Modeling Patient Flows: A Temporal Logic Approach

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

This is a copy of the final published version of an article published in GSTF Journal on Computing (JoC), [S.l.], v. 6, n. 1, May 2018. ISSN 2010-2283.

It is available from the publisher at:

http://dl6.globalstf.org/index.php/joc/article/view/1516

GSTF Journal on Computing (JoC) Vol.6 No.1 2018

Modeling Patient Flows: A Temporal Logic Approach

Irfan Chishti
Department of Computer Sciences
University of Westminster
London, UK
i.chishti@westminster.ac.uk

Artie Basukoski
Department of Computer Sciences
University of Westminster
London, UK
a.basukoski@westminster.ac.uk

Thierry Chaussalet
Department of Computer Sciences
University of Westminster
London, UK
chausst@westminster.ac.uk

Abstract— **Constructing a consistent process model can be instrumental in streamlining healthcare issues. Current process modeling techniques used in healthcare, such as flowcharts, unified modeling language activity diagram (UML AD), and business process modeling notation (BPMN) are intuitive and imprecise. These techniques are vague in process description and cannot fully capture the complexities of the types of activities and full extent of temporal constraints between them. Additionally, to schedule patient flows, current modeling techniques do 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 terms/concepts to describe a general knowledge basis for Business and Healthcare domains. Definitions are provided to present the semantics of concepts i.e. based on their ontology. Furthermore, this logical basis is supported by Point graph (PG) notation; a graphical tool, which has a formal translation to a point interval temporal logic (PITL), and is used to model Patient flows suitable for enhanced reasoning and correct representation. We will 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 the healthcare sector has an important impact on the timeliness and quality of patient care. Due to this complexity, attempts to model a whole hospital is rare [5] and the possible reason is the difficulty of representing the interdependencies of hospital activities within a model [21]. However, it may be easier to select one part of a hospital activity, for example modeling a patient flow separately and combine them within a consistent framework. 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[and [22]. However, this issue is either neglected fully or not received enough attention [23].

A good system refers to a consistent model which can be instrumental in addressing issues such as consistent patient flow modeling. The concepts of 'breakable' and 'non-breakable/atomic' work items adapted by the modeling techniques serve as standards for the industry such as unified modeling language activity diagram (UML AD) [20], and business process modeling notation (BPMN) [19], without providing a rigorous definition for those concepts i.e. terms, that could result in modelling a consistent model. On one hand, current process modeling techniques such as flow chart, UML AD and BPMN used in healthcare are intuitive, imprecise and provide vague descriptions of the tasks, temporal flow of the tasks and their corresponding relationships. Also, they use differing terms, for instance UML AD uses 'action' to represent an atomic unit of work, whereas BPMN uses 'task' to represent the same. On the other hand, they all lack quantitative representation of the processes involved such as duration and/or start/end time. Therefore, the healthcare sector uses scheduling approaches such as critical path method (CPM) and program evaluation review technique (PERT) [16]. These techniques only allow temporal relations between the activities such as finish-start barrier and cannot fully capture the systems' complexities that address all available temporal constraints to construct a consistent model.

If both qualitative and quantitative information is made available with a precise description of the terms under one platform, then it could provide aid not only for correct modeling but can help in improved scheduling and further optimizing the flows involved. A recent survey in [24] analyses current modeling techniques with the temporal perspective that capture complex temporal constraints to provide a consistent model. It reveals that even a variant of BPMN that attempts to provide temporal perspective i.e. TIME BPMN, doesn't allow to model temporal constraints those relates to the duration of the business process activities e.g. the activity lasts 'x' time units, and 'x' may be limited by a given interval. However, this survey lacks in identifying the temporal concepts such as point and interval and corresponding treatment of them which is 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

DOI: 10.5176/2251-3043_6.1.107

modeling techniques which require a logical basis for the 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. A novel approach that provides a state of the art methodological framework by identifying the core concepts used in the current modeling techniques. Subsequently it provides formal semantics that could be used to construct a consistent model.

