View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by WestminsterResearch

UNIVERSITY OF
FORWARD
THINKING
WESTMINSTER

WestminsterResearch
http://www.westminster.ac.uk/westminsterresearch

Modeling and Optimizing Patient Flows

Chishti, I., Basukoski, A. and Chaussalet, T.J.

A paper presented at the *8th Annual International Conference on ICT: Big Data, Cloud & Security,* Singapore 21 to 22 Aug 2017.

The published version is available under DOI:10.5176/2251-2136_ICT-BDCS17.52

© Copyright 2017 GSTF.

The WestminsterResearch online digital archive at the University of Westminster aims to make the research output of the University available to a wider audience. Copyright and Moral Rights remain with the authors and/or copyright owners.

Whilst further distribution of specific materials from within this archive is forbidden, you may freely distribute the URL of WestminsterResearch: ((http://westminsterresearch.wmin.ac.uk/).

In case of abuse or copyright appearing without permission e-mail repository@westminster.ac.uk

Modeling and Optimizing Patient Flows

Irfan Chishti, Artie Basukoski, Thierry Chaussalet
Department of Computer Science
University of Westminster
London, UK
I.chishti@westminster.ac.uk

Abstract*—constructing a consistent process model and its simulation can be instrumental to be used in healthcare issues such as Consistent patient flow modeling. Current process modeling techniques used in healthcare are intuitive and imprecise such as flowcharts, unified modeling language activity diagram (UML AD) and business process modeling notation (BPMN). These techniques are vague in process description and cannot fully capture the complexities of the types of activities and types of temporal constraints between them. Additionally, to schedule patient flows; current modeling techniques does not offer any mechanism so healthcare relies on critical path method(CPM) and program evaluation review technique (PERT) that also have limitations i.e. finish-start barrier. It is imperative that temporal constraints between the start and/or end of a process needs to be specified, e.g., the start of A precedes the start (or end) of B, etc., however, these approaches failed to provide us with a mechanism for handling these temporal situations. This paper proposes a framework that provides enumeration of core concepts to describe a general knowledge base for Business and Healthcare domains. Algorithms are provided to represent the semantics of concepts i.e. based on their ontology. Furthermore, this logical basis is supported by Point graph (PG); a graphical tool, which has a formal translation to a point interval temporal logic (PITL) is used to simulate Patient flows for enhanced reasoning and correct representation. We will briefly evaluate an illustrative discharge patient flow example initially modeled using Unified Modeling Language Activity Diagram (UML AD) with the intention to compare with the technique presented here for its potential use to model patient flows.

Keywords—patient flow, business process modeling; point interval temporal logic; scheduling; optimizing; ontology; semantics; point graph

I. Introduction

The scale and complexity of healthcare sector have a huge impact on the level of patients' care. Therefore, attempts to model a whole hospital are rare [5], and the possible reason is the difficulty of representing the complexity of hospital activities within a simulation model [19]. However, it may be easier to select one part of hospital activity, for example modeling a patient flow separately. Because of this, there is an increasing recognition that developing a good systems' understanding of how a healthcare process works is an essential step to effective quality improvement [4, 20]. Such a systems' understanding is often lacking in healthcare [21].

A good system refers to a consistent model and a simulation that can be instrumental in addressing issues such as consistent patient flow modeling. On one hand side, current process modeling techniques used in healthcare are intuitive, imprecise and provide vague descriptions e.g. temporal flow of tasks and their corresponding relationships in the flow chart, unified modeling language activity diagram (UML AD) [18] and business process modeling notation (BPMN) [17]. Also, they use differing concepts such as UML AD use 'action' to represent an atomic unit of work and BPMN use 'task' to represent the same. On the other hand, they all lack quantitative representation of processes involved; therefore, healthcare sector use scheduling approaches such as critical path method (CPM) and program evaluation review technique (PERT) [15]. They only allow temporal relations between activities i.e. finish-start barrier and cannot fully capture the systems' complexities that address all available temporal constraints to construct the model which is precise enough.

If both qualitative and quantitative information provided with a precise description of concepts under one platform then it could provide an aid not only for correct modeling but can help in improved scheduling. A recent survey in [22] analyses current modeling techniques in terms of providing temporal perspective to capture complex temporal constraints to provide a consistent model. But reveals that current standard such as business process modeling notation (BPMN) variant TIME BPMN doesn't allow to model temporal constraints relating to the duration of the business process activities such as the activity lasts 'x' time units, and 'x' may be limited by a given interval. This survey lacks in identifying the temporal objects which are crucial if one needs to deal with the complexities of temporal constraints.

From the above, we have identified two issues that need addressing. First is the knowledge base used by the process modeling techniques which require a logical basis for the concepts/terms used and if provided this could improve its reasoning and representation [10]. Second the inference mechanism is missing that can be provided using the lexicon of the logic that offers a qualitative and quantitative representation of points and intervals of a system, e.g., the start of process A precedes the start (or end) of process B etc. A tremendous amount of work to solve such problems has been done; however, we find very little effort in overcoming the stated shortcomings of the traditional modeling and scheduling approaches. The state of the art framework proposed here is based on methodological approach by identifying the core concepts/terms used in current modeling techniques. Subsequently it provides formal semantics that could be used to construct a consistent model.

The rest of the paper is organized as follows; Section II describes the framework providing an enumeration of core concepts; based on their ontology they are formally defined i.e. semantics, to construct a consistent model based on a class of temporal logic i.e. point interval temporal logic (PITL) [1]. Section III provides the verification of the model presented in Section II. Section IV provides validation of the proposed system using point graph (PG) [2] which can provide enhanced reasoning. Section V discusses the application of the framework for evaluating a UML AD model of a discharge patient flow and compares it with the approach presented in this paper to construct a consistent model used for effectively scheduling; Section VI concludes this research paper.

