Transformation of UML Activity Diagram for Enhanced Reasoning

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

Transformation of UML Activity Diagram for Enhanced Reasoning

Irfan Chishti[1], Artie Basukoski[2], Thierry Chaussalet[3] and Neeraj Beeknoo[4]

[1,2,3] Department of Computer Science, University of Westminster, UK
[4] Kings; College Hospital, NHS Foundation Trust, UK
i.chishti@westminster.ac.uk

Abstract. IT industry has adopted unified modeling language activity diagram (UML-AD) as a de facto standard. UML AD facilitates modelers to graphically represent and document business processes to show the flow of activities and behavior of a system. However, UML AD has many drawbacks such as lack of formal semantics i.e. ontology used for the constructs based on intuition, that vaguely describes processes and no provision for verifiability. Petri Net (PN) has been around for decades and used to model the workflow systems but PNs and its variants are too complex for business process modelers with no prior experience. A logical foundation is desirable to construct a business process with a precision that facilitates in transforming UML AD into a formal mechanism supported by verifiability capabilities for enhanced reasoning. Therefore, in this paper, we will provide a framework that will provide formal definitions for UML AD core terms and constructs used for modeling, and subsequently transform them to formal representation called point graph(PG). This will provide an insight into UML AD and will improve the overall functionality required from a modeling tool. A case study is conducted at King's College Hospital trust' to improve their patient flows of an accident and emergency (A&E) department.

Keywords: Transformation, UML Activity Diagram, Ontology, logical foundation, Point Interval Temporal Logic, Point Graph.

1 Introduction

1.1 Motivation

Enterprises' from all domains including healthcare are increasingly depending upon process orientation. Processes are continuously evolving with the changing needs of the patients that can represent the system' organization and its corresponding flow. Unified modeling language activity diagram (UML-AD) is adopted as a standard for IT industry to model business processes (BPs) and workflows. UML-AD facilitates modelers to graphically represent, specify, construct and document BPs to show the systems' behavior. To represent system, UML-AD is comprised of constructs i.e. diagrammatical elements to illustrates the control flow behavior. The behavior depicts coordination of activities in a model that can be initiated due to changes or occurrences

internally or externally, for example, an activity may have finished executing or an event occurred external to the flow respectively [1]. In [2], it is claimed, that UML AD is chosen to model BPs because stakeholders can easily understand the graphical representation and can be used as a communication channel between them. Since its been adopted as a standard, its applications in different domain has increased. Healthcare sector specifically has shown interest in using UML (AD) to model patient flows (PFs) i.e. BPs.

On the contrary, UML AD has some limitations in expressing structural and temporal properties. This could be in the shape of the full extent of qualitative information and absence of quantitative information to represent a complete system. UML AD is not an execution language and not being validated [3], [4] especially in a real business environment. Due to this, modeler fails to detect errors that are only possible while monitoring the process execution. UML AD has always been poorly integrated, lacks expressiveness, which is a resultant of inadequate semantics and no provision for verifiability. UML AD is also examined by a series of authors in [5], [6] and [7] for its suitability, expressiveness, and adequacy and capabilities to model the resource perspective of BPs. UML AD is supposedly based on Petri Net semantics [8]. However, Petri Net and its variants are too complex to be used by business modelers.

Patient flows (PFs) contain healthcare and clinical knowledge that is used to ensure and improve quality of healthcare process model, and to reduce unnecessary variations in PFs e.g. discharge and admissions etc. to support decision making. National health service (NHS) is facing many system issues that affect their service. One of the system issue is the process longevity and other includes difficulty in meeting competing demands of under performed departments e.g. A&E etc. Unfortunately, UML AD doesn't fully comprehend the patient flow to provide qualitative and quantitative information to healthcare professionals.