The rest of the paper organized as Section II describes the framework providing an enumeration of core concepts based on their ontology, and they are formally defined to provide semantics to construct a consistent model based on a class of temporal logic known as point interval temporal logic (PITL) [1]. Section III presents the verification of the system presented in Section II. Section IV provides validation of the proposed system using a formal graphical tool called 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 for effectively scheduling; Section VI concludes this research paper.

II. FRAMEWORK

A. *Axiomatic System*

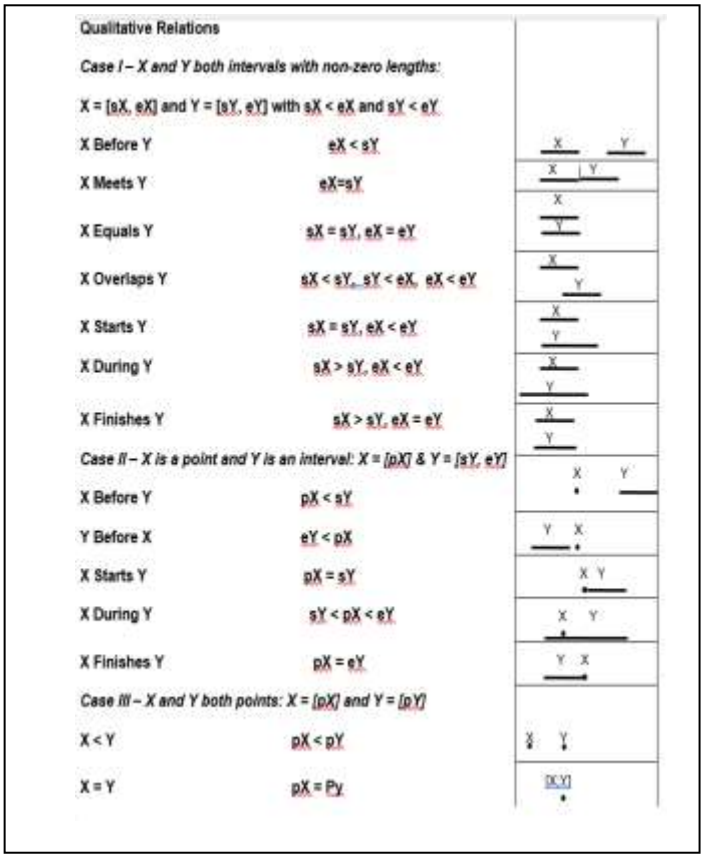
Temporal logic (TL) has been used to clarify when the concepts are poorly defined, complex, nonlinear, time-varying and stochastic. Business and healthcare processes known as patient flows deals with the practical problems in real life therefore modeling them needs clarity of concepts used. We consider TL to define the business process modeling concepts as it provides consistency based on explicit axioms and a proof theory. We will provide an enumeration comprised of core concepts for the proposed system. Explicit axioms i.e. consistent semantics, of those core concepts, are provided based on point interval temporal logic (PITL).

In the spectrum of TL, many temporal theories are provided but leave some unanswered questions. To maintain knowledge about intervals in [13] presents a class of TL based on intervals considered as primitive along with an introduction of its 13 qualitative temporal relationships. In [15] 'moment' i.e. an interval which cannot be broken down further, was introduced to be used as an alternative to a point. McDermott introduced a point algebra [7] to model processes and events using a temporal point as primitive. However, [1] and [17] proposed a class of TL which considers a point, an interval and both the point and interval as primitives and have used in this paper. 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.

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 a point interval temporal logic (PITL): 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

We use a model-theoretic approach [6] in this paper to build our system. A schema in [11] is adopted here for presenting the idea of an abstract process model that refers to theory and an instance that refers to the real-life model of the theory. Subsequently, a mapping is performed to provide verification of the system.