II. FRAMEWORK

A. *Axiomatic System*

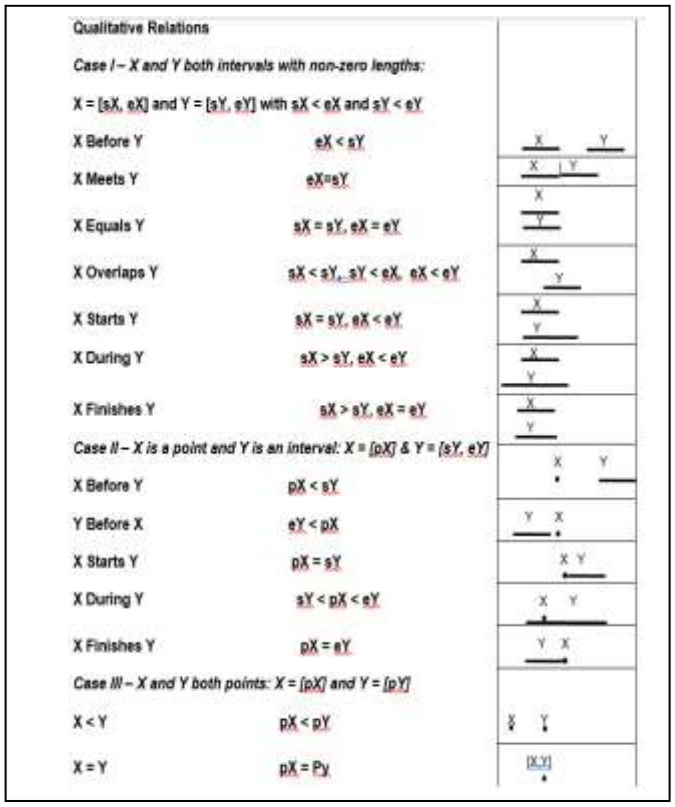
Intelligent control techniques such as temporal logic (TL), fuzzy logic (FL) and neural networks (NN) been used where the concepts/terms are ill-defined, complex, nonlinear, time-varying and stochastic. To use logic, we consider TL; as it provides consistency based on explicit axioms whereas modeling business and healthcare processes i.e. patient flows, deal with the practical problems from real life processes. We will provide an enumeration of core concepts and a corresponding ontology that can be used in current modeling techniques. Further explicit axioms i.e. consistent semantics, are provided based on point interval temporal logic (PITL). We use PITL to reason and represent a consistent but general knowledge base that can be used in business and healthcare domains to model processes/patient flows such as process/discharge, action/treatment, event/admission time etc.

In the spectrum of TL, many temporal theories provided but always left some unanswered questions. To maintain knowledge about intervals; a theory based on intervals taken as primitive and its 13 qualitative temporal relationships in time is presented in [12]. In [14], 'moment' i.e. an interval which cannot be broken down further, was introduced to be used as an alternative to point. McDermott introduced point algebra [6]; to model processes and events using temporal point as primitive. However, [1, 17] proposed a class of TL which considers a point, an interval and both point and interval as primitives, which we have used in this paper.

The following conventions constituting a range of connectives and quantifiers; will be used throughout in this paper in their standard interpretation as:

- $\wedge$ conjunction, $\vee$ disjunction, $\neg$ negation,
- $\Rightarrow$ implication, $\Leftrightarrow$ equivalence, $\vdash$ provable, $\vDash$ logical entailment
- $\forall$Universal quantifier, and $\exists$ Existential quantifier

Formalism provided in [2] considers a single time line. To show the qualitative temporal relation between any two intervals with nonzero lengths on the time line are related by one of the seven relationships as shown in fig. 1 using point interval temporal logic (PITL), i.e. case I specify relations between two intervals, case II specify relations between a point and an interval and case III specify relations between two points. We can also use PITL to reason and represent quantitative temporal information.



Qualitative Relations	
Case I – X and Y both intervals with non-zero lengths:	
X = [sX, eX] and Y = [sY, eY] with sX < eX and sY < eY	
X Before Y	eX < sY
X Meets Y	eX=sY
X Equals Y	sX = sY, eX = eY
X Overlaps Y	sX < sY, sY < eX, eX < eY
X Starts Y	sX = sY, eX < eY
X During Y	sX > sY, eX < eY
X Finishes Y	sX > sY, eX = eY
Case II – X is a point and Y is an interval: X = [pX] & Y = [sY, eY]	
X Before Y	pX < sY
Y Before X	eY < pX
X Starts Y	pX = sY
X During Y	sY < pX < eY
X Finishes Y	pX = eY
Case III – X and Y both points: X = [pX] and Y = [pY]	
X < Y	pX < pY
X = Y	pX = Py

Fig. 1. Qualitative temporal relations providing semantics

To construct a model, we use a model-theoretic approach [5] in this paper and use a schema [11] based on temporal basis for representing the idea of an abstract process i.e. theory, and subsequently, an instance will be used to model it i.e. verification of the abstract model.

B. *Abstract model*

In this section, we provide core concepts that will constitute an abstract model. An abstract model is considered to represent an abstract process which is comprised of core conceptual terms such as an atomic process i.e. task/action, process, sub-process, and special atomic process i.e. event, and temporal relationship to represent the flow between them. Core axioms are provided defining ontology of these concepts i.e. formal semantics. For the convenience of expression, we use an interval relation 'In' [13], relation 'Part' which accommodates both interval and point [16], and are given below.

$$\mathrm{In}(t_1,t_2) \Leftrightarrow \mathrm{Starts}(t_1,t_2) \vee \mathrm{During}(t_1,t_2) \vee \mathrm{Finishes}(t_1, t_2) \qquad \text{(R 1)}$$

$$\mathrm{Part}(t_1, t_2) \Leftrightarrow \mathrm{Equal}(t_1, t_2) \vee \mathrm{In}(t_1, t_2) \qquad \text{(R 2)}$$

Using a predicate 'Occurs' to represent an abstract process that occurs over a time element with a duration assignment and can be expressed as Occurs (a, t, D(t)). Where 'a' stands for a process symbol, 't' represents time element with a corresponding duration assignment 'D(t)'. In general, to provide axiomatization of abstract processes that applies to divisible intervals, non-divisible moments or time points using temporal relation R1 and R2 is given below:

$$\text{Occurs}(a, t, D(t)) \Rightarrow \forall t_1 \wedge D(t_1) > 0\ (\text{In}(t_1, t)) \Rightarrow \exists t_2 \wedge D(t) \geq 0\ (\text{Part}(t_2, t_1) \wedge \text{Occurs}(a, t_2)) \quad \text{(Ax. 1)}$$

1) *Definition 1-Atomic Process:* An atomic process is the basic element of the abstract model that may apply to non-divisible moments or time points [8]; using R1 it can be expressed as:

$$\text{Occurs}(a,t,D(t)) \Rightarrow \neg\exists(t_1 \wedge \text{In}(t_1,t) \wedge \text{Occurs}(a,t_1,D(t_1))) \quad \text{(Ax. 2)}$$

Ax. 2 defines an atomic process that occurs over the time moment or time point. If an atomic process associated with the time moment then it is referred to general terminologies used in business process modeling (BPM) and patient flow modeling (PFM) such as task, action, assessment of a patient respectively, which can be assigned to a single agent responsible for its completion. Once an atomic process is started, it continues to completion without reference to other atomic processes. It neither wait for other atomic processes to complete, nor initiating other atomic processes before its completion. An atomic process that is associated with a time point is notated here as special atomic process. It can be referred to BPM and PFM terminologies such as event, hospital i.e. patient admission and discharge time respectively.