1.2   Approach and Contribution

This paper provides a state-of-the-art framework by providing formal semantics to the core terms used in UML AD. That would improve description, correct construction and enhanced reasoning of the dynamics of large and complex systems. This would also facilitate a transformation of UML AD to a formal representation called point graph (PG) based on point interval logic (PIL) presented in [9]. PG can assist in providing qualitative and quantitative information. The following attributes of our approach explain better for the transformation:

i)   Identify the UML AD core terms,
ii)   Provide formal semantics by defining them.
iii)   Transform UML AD constructs to PG. The algorithm provided by PG such as Branch (Join) folding, unification, and PIL inference mechanism can be beneficial for precise modeling.
iv)   Construct a PG that is precise, simple and provides enhanced reasoning.

The transformation framework presented here would facilitate in describing, constructing of correct process models to satisfy the structural and temporal properties i.e. verification and validation. There has been no effort been made to transform UML AD to PG which makes this work unique and contribution to the knowledge in the field

of modeling business processes (BPs) and patient flows (PFs). PG can be used for modeling processes using nodes and edges to represent activities and their corresponding flow. These are fundamentally like those of UML ADs. Thus, PG seems a natural technique for modeling BPs and PFs.

The organization of rest of the paper includes a background study of the existing research in the field to identify the importance of a logical foundation and transformation of UML activity diagram. In section 3, UML-AD is being reviewed that will be followed by the introduction of the framework in Section 4. This will lay down a logical foundation for the terms used in UML AD and subsequently a transformation of them carried out in Section 5. Section 6 provides an application of the framework introduced in this paper by considering an illustrative example of a patient flow from the Kings College hospital. Section 7 is used to conclude the discussion and provide a brief on the future work.

2   Background

There are many reviews available in the literature to describe and categorize business process modeling methods and techniques based on the process representation [10] (see Fig. 1).
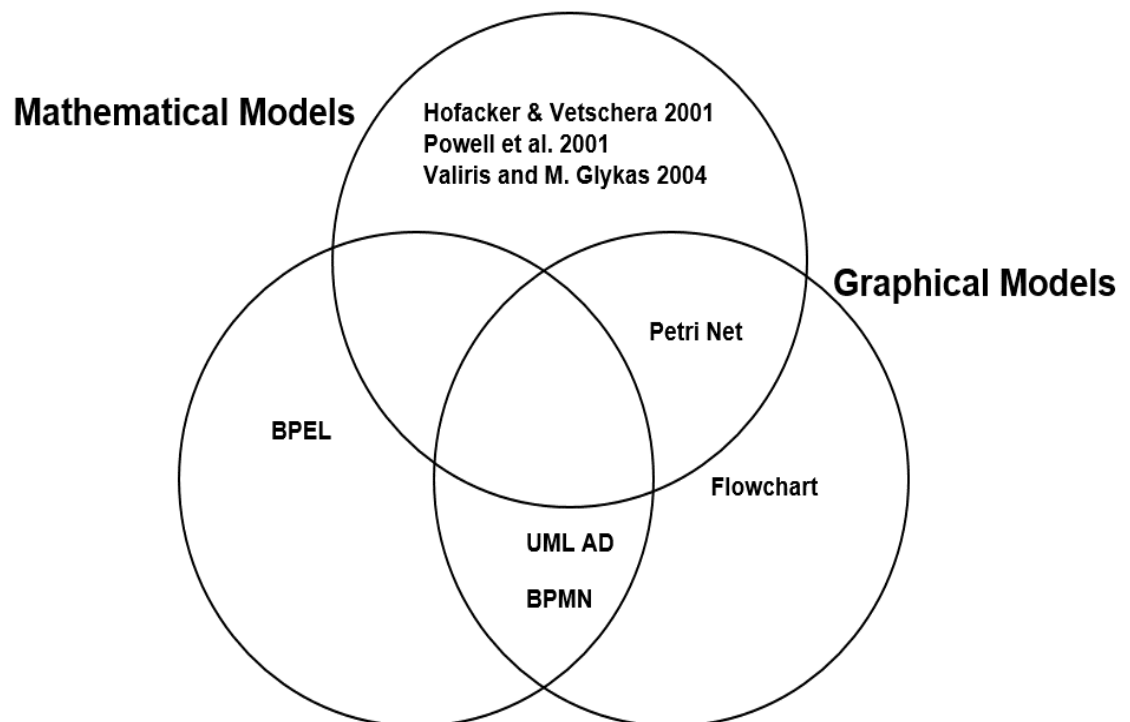
Mathematical Models