Now we provide an enumeration of the core concepts that will be used to build our system. The core concepts considered here are general enough that may be referred to the terms used by commercial modelling languages such as unified modeling language activity diagram, and notated here as atomic process, business process, sub-process, special atomic process and temporal constraints. Definitions of these lexicons are provided below

1) Definition 1 – Abstract Process Model: An abstract process model is a nonempty set of atomic processes that 'Meets' over time element having a duration assignment function, D(t), from time elements to non-negative real numbers. To formally define a predicate '*Occurs'* is used to represent an abstract process model and can be expressed as

$$\text{Occurs}\,(A, T, D(T)) \qquad \text{(Ax. 1)}$$

Where 'A' stands for a set of process symbols or names, 'T' represents set of time elements and D(T) represents a set of corresponding duration assignment. The temporal theory used here can be described in terms of the single 'Meets' relation. For the convenience of expression, we use an interval relation 'In' [14] and relation 'Part' which accommodates both the interval and point [18], are given below.

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

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

In general, to provide axiomatization of abstract processes that applies to divisible intervals, non-divisible moments or time points using temporal relation R2 is given below:

$$\text{Occurs}(a,t,D(t)) \Rightarrow \forall t_1 \wedge D(t_1) \geq 0 \ \text{Part}(t_1,t) \wedge \text{Occurs}(a, t_1) \quad \text{(Ax.2)}$$

2) Definition 2a - Atomic Process: An atomic process is the basic element of the abstract model that may apply to non-divisible moments or time points; using R1 but considering it represents atomicity that can be expressed as:

$$\text{Occurs}(a,t,D(t)) \Rightarrow \neg\exists(t_1 \wedge \text{In}(t_1,t) \wedge \text{Occurs}(a,t_1,D(t_1))) \quad \text{(Ax. 3)}$$

Ax. 3 defines occurrences of a process over a time moment or a time point. If time element is a moment then the atomic process is referred to the 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.

3) Definition 2b – Special Atomic Process: Ax.3 also defines occurrence of an atomic process over a time point and 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. In fig. 2, a_S and a_E represents two special atomic processes i.e. start and end events, occurring over an atomic process.

4) Definition 3 – Business Process: In this paper, we will be using term business process (BP) and process interchangeably. A business process P is a schema which contains a pair (A,R(A)) where

$$P = (A, R(A)) \qquad \text{(Ax.4)}$$

'*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 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.

a) Example: If we consider an assumptive but realistic example of a diagnosis process from an accident and emergency (A&E) department of a hospital. To diagnose, a physician requires clinical staff to conduct two examinations, one is collecting a blood sample and another is conducting an ultrasound examination of the patient. In this instance, diagnosis process is breakable that is comprised of two atomic processes i) taking a blood sample and ii) an ultrasound examination, we can associate the diagnosis process with an interval i.e. breakable, and subsequently referred to our process definition.

5) Definition 4 - 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:

$$DR(A) \vDash R(A) \qquad \text{(Ax. 5)}$$

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. $DR(A) \vDash R(A)$.

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

Ax. 5 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).

*6) Definition 5-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. 5, we can have $DR(A_1) \vDash R(A_1)$ therefore we could say that the P_1 is a sub-process of a process P.

B. 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 6-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

$$DR(A) \vdash R(A) \qquad \text{(Ax. 8)}$$

It follows logically from Ax.5 that is $DR(A) \vDash R(A)$.

2) Definition 7-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. 5 and the proof procedure can prove R(A), i.e. Ax. 8. Now, we will follow a proof procedure presented in [3] and provide 2 theorems to prove the soundness and completeness of the 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 $DR(A) \vdash R(A)$. We want to prove that the derivation is logically sound i.e. (Ax 5). Let us prove the theorem by the method of contradiction, presume that the consequent of $DR(A) \vDash R(A)$ is false, which means $DR(A) \vDash \neg R(A)$. Thus, $\neg R(A)$ 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 8) is a contradiction. Hence, the assumption $DR(A) \vDash \neg R(A)$ is false, and consequently (Ax 5) 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 $DR(A) \vDash R(A)$ 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. 8). We shall prove it by the method of contradiction, let's assume $DR(A) \vdash R(A)$ is false that means $DR(A) \vdash \neg R(A)$. 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 8). therefore, the assumption is wrong, and hence $DR(A) \vDash R(A)$ satisfies i.e. (Ax 5) and proves that abstract model is complete.