2) *Definition 2-Business Process (Process):* In this paper, we will be using term business process (BP) and process interchangeably. A business process P is defined as a pair (A,R(A)) where

$$A=\{a_1, a_2,\ldots\ldots,a_n\} \quad \text{(Ax. 3)}$$

'*A*' is a finite set of atomic process names, where for all 'a_i' can be expressed as Occurs (a_i, t_i, $D(t_i)$). A process occurs over a time interval may comprise of several atomic processes for instance, a process occurs over time interval '*i*' which is decomposable; comprised of two-time elements 't_1' and 't_2' such that '$i = t_1 \oplus t_2$'; where 't_1' and 't_2' may refer to 2 atomic processes. A process P defined here refers to business processes of BPM and patient flows such as diagnosis, discharge and treatment processes of PFM that can be broken down further. Corresponding temporal relation between atomic processes is given as $R(A) = \{R(t_i, t_j) \mid 1 \leq i, j \leq n\}$. R(A) using 'Meets' relation. This defines a BP of logical conjunction and disjunction.

3) *Definition 3-Deduced Temporal Constraint:* In this paper, for conformance of temporal relations in R(A) we use DR(A) to denote the deduced temporal constraints which contains all the relations plus all the other relations that can be derived from R(A). These constraints are used to control the flow of the processes in the model and is given as:

$$DR(A) \vDash R(A) \quad \text{(Ax. 4)}$$

Let's prove it by deduction theorem, assume the following

a) Assumption 1: Every relation of DR(A) also belongs to a relation of R(A). Let R{(a)} be any relation of DR(A), if R{(a)} is a relation of DR(A), then it follows that R{(a)} also belongs to a relation of R(A). i.e. DR(A) ⊨ R(A).

b) Assumption 2: Let R{(a)} be any relation of R(A) that is also a relation of DR(A), i.e. DR(A)⊨R(A).

Ax. 4 is valid as it holds bidirectional and there exists at least one transitive relation containing R(A), and the disjunction of transitive relations is transitive. Hence the transitive closure of DR(A) is the disjunction of all transitive relations containing R(A).

C. *Properties of the Abstract Model.*

Soundness and completeness are two major issues in verifying a formal system; in our case its abstract model. Soundness refers to the correctness of the abstract process and completeness implicates that all the possible inferences can be derived by using the resolution algorithm in [3]. Formal definitions of these are presented here for convenience.

1) *Definition 4-Abstract Model is Sound:* An abstract model is called sound, if any temporal relation R(A) has been proved from a set of deduced temporal constraint DR(A) by a proof procedure such that

$$DR(A) \vdash R(A) \quad \text{(Ax. 5)}$$

It follows logically from DR(A), i.e., (Ax.4); DR(A)⊨R(A).

2) *Definition 5: Abstract Model is Complete:* An abstract model is called complete**,** if for any R(A), that follows logically from a given set of deduced temporal constraint DR(A), i.e. Ax. 4 and the proof procedure can prove R(A), i.e. Ax. 5. Now, we will follow a proof procedure presented in [3] and provide 2 theorems to prove the soundness and completeness of abstract model defined above.

a) Theorem I-Abstract Model is Sound: Proof: Given a set of deduced temporal constraints DR(A) and a goal R(A). Suppose we derived R(A) from DR(A) by the resolution theorem. We thus have DR(A) ⊢ R(A). We want to prove that the derivation is logically sound i.e. (Ax 4). Let us prove the theorem by the method of contradiction, presume that the consequent of DR(A)⊨R(A) is false, which means DR(A) ⊨ ¬ R(A). Thus, ¬ R(A) is satisfiable or true. To satisfy, we assign truth values (true/false) to all temporal relations that are used in R(A). We now claim that for such assignment, resolution of any two relations from DR(A) will be true. Thus, the resulting temporal relation even after exhaustion of all possible relations through resolution will not be false. Thus (Ax 5) is a contradiction. Hence, the assumption DR(A) ⊨¬ R(A) is false, and consequently (Ax 4) holds, and proves that abstract model is sound.

b) Theorem II-Abstract Model is Complete: Proof: Let R(A) be a temporal constraint such that from a given set of deduced temporal constraints DR(A), we have DR(A)⊨R(A) i.e. R(A) can be logically proved from DR(A). We must show there exists a proof procedure for R(A) i.e. (Ax 5). We shall prove it by the method of contradiction, let's assume DR(A) ⊢ R(A) is false that means DR(A) ⊢ ¬ R(A). In other words, R(A) is not derivable by a proof procedure from DR(A).

By using ground resolution theorem [3] that "if a set of ground derived temporal constraint is false, then the resolution closure of those deduced temporal constraints contains the 'false' deduced temporal constraint. Thus, $DR(A_1)$ is false, the resolution closure of $DR(A_1)$ yields the null relation, which causes a contradiction to (Ax 5). therefore, the assumption is wrong, and hence $DR(A) \vDash R(A)$ satisfies i.e. (Ax 4), and proves that abstract model is complete.

3) *Definition 6-Sub Process:* A process $P_1 = (A_1, R(A_1))$ is called a sub-process of a process P = (A, R(A)), iff

$$A_1 \subseteq A \qquad \text{(Ax. 6)}$$

$$DR(A_1) \subseteq DR(A) \qquad \text{(Ax. 7)}$$

So, we can say that $A \Leftrightarrow (A \wedge (\neg A \vee A_1))$ and $DR(A) \Leftrightarrow (DR(A) \wedge (\neg DR(A) \vee DR(A_1)))$. From Ax. 4, we can have $DR(A_1) \vDash R(A_1)$ therefore we could say that $(A_1, R(A_1))$ is a sub-process of a process (A, R(A)) i.e. P_1 is a sub-process of P.

In the next sub-section, we would provide the verification of the abstract model.

