. 59-76, Springer-Verlag, Berlin Heidelberg New York, (2003)

17. J. Eder, W. Gruber and E. Panagos.: Temporal Modeling of Workflows with Conditional Execution Paths, Database and Expert Systems Applications, Int. Conf., Springer, (2000) 243–253

18. J. Eder, and E. Paganos.: Managing Time in Workflow Systems. In Workflow Handbook 2001, Layna Fischer (Ed.), Future Strategies Inc., USA, (2001)

19. A. Meyer, S. McGough, N. Furmento, W. Lee, S. Newhouse and J. Darlington.: ICENI Dataflow and Workflow: Composition and Scheduling in Space and Time. In Proc of UK e-Science All Hands Meeting 2003, EPSRC, (2003)

20. A. Meyer, S. McGough, N. Furmento, W. Lee, M. Gulamali, S. Newhouse and J. Darlington.: Workflow Expression: Comparison of Spatial and Temporal Approaches. Workflow in Grid Systems Workshop, GGF-10, Berlin, (2004)

21. J.L. Zhao and E.A. Stohr.: Temporal Workflow Management in a Claim Handling Systems. In Georgakopoulos D. et al. (Eds.): Proceedings of the international joint conference on Work activities coordination and collaboration (WACC '99) . ACM Press, New York, NY, (1999) 187-195

22. R. Lu, S. Sadiq, V. Padmanabhan, and G Governatori.: Using a Temporal Constraint Network for Business Process Execution. In G. Dobbie and J. Bailey, editors, Proceedings of the 17th Australian Database Conference(ADC2006), volume 49 of Conferences in Research and Practice in Information Technology (CRPIT). Hobart, Australia, (2006)

23. N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar.: Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22, BPMcenter.org, (2006)

24. Lean Enterprise Institute home page, retrieved April 4, 2007: http://www.lean.org/

25. P.Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, and N. Russell.: On the Suitability of BPMN for Business Process Modelling. S. Dudstdarr, J.L. Fiadeiro, and A. Sheeth (Eds.): BPM2006, LNCS4102, pp161-176, Springer-Verlag, Berlin Heidelberg New York, (2006)

26. Microsoft Project, Online Documentation. Retrieved April 2, 2007. http://office.microsoft.com/enca/assistance/CH790018101033.aspx