III. VERIFICATION OF THE ABSTRACT PROCESS 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 a 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 [11], 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 any domain from the real world can be chosen and mapped into 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 8-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 model 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 model. To do this, we introduce a mapping function ϕ from abstract process model to its corresponding instance from real world that can be expressed as:

$$\phi\ (a, t, D(t)) \Rightarrow (a_R, t_R, D(t(a_R)) \qquad \text{(Ax. 9)}$$

Note that the existence of the real world interpretation ensures automatically the consistency of the abstract model.

2) Definition 9a-Atomic Process Instance: For each atomic process instance a_R, we may write it as $a_R=[Name(a_R),t(a_R),D(t(a_R))]$. Where $Name(a_R)$ is a function from the set of atomic process instances A_R, $t(a_R)$ is a function from the set of process instances A_R to the set of time elements (instances) in t_R, $D(t(a_R)$ is a function from an atomic process *instances a_R to $D(t_R)$ duration assignment, where $D(t) > 0$ represents atomic processes i.e. tasks, actions and can expressed as*

$$\forall a_R \in A_R\,(Occurs(a_R, t(a_R), D(t(a_R)) > 0)) \qquad \text{(Ax. 10)}$$

Since each process instance is distinct, therefore, for any atomic process instance a_R, I impose that:

$$\forall a_R, t_R, D(t_R) \Rightarrow t_R = t(a_R), D(t_R)=D(t(a_R)) \qquad \text{(Ax. 11)}$$

3) Definition 9b-Special Atomic Process Instance: If occurring time element has duration of zero i.e. $D(t) = 0$ then it represents the special processes i.e. events, and we will use temporal predicate Occurs to denote special atomic process instance a_R occurs over time point (event) instances $t(a_R)$:

$$\forall\, t_R, D(t_R) \Rightarrow t_R = t(a_R), D(t(a_R)) = 0 \qquad \text{(Ax. 12)}$$

4) Definition 10-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

$$\forall t_i,t_j \in t(Meets(t_i,t_j) \in R(A) \Leftrightarrow Meets(\phi(t_i),\phi(t_j)) \in R(A_R) \qquad \text{(Ax. 13)}$$

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.

5) Definition 11-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

$$\exists t', t'' \in t_1 \Leftrightarrow \phi(t_1)=t_{1R1}, Meets(\phi(t'),\phi(t'')) \in R(A_{R1}) \qquad \text{(Ax. 14)}$$

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. $P_{R1}=(A_{R1},R(A_{R1}))$.

IV. VISUAL REPRESENTATION

A. *Process modeling using Point Graph*

Point Graph (PG) is based on PITL and is a diagrammatic representation of 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 [2] is defined below.

1) Definition 12-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 '$\leq$' (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 this 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 13-Source and Sink Nodes: In PG A source node V_{in} and a sink node V_{out}

- $\forall v_i$, $v_i \in V$ 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).

- $\forall v_{i,}\ v_i \in V$ 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 14-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

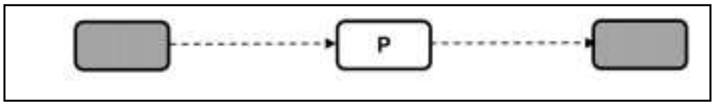


Fig. 2. Lower and Upper bounds on stamp

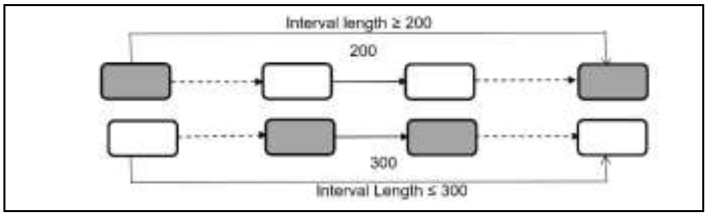


Fig. 3. Lower and Upper bounds on length(interval)

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 15-Unification: 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.