III. Verification of the Abstract Model

So far, the model introduced above is abstract. We may refer abstraction as theory or process type or process class and the interpretation as real world model or process token or process instance respectively. To achieve this, we formally define the meaning of the relationship of theory to model, type to token, or process class to the process instance. In this paper, we follow the axiomatic method [14], that defines theory as an axiomatic system i.e. abstract model/process, and a concrete realization as its interpretation in some real world domain i.e. real world model. By interpretation, we mean that a domain of real world can be chosen with its constant elements and predicates; taken in such a way that, with this enumeration of the primitive elements of the theory and their corresponding axioms are true propositions. Note that the existence of the real world interpretation ensures temporal consistency of the abstract model. To do this, we define instances of core elements of the abstract model along with a function that will map abstract model to concrete model.

1) *Definition 7-Abstract Model Instance:* An abstract model is a triad of (a, t, D(t) where $a \in \mathbf{A}$, $t \in \mathbf{T}$ and $D(t) \in \mathbb{R}$. Therefore, an abstract process instance can be defined as $a_R \in \mathbf{A_R}$, $t_R \in \mathbf{T_R}$ and $D(T(a_R) \in D(T_R) \geq 0$. For a real world domain containing abstract process instance symbols 'a_R' with its occurring time instance t_R, and duration assignment instance $D(t(a_R)$. The function $D(t(a_R))$ into $D(t_R)$ is just a real duration assignment of the occurrences of the abstract process. In this case, the real world model is an instance of the abstract process. Note that the existence of the real world interpretation ensures automatically the consistency of the abstract model. To do this, we introduce a mapping function from abstract to a real world.

2) *Definition 8-AtomicProcess Instance:* We use mapping ϕ for interpretation of the abstract model to its corresponding instances from a real world. We consider A_R a set of atomic process instances such that there exists a mapping ϕ between the set of atomic processes 'A' in the abstract model to those in the instance/realization and denote this mapping as

$$\phi(A) \rightarrow A_R \qquad \text{(Ax. 8)}$$

However, a set of atomic processes is comprised of process names, occurring time elements and their corresponding duration assignment. For convenience, mapping ϕ of a process element p is denoted as p_R, so that $\phi(p) = p_R$. Similarly, there exists mapping ϕ, from time instances t_R and duration assignment of corresponding instances of time elements $D(t_R)$ to the time elements t and duration assignments $D(t)$ in the abstract model; can be expressed as $\phi(t) = t_R$ and $D(\phi(t)) = D(t_R)$ where

$$\forall t \in t_R (D(t) = r) \Leftrightarrow D(\phi(t_R) = r) \in D(t_R) \qquad \text{(Ax. 9)}$$

This mapping function allows us to define instances of an atomic process class i.e. real-world action/task instance, and special atomic process class i.e. event instance.

3) *Definition 8-Business Process (Process) Instance:* A process instance P_R $(A_R, R(A_R))$ of the abstract model P(A,R(A)) is an actual realization, i.e. $\phi(A) \rightarrow A_R$, shows the mapping of a process of the abstract model to a real world process. However, $R(A_R)$ is a set of temporal relation instances such that there exists a mapping ϕ between the temporal relations R(A) in the abstract model and those in the instance, denoted the mapping as $\phi(R(A)) \rightarrow R(A_R)$. We define $R(A_R)$ in the interpretation, as

$$\forall t_i,t_j \in t(Meets(t_i,t_j) \in R(A) \Leftrightarrow Meets(\phi(t_i),\phi(t_j)) \in R(A_R) \text{ (Ax. 10)}$$

For a concrete process to be an instance of the process of the abstract model, we must be able to establish the mapping 'ϕ' from the processes of a concrete realization with the real-world times expressing their duration to the processes in the abstract model. But the real-world processes must satisfy the same sequencing constraints as are specified in the abstract model, i.e. $P_R = (A_R, R(A_R))$ must be temporally consistent.

4) *Definition 9-Sub-Process Instance:* A sub-process instance $P_1(A_1, R(A_1))$ of a sub-process of abstract model is the actual realization of P_{R1} $(A_{R1}, R(A_{R1}))$, we derive from the mapping of $\phi(A)$, i.e. $\phi(A_1) \rightarrow A_{R1}$. $R(A_{R1})$ is a set of temporal relation instances such that there exists a mapping between the temporal relation $R(A_1)$ in the abstract model to those in the instance and denote the mapping as $\phi(R(A_1)) \rightarrow R(A_{R1})$, we define $R(A_{R1})$ in the interpretation as

$$\forall t',t'' \in t\ (Meets(t',t'') \in R(A_1) \Leftrightarrow Meets(\phi(t'), \phi(t'')) \in R(A_{R1}) \qquad \text{(Ax.11)}$$

A sub-process of the abstract model is a part of the parent process of the abstract model, which is consistent such that there exists mapping ϕ of sub-process of abstract model to concrete, real-world sub-process and must be temporally consistent that must satisfy the sequencing constraints as are specified in the sub-process of the abstract model i.e. $P_{R1}=(A_{R1},R(A_{R1}))$.

To validate, we are going to present a visual representation of the abstract model using Point Graph (PG) is given in next section

IV. VALIDATION

A. *Process modeling using Point Graph*

Point Graph (PG) is based on PITL and represents the temporal statements. An inference engine based on PITL infers new temporal relations among system intervals/moments, identifies temporal ambiguities and errors (if present) in the system's specifications, and finally identifies the intervals of interest defined by the user. In a PG, a node represents a point (or a composite point), and an edge between two points represents one of the two temporal relations, *before and precedes*, between the two. Two or more points are represented as a composite point $[p_i;p_j;...;p_n]$, or a single node in a PG, if all are mapped to a single point on the timeline. The statements in PITL can be converted to an equivalent PG representation with the help of the corresponding analytic inequalities shown in fig. 1. For convenience. PG is defined PG [2] below.

1) Definition 10-Point Graph(PG): A Point Graph (PG), (V,E_A, D, T) is a directed graph with:

- V: Set of vertices with each node or vertex $v \in V$ representing point instant on the timeline. Points P_i, P_j, …, P_n are represented as a composite point $[P_i; P_j;…; P_n]$ if all are mapped to a single point on the line.
- E_A: Union of two sets of edges: $E_A = E \cup E_{\leq}$, where E: Set of edges with each edge $e_{12} \in E$, between two vertices v_1 and v_2, also denoted as (v_1, v_2), representing a relation '<'(before) between the two vertices, i.e., $(v_1<v_2)$. The edges in this set are called LT edges; and $E_{\leq}$: Set of edges with each edge $e_{12} \in E_{\leq}$, between two vertices v_1 and v_2, also denoted as (v_1, v_2), representing a relation '≤' (precedes) between the two vertices, i.e. $(v_1 \leq v_2)$. The edges in this set are called LE edges.
- D: Edge-length function (every edge is assigned a length): $E \in \Re$
- T: Vertex-stamp function (a vertex may or may not have stamp): $V \in \Re$.

This graphical representation with the underlying logical structure forms the link between the axiomatic system presented in Section II and III, and practical modeling techniques of business processes and patient flows of the healthcare sector. Keeping thus in mind, a PG of a process instance P_R is the same as that of a process P in the abstract model, such that connected with unique start and end vertices. These vertices correspond to the process start and end instances i.e. special atomic processes (events). For any given atomic process instance a_R, we accordingly term the two special atomic processes as start vertex (a_{RS}) and end vertex (a_{RE}). However, each time element *t* is denoted as a directed arc of the PG labeled by *t* representing its duration (if it is known).

2) Definition 11-Source and Sink Nodes: In PG A source node V_{in} and a sink node V_{out}

(a) $\forall v_i$, $v_i \in V$ such that ${}^*v= \varphi$, i.e., null set, connect the source node V_{in} to all v_i's by LE type edges (V_{in}, vi).

(b) $\forall v_i$, $v_i \in V$ such that $v^* = \varphi$, connect the sink node V_{out} to all v_i's by LE type edges (v_i, V_{out}).

3) Definition 12-Pre-set (Post-set): A pre-set (post-set) of a node contains all the nodes in V that have directed edges originating from (terminating at) them and terminating at (originating from) node v. The notation *v (v*) represents the pre-set (post-set) of a node v, where $\forall$ v_i, $v_i \in {}^*v$, then $(v_i, v) \in E_A$. Similarly, $\forall v_i$, $v_i \in v^*$, then $(v,v_i) \in E_A$.

The quantitative information is in the form of stamps for points and lengths for intervals. Sometimes the quantitative temporal information is not exact but is in the form of lower and upper bounds to actual values. PITL specification utilize virtual nodes, i.e. no temporal variable, to allow lower and upper bounds on the stamp (point) or the length (interval) as given in Table I. Since no value attached to it, so it doesn't appear in any PITL statements as shown in fig. 2 and 3 respectively.

TABLE I. QUANTITATIVE REPRESENTATION OF PITL

Temporal Objects	**Quantitative Representation**		
	PITL Expression	***LB Stamp***	***UB Stamp***
X; a point; [pX]	Stamp X=d; pX=d	Stamp X ≥ d; pX ≥ d	Stamp X ≤ d; pX ≤ d
Y; an interval; [sY,eY]	Length Y=d; eY-sY=d	Length Y ≥ d; ey – sy ≥ d	Length Y ≤ d ey – sy ≤ d

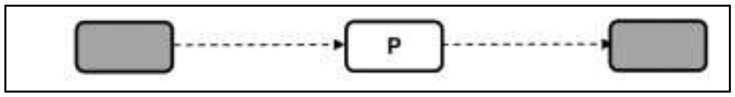


Fig. 2. Lower and Upper bounds on stamp

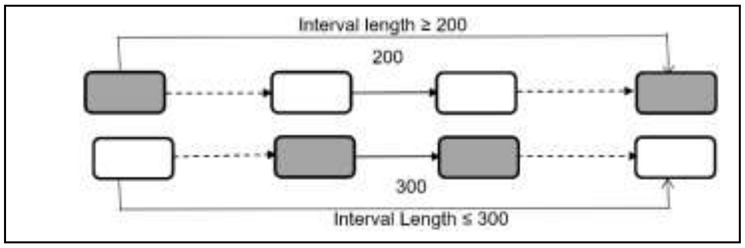


Fig. 3. Lower and Upper bounds on length(interval)

PG provides further algorithms to ensure a consistent flow; these algorithms are unification, branch/merge, i.e., branch folding and join folding and are defined below.

4) Definition 13-Unification:

a) Let vi = [pi;...;pn] and vj = [pj;...;pm] be two nodes in a PG representation. If there exists a point pk such that pk ∈ [pi;...;pn] and pk ∈ [pj;...;pm] or T(vi) = T(vj) then the two nodes are merged into a single composite node 'vi; vj' such that: vi;vj = [pi;...;pn] ∪ [pj;...;pm] where *vi;vj=*vi ∪ *vj and vi;vj* = vi* ∪ vj*.

The change in pre-and post-sets of unified nodes results in the redefinition of the set E_A in the PG representation. The nature of the edges involved in the unification does not change in the redefinition.

b) For all vi and vj ∈ V, such that T(vi) < T(vj) construct a directed edge from node vi to vj with D(vi, vj) = T(vj) – T(vi). The corresponding sets V, E_A, and the functions D, T are accordingly updated.

The unified PG is then scanned for branch and join nodes with quantitative information on their incoming and outgoing edges, respectively.

5) Definition 14-Branch Folding: PG folding process establishes new relations among system intervals, inferred through the quantitative analysis of the known relations specified by interval lengths and stamps [2]. A branch node vi ∈ V is said to be folded if, for all vj and vk in the post-set of vi.

a) D(vi, vj) < D(vi, vk) the edge from vi to vk, denoted as (vi, vk), is replaced by an edge (vj, vk) with D(vj,vk) = D(vi,vk) − D(vi,vj) and the vertex vk removed from the post-set.

b) D(vi, vj) = D(vi, vk), the two vertices vj and vk are merged into a single vertex with composite label 'vj;vk', and D(vi,vj;vk) =D(vi,vk){=D(vi,vj)}

c) vi has multiple edges to vj; if the edges are all of the same type (LT or LE) then only one edge is retained and others are deleted; if at least one of them is of type LT then it is retained, and others are deleted; if D(vi,vj) is defined for one of these edges, the value is assigned to the surviving edge. The corresponding sets V, E_A, and the functions T, D are accordingly updated. The methodology applies the branch folding process to all the original and newly created (formed during the folding process) branch nodes in the unified PG. The branch folding process, when applied to all the branch nodes of a PG, yields a partially folded PG having nodes with at most one outgoing edge with edge-length expression. Since all the edges in the PG may not have edge lengths associated with them, the branch folding may not result in a branch-node-free PG.