- 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 16-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 17-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 (Quantitative Information)

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' which is labelled with this information at the top of a node 'v' showing in a format earliest/late/latest i.e. times and for convenience defined below.

1) Definition 18-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.

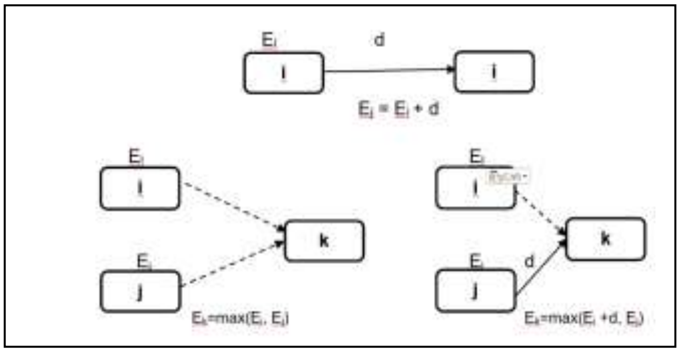


Fig. 4. Earliest occurrence time (E_V)

2) Definition 19-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].

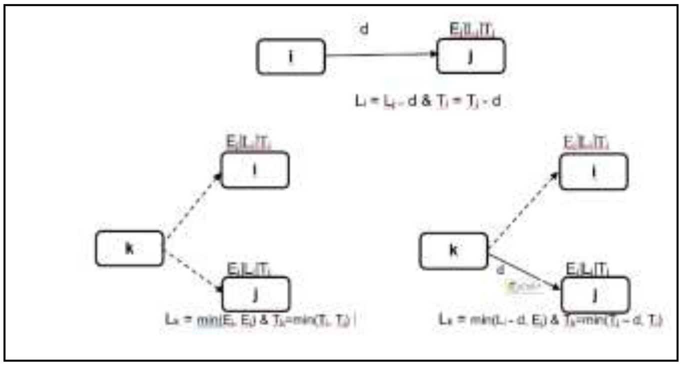


Fig. 5. Late/Latest occurrence time ($L_{V/}T_v$)

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

b) Atomic Process: 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 20-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 21-Stretch Float (SF): 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].

6) Example: A set of PITL statements representing processes and temporal constraints for a fictitious patient flow shown inf fig.6 where nodes are labelled with quantitative infrmation available in Table II.

TABLE II. CONSTRAINTS

Processes	***Length***	***PITL***
A	7	A meets B
B	7	-
C	7	C precedes B
D	4	C meets D
E	14	eE precedes eD

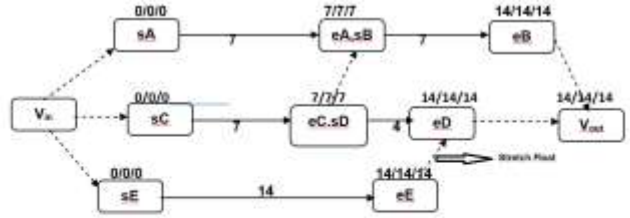


Fig. 6. A PH showing the stretch float

Constraint such as 'eE precedes eD' cannot be modelled in the traditional approaches due to their finish start barrier. Inference mechanism provided by the PITL using FindPath lower bound and upper bound algorithms establish undirected paths that leads in deriving relationships between two undirected nodes. representation. To explore more on this readers' can see the detail of this algorithm in [2]. To explore more on this readers' can see the detail of this algorithm in [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. 7. In UML AD although the 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.