6) Definition 15-Join Folding: A join node vi ∈ V is said to be folded if, for all vj and vk in the pre-set of vi, with:

a) D(vj, vi) < D(vk, vi), the edge (vk, vi), is replaced by an edge (vk, vj) with D(vk, vj) = D(vk, vi) – D(vj, vi) and the vertex vk removed from the pre-set.

b) D(vj, vi) = D(vk, vi), the two vertices vj and vk are merged into a single vertex with composite label 'vj;vk,' and D(vj;vk, vi) = D(vk, vi) = D(vj, vi).

c) vi has multiple edges from vj. If the edges are all the same type (LT or LE), then only one edge is retained, and others are deleted. If at least one of them is of type LT, then it is retained, and others are deleted. If D(vj, vi) is defined for one of these edges, the value is assigned to the surviving edge. PG's corresponding sets V, E_A, and the functions T, D are accordingly updated. (Note: Case c is redundant if branch folding is applied before join folding).

B. *Scheduling using Point Graph*

PG specification provides scheduling feature to construct a consistent model. The scheduling algorithms applied to the PG representation calculate three parameters for each node in the PG. The parameter values are calculated by running two sets of algorithms, Forward* followed by Reverse*, on the graph [2]. The values of these parameters help determine the critical processes i.e. atomic processes and special atomic processes, and time floats/slacks for intervals in the system defined to be represented using PG. The three parameters are called *earliest occurrence (E_v)*, *Late occurrence (L_v)*, and *latest occurrence (T_v)* of a node 'v' and for convenience defined below.

1) Definition 16-Earliest Occurrence Time (E_v): E_v is the smallest time stamp on the node that satisfies the earliest occurrences of the preceding nodes requiring a forward traversal of the PG starting from the sink node, which by default is given a 0 value for the earliest occurrence of time [2] as shown in fig. 4.

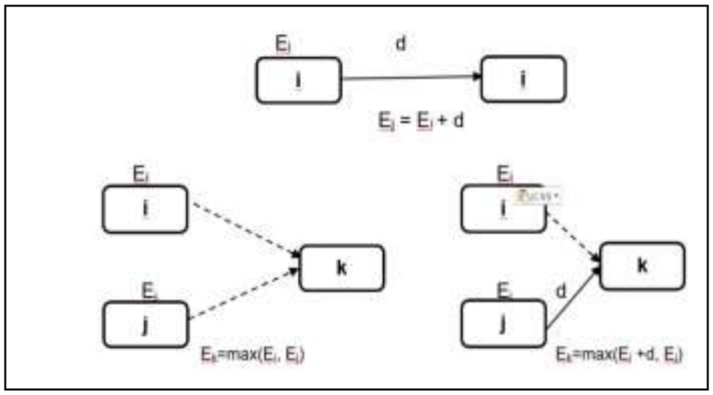


Fig. 4. Earliest occurrence time (E_V)

2) Definition 17-Late/Latest Occurrence Time (Lv/Tv): L_v (T_v) is the largest time stamp on the node that satisfies the earliest (latest) occurrences of the following nodes as shown in fig. 5. The calculation of these two parameters requires a reverse traversal of the PG starting from the sink node, which is by default initialized to the earliest occurrence time, calculated during the forward sweep, for both late and latest occurrence times [2].

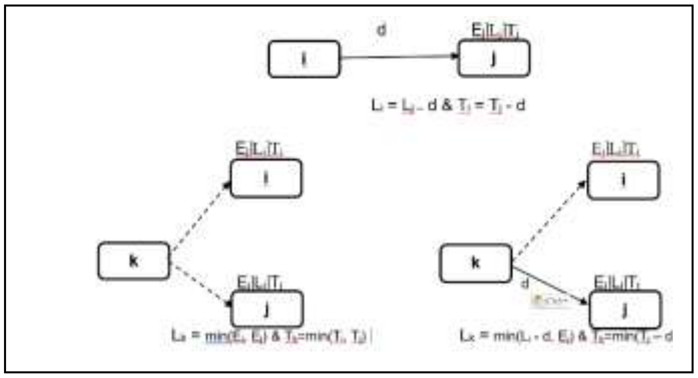

d
i
$E_j|L_j|T_j$
j
$L_i = L_j - d$ & $T_i = T_j - d$
$E_i|L_i|T_i$
i
k
$E_j|L_j|T_j$
j
L_k = min(E_i, E_j) & T_k=min(T_i, T_j)
$E_i|L_i|T_i$
i
k
d
$E_j|L_j|T_j$
j
L_k = min(L_j - d, E_i) & T_k=min(T_j – d

Fig. 5. Late/Latest occurrence time (E_V)

Identifying critical and non-critical processes can provide aid in optimizing a business process model or patient flow model.

3) Definition 18-Critical Activity: An activity is defined to be critical if:

a) delay in its start will cause a delay in the completion time of the entire system,

- for a special atomic process (point) i.e. event, v ∈ V, Ev=Tv;
- for an atomic process (moment) i.e. action/task [v1, v2], where v1, v2 ∈ V, v ∈ [v1, v2], Ev = Tv or

b) for an atomic process (moment), it 'Meets' another critical process. For a special atomic process (point), it 'Starts,' and/or 'Ends' another critical process or

c) an earliest (or latest) occurrence of its start node does not ensure an earliest (or latest) occurrence of its end node, i.e., for [v1,v2], Ev1 + D([v1,v2]) < Ev2, or Tv1+D([v1,v2]) < Tv2.

The condition (c) represents an atomic process that, for a given start-to-end system duration is required to start and end at specific times, to satisfy the preceding and following atomic process timings.

4) Definition 19-Total Float (TF) and Free Float (FF): Total Float (TF) is the difference between the maximum time available to perform an atomic process and its duration. Free Float (FF) is defined by if all the atomic processes start as early as possible. It is the excess time available over its duration [9].

a) Total float (TF) and free float (FF) for a non-critical special atomic process (point/event), v, are calculated from TFv=Tv – Ev and FFv = Lv – Ev.

b) Total float (TF) and free float (FF) for a non-critical atomic process [v1, v2], are calculated from:

- TF [v1,v2] = Tv2 − Ev2 = Tv1−Ev1
- FF [v1,v2] = Lv2 − Ev2 = Lv1−Ev1
- For all critical activities, TF = FF = 0

The difference between the actual duration and the required duration is called stretch float (SF) and is defined below.

5) Definition 20-Stretch Float (SF): For a critical process [v1, v2] of type defined in Definition 18(c), Stretch Float (SF) is defined to be the excess time available over the duration between the earliest occurrences of its start 'v1' and end 'v2' nodes, i.e., SF[v1,v2]=Ev2−Ev1−D([v1,v2]) or SF[v1,v2]=Tv2−Tv1−D(v1,v2). if SF exists, then it presents the following set of alternatives to a plan.

a) For a critical process [v1, v2] with SF, any one of the following may hold:

- Lv1 + D([v1,v2]) = Ev2;
- Tv1 + D([v1,v2]) = Lv2;
- Tv1+D([v1,v2]) =Ev2.

Then, the process is scheduled in the corresponding interval.

b) For the process Tv1 + D([v1,v2]) < Ev2: If started at the latest time still ends earlier than required by some of the preceding atomic processes, but the process' end time can be delayed by an amount equal to its SF after its start. Then, the process is stretched.

c) For a process that does not satisfy any conditions in part (a) and cannot be stretched, i.e. part (b); then the system cannot be planned without extending the start-to-end duration of the system. A dummy activity is created with length equal to the new duration (value of the objective function) and added to the list of the system processes. The analysis is applied to the new PG so obtained [2].

V. Application

Now we take an illustrative example from the real world scenario of a patient flow from a hospital to discharge a patient [8]. It has been modeled in Unified Modeling Language Activity Diagram (UML AD) as shown in fig. 6 below, which can systematically be converted into a PIL representation, and equivalent Point Graph notation.

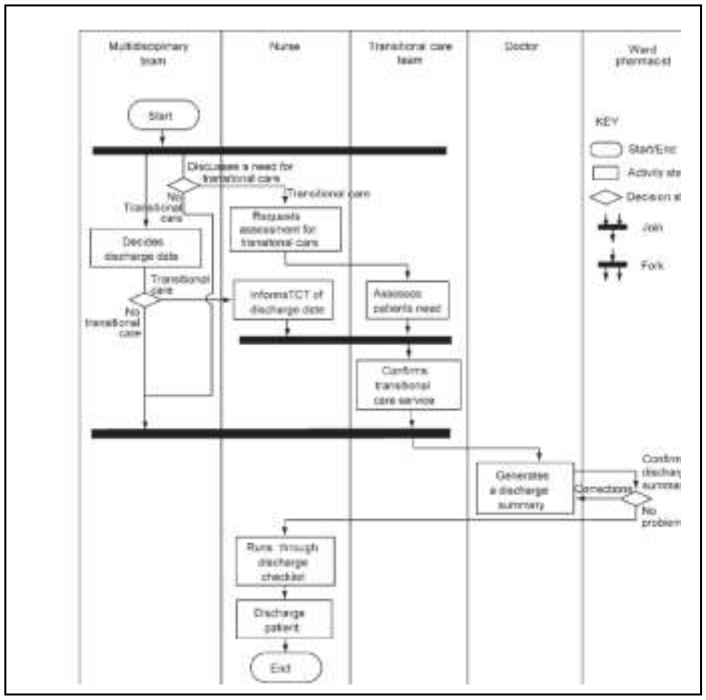


Fig. 6. Discharge patient flow modelled in UML AD

In UML AD although syntax of a model can usually be checked by a static inspection, dynamic semantics such as the conflicting constraints, an absence of deadlocks and livelocks cannot be completely verified until runtime. The problem here is that UML AD is not an executable language. Also, we understand UML AD is a conceptual modelling language and therefore chose not to represent quantitative information. But if provided then it can additionally be helpful not only for modelers for consistent modeling but also to assist in the scheduling of the processes involved in a model for its optimization.

After visual inspection of fig. 6 drawn in UML AD, we have found that the discharge patient flow has semantic incorrectness;

inaccurate and inconsistent. For instance, at a decision point, soon after the start of patient's discharge process, in case no transitional care needed, an atomic process i.e. decides discharge date, occurs, and after completion of this atomic process, another decision point has been placed to make the same decision which is inaccurate representation. Similarly, a decision about discharge date by the multi-disciplinary team has informed transitional care team occurs after a patient's needs been assessed. However, this is shown as one of the possibility of the decision, which causes semantic error. With the aid of PITL and PG, we can overcome many issues such as enhanced reasoning and consistent representation of processes of an organization and process optimization. To elaborate this, we transform the above example in natural language processing as shown in Table II below.

TABLE II. NATURAL LANGUAGE REPRESENTATION

Natural Language Representation		
Atomic Processes	***Description***	***PITL***
A	Deciding the discharge date	A meets D & F
B	Request for Assessment for transitional care	B meets C
C	Assess patient needs	C meets E
D	Informs TCT discharge date	D meets E
E	Confirms transitional care service	E meets F
F	Generate discharge summary	F meets G
G	Runs through the discharge checklist	G meets H
H	Patient discharged	eC precedes eD

In Table II, a PITL statement which is derived from the given scenario which UML AD representation of fig. 6 cannot capture i.e. a relation that eC precedes eD is derived, and shows inconsistency using PG representation in fig. 7.

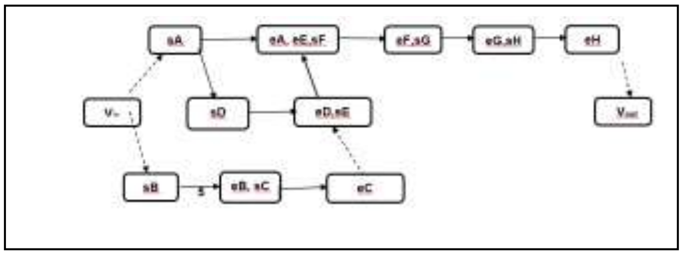


Fig. 7. Discharge patient flow (inconsistent) modeled in PG

The above investigation establishes that derived temporal relations can assist in capturing the complexities of a system, that may assist in constructing a consistent model. However, different researchers used either point or interval based systems which cannot derive all the temporal relations that PITL can.

The investigation conducted also establishes that both qualitative and quantitative temporal information, if available, is crucial not only in constructing a correct model but also to optimize it. This could be achieved as shown in fig. 8 by using PITL relations provided in fig. 1 with some assumptive but realistic quantitative information given in Table III.