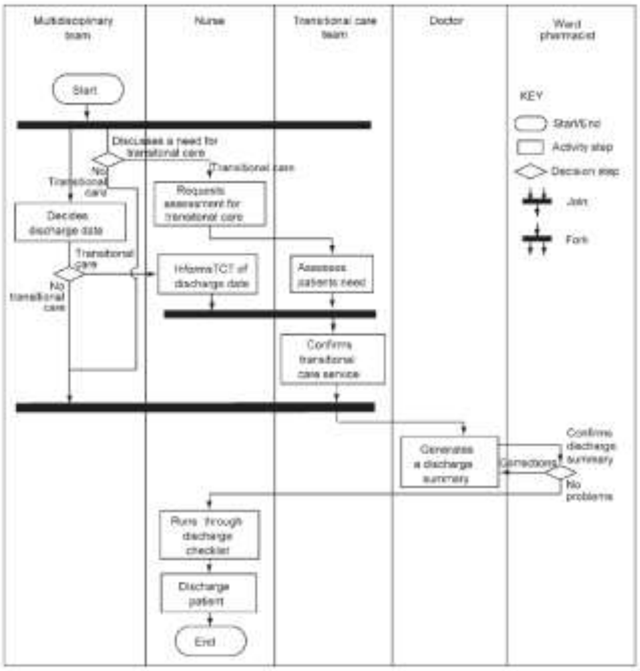


Fig. 7. Discharge patient flow modelled in UML AD

After visual inspection of fig. 7 drawn in UML AD, we have found that the discharge patient flow has some semantic incorrectness and inconsistencies. For instance, at a decision point, soon after the start of discharge process, in case when no transitional care is needed, an atomic process i.e. decides discharge date occurs, and after its completion, 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 the transitional care team, occurs after a patient's needs have been assessed. However, this is shown as one of the possibilities of the decision, which causes semantic error. With the aid of PITL and PG, we can overcome such issues. To elaborate this, we transform the above example in natural language processing as shown in Table III below.

TABLE III. NATURAL LANGUAGE REPRESENTATION

Natural Language Representation		
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 III, a PITL statement which is derived from the given scenario which UML AD representation of fig. 7 cannot capture i.e. a relation that eC precedes eD is derived, andshown in fig. 8 using PG.

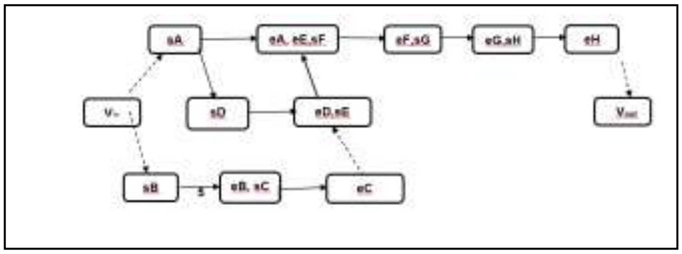


Fig. 8. Discharge patient flow modeled in PG

The above investigation establishes that derived temporal relations can assist in capturing the complexities of a system to construct a consistent model. Both qualitative and quantitative temporal information, if available, is crucial not only in constructing a correct model but also to optimize it. The Table IV provides some assumptive but realistic quantitative information and if combined with the qualitative information of PITL provided in fig. 1 that can assist in constructing a consistent PG as shown in fig. 9.

TABLE IV. 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

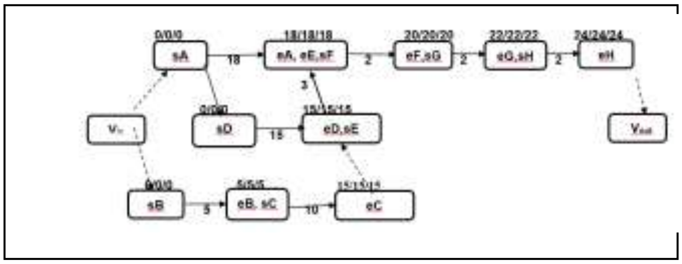


Fig. 9. Discharge patient flow modeled in PG

We have validated the axiomatic system based on PITL provided in Section II using PG. With the added feature of quantitative information provision and inference mechanism of PITL, PG can construct not only a consistent model but also can optimize the process model.

To show the functioning of sub-process, part of core concepts, defined in section II, we use a sub-type of PG named hierarchical point graph (HPG) [12], which is defined below for convenience.

1) Definition 22-Hierarchical Point Graph(HPG): A HPG=(PG, M) is a directed graph, where

- PG (V, E_A, D, T) (defined earlier) and
- M; A set of ordered pairs; it maps a pair of node in PG to another HPG; where M = {((a, b), HPG_{ab}) | a, b ∈ V and there is no directed path from b to a in PG, and HPG_{ab} is a Hierarchical Point Graph}.