TABLE III. SCHEDULING FOR PROCESS OPTIMIZATION

Scheduling						
Atomic Processes	***Dur***	***Ev***	***Tv***	***Critical***	***TF***	***FF***
A	18	0	18	Yes	0	0
B	5	0	5	Yes	0	0
C	10	5	15	Yes	0	0
D	15	0	15	Yes	0	0
E	3	15	18	Yes	0	0
F	2	18	20	Yes	0	0
G	2	20	22	Yes	0	0
H	2	22	24	Yes	0	0

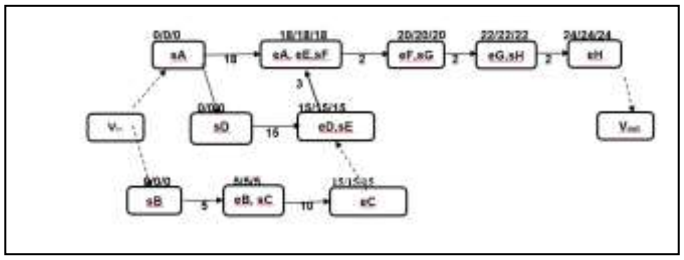


Fig. 8. A consistent and *optimized* discharged parient flow

We have simulated the axiomatic system based on PITL provided in Section II; to construct a consistent process model with the core concepts i.e. process and atomic process, special atomic process. With added features such as quantitative information handling of a qualitative represented consistently model; an optimized process model can be achieved which current modeling standards such as UML AD lacks to present.

VI. CONCLUSION & FUTURE WORK

In this paper, we have identified the gap of consistent business process modeling and its optimization which is present both in theory and practice. Rigorous investigation of current business process modeling standards i.e. UML AD and BPMN is conducted, and their ability to construct a precise model; which they are lacking. The reason is that they do not provide the formal semantics which leads to ambiguous models. Another issue identified is that these standards do not provide any aid to optimize by scheduling of resources such as time which is a pivotal feature and if provided it could improve the waiting times of patients in a hospital i.e. an Accident and Emergency department's waiting times, for instance.

The framework proposed in this paper uses a methodological approach. This presents core concepts and their corresponding ontology which in turn provides the formal semantics to construct a consistent model. Which is desired by most organizations in general but especially by healthcare sector to correctly model hospital patient flows.

This framework is general enough to be used as a knowledge base for business and patient flow modeling. This could also be used as an analytical tool to provide an insight for the modelling standards to report errors and ambiguity. Subsequently, it can be used to correct these errors and assist in removing any ambiguity attached with the models constructed using UML AD and/or BPMN by providing enhanced reasoning to assist in constructing a correct model. We have also used a graphical tool i.e. PG, that has not only the ability to validate the model which is proposed here but also provide additional features such as scheduling to optimize the resource usage within a process model.

As a continued effort to provide formal semantics to UML AD and BPMN, and subsequently unify them; in the future, we will transform the UML AD and BPMN into an equivalent PG representation that may become a standard for the both business and IT industries.

References

[1] A.K. Zaidi, On temporal logic programming using Petri nets, IEEE Transactions on SMC, 1999, 29(3), pp 245-254.

[2] A.K. Zaidi, and L.W. Wagenhals. Planning temporal events using point–interval logic. Mathematical and computer modeling, 2006, 43(9), pp.1229-1253.

[3] A. Konar. Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain, CRC Press, 1999.

[4] A. Newell, The knowledge level. Artificial Intelligence, 1982, 18(1), pp. 87–127.

[5] D. Fone, S. Hollinghurst, M. Temple, A. Round, N. Lester, A. Weightman, K. Roberts, E. Coyle, G. Bevan, S. Palmer. Systematic review of the use and value of computer simulation modelling in population health and health care delivery. Journal of Public Health. 2003; 25(4), pp. 325-35.

[6] D. Hilbert. *Grundlagen die geometrie*. Leipzig: Dritte Auflage, 1909.

[7] D. McDermott, A Temporal Logic for Reasoning about Processes and Plans, *Cognitive Sc*, 1982, 6, pp. 101-155

[8] G.T. Jun, J. Ward, Z. Morris, J. Clarkson. Health care process modeling: which method when?. International Journal for Quality in Health Care. 2009 Apr 10: mzp016.

[9] H.A.Taha. Operation Research: An Introduction. Collier Macmillan, 1992.

[10] I. Chishti, A grounding of business process modeling based on temporal logic, International Conference on Information Society, IEEE, Inc., Piscataway, NJ, USA, 2014, pp. 266-273.

[11] I. Chishti. Towards a general framework for business process modeling. Infonomics Society, 2014, vol 5(3), pp 443-453.

[12] J.F. Allen, Maintaining knowledge about temporal intervals, ACM, 1983, 26(11), pp.832-843.

[13] J.F. Allen, Towards a General Theory of Action and Time", Artif Intelligence, 1984, 23, pp.123-154.

[14] J.F. Allen. & P.J. Hayes, Moments and Points in an Interval-based Temporal-based Logic, Computational Intelligence, 1989, 5, pp225-238.

[15] J.J. Moder and C.R. Phillips. Project Management with CMP and PERT. Van Nostrand Reinhold Company; 1970.

[16] J. Ma, and B. Knight, A General Temporal Theory, the Computer Journal 1994, 37(2). pp 114-123.

[17] J. Recker. Opportunities and constraints: the current struggle with BPMN. Business Process Management Journal, 2010,16(1), pp.181-201.

[18] K. Lano. UML 2 Semantics and Applications. John Wiley & Sons, 2009

[19] M. M. Gunal and M. Pidd, Supporting smart thinking to improve hospital performance, proceedings of the DGHPSim Conference, Miami, FL, IEEE Comp Society Press, NJ, 2008, pp 1484–1489.

[20] N. Edwards, Can quality improvement be used to change the wider healthcare system? QualSaf Health Care, 2005, 14, pp. 75-75.

[21] PJ. Clarkson, P. Buckle, R. Coleman, D. Stubbs, J. Ward, J. Jarrett, R. Lane, J. Bound. Design for patient safety: a review of the effectiveness of design in the UK health service. Journal of Engineering Design. 2004, 15(2),1pp. 23-40.

[22] S, Cheikhrouhou, S. Kallel, N. Guermouche, and M. Jmaiel, "The temporal perspective in business process modeling: a survey and research challenges. *Service Oriented Computing and Applications*, 2015, *9*(1), pp.75-85.