Notice that for ordinary PGs, the set M is empty. Since M is a relation, it is possible to associate multiple HPGs with a single interval (a pair of nodes), which corresponds to the case when a process in the high-level PG can be substituted by multiple low-level parallel sub-processes. An interval/process in a PG can be substituted by yet another PG. This process of substitution can continue until the intervals/processes in the PG represent a system's primitive atomic processes i.e. moments.

HPGs encourage a modular and distributive approach to construct detailed models and plans separately from the high-level processes. Representing corresponding completion times used as constraints that bind different levels together. Fig. 10 shows a HPG example.

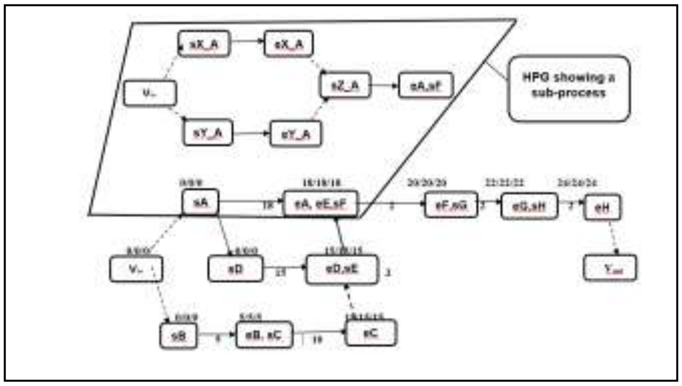


Fig. 10. A sub-process using HPG

VI. CONCLUSION

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' Accident and Emergency department, for instance.

The framework proposed in this paper uses a methodological approach. It 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. It also serves as an analytical tool to provide an insight for the modelling standards and report errors. This framework can be used to correct these errors and assist in removing any ambiguities attached with the models constructed using UML AD and/or BPMN. 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 resources 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 to provide formal semantics to these standards, and will be used as a unified modeling approach and 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] I. Mashood, A.K. Zaidi, and A.H. Levis. Project management using point graphs. *Systems Engineering*, 2009, 12(1), pp.36-54.

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

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

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

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

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

[18] J. Ma. “Ontological Considerations of Time, Meta-Predicates and Temporal Propositions”. *Applied Ontology*, 2007, 2(1), pp 37-66.

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

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

[21] 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.

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

[23] 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.

[24] 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.

[25] Irfan has completed BSc in Engineering from the Univerity of Punjab, Pakistan, MSc in Distributed Information Systems from Brunel University, London and MPhil in Computing and Information Systems from University of Greenwich, London. He has worked in the UK higher education sector in different academic and management roles for 20 years. Currently, he is a lecturer of Computer Science at University of Westminster and University of Greenwich and also a fellow member of Higher Education Academy (FHEA). His research interests are intelligent systems, temporal reasoning and representation, ontology, process modeling and mining techniques and their applications in healthcare domain.

Artie received his BSc in Computing Science from the University of Technology, Sydney (UTS). He then spent 10 years in industry, initially developing trading systems with the Union Bank of Switzerland in Sydney and Singapore, and later consolidating international credit card transaction systems as a Regional Project Manager for Citibank Singapore. With a desire to return to research he moved to the UK where he completed a PhD in Automated Reasoning from the University of Westminster. His current research interests are on the application of Process Mining, Data Mining and Machine Learning techniques in particular to the Health Care sector.

Thierry holds a PhD in Probability and Stochastic Processes from North Carolina State University (USA). He became lecturer in Decision Sciences and with growing interest in healthcare modelling, he founded the Health and Social Care Modelling Group in 1998 at the University of Westminster. His research interests are quantitative modelling of management processes and intelligent data driven methods for informed decision making. He serves on the Editorial Board of various healthcare modelling journals. He is member of the NIHR Peer Review panel, and was member of EPSRC Peer Review College 1996-2016 and expert evaluator for the EU FP7 ICT programme.