Hofacker & Vetschera 2001
Powell et al. 2001
Valiris and M. Glykas 2004

Graphical Models

Petri Net

BPEL

Flowchart

UML AD

BPMN

Fig. 1. Categorization of modeling techniques

Graphical techniques to model processes such as flowchart, UML-AD, and Business Process Modeling Notation (BPMN) are considered informal [11]. The reason is that they do not provide support for consistency for complex processes due to no formal underpinning [12]. However, new methods and tools may be developed for analysis using formal underpinning [13]. It may also construct models that are improved and consistent [14]. These techniques rely on analyst skills to perform any analysis [15]. However, formal underpinning to process modeling methodologies can overcome such problems [16]. Using mathematical models to represent real-life processes that may be complex and not possible to show decision points and parallel or hierarchical flow [17], [18] and [19]. On the contrary, a formal system may provide verification and validation of the business models that may satisfy all the temporal aspects [20].

In the literature, there are transformations of UML-AD to different formal techniques including variants of Petri Net (PN) in [21], [22] and [23]. UML-AD has no semantical support to represent the action(s) with upper and lower time bounds [24]. Whereas to develop a consistent model, concise semantics play a fundamental role in constructing a precise model that may assist in further analysis [25] including its corresponding temporal flow. Also, it doesn't provide extended qualitative and quantitative information to further analyze the business process model and modeler requires prior technical information to make use of its constructs [26].

Point interval temporal logic (PIL) can provide enhanced reasoning and representation about a business process e.g., precedence constraint that establishes whether a patient flow X precedes the patient flow Y extreme boundary points etc. Ultimately the industry is interested in improving the understanding of enterprises and their processes, facilitating process analysis and design and supporting process management in general and especially it's modeling.

3   UML-AD (Overview)

Unified Modeling Language activity diagram (UML-AD) metamodel provides informal semantics comprised of wide variety of constructs to graphically represent business processes. UML-AD notation based on the abstract syntax is called Activity that represents the systems' behavior. However, an Action serves as a fundamental unit of an activity that can have a set of inputs and outputs and may also change the state of the system. But the standard is burdened with a large set of concepts and corresponding constructs that are unused and also analyst find it cumbersome to construct a consistent model. To address such issue we identify the core artifacts of UML-AD from OMG 2015 to define them formally in the next sub-section.

3.1   Executable Node (Action)

Abstract syntax defined actions as executable nodes in the corresponding metamodel. UML-AD represents the activities diagrammatically using actions that can be invoked either directly i.e. call behaviour, or indirectly i.e. call operation. To start and complete an action there are input conditions that needs to be met. Additionally, an end of action may trigger proceeding executable nodes.

3.2 Edge

Edge is used between actions and activities to show the direction of the flow and may be labeled with guards.

3.3 Control Flow

Tokens are passed between the different action nodes of an activity and their corresponding flow is controlled by different control nodes. These control nodes are comprised of initial and final nodes, branching nodes i.e. decision and merge, and concurrent nodes i.e. fork and join nodes.

Initial and Final Node: To start an activity or an action, an initial node is used. It is possible to have more than one initial node of an activity to represent several flows. However, an activity is completed by accepting all the token on its inflow edges with the use of a final node construct having no outflow. UML-AD also provides a flow final node that terminates a flow. We will only consider initial and final nodes for the transformation purposes.

Decision and Merge Node: Branching behaviour of a system is represented by using decision and merge nodes. These nodes are in action when some of the activities have conditional flow. In situations such as where only one outflow is required i.e. xor split, a decision node is utilised where decision is based on the guards with no evaluation mechanism. In situations where flow of the system requires joining of inflows but no synchronisation (no tokens joining), a merge node is utilised to represent one outflow.

Fork and Join Node: Concurrent flow of a system is managed in UML-AD using fork and join nodes. Fork node is used to represent the split behaviour, where several outflows bearing replicated tokens are generated from a single inflow. However, to represent the synchronized behaviour of a system a join node is used. Both concurrent flows are represented using a same construct. With the help of both concurrent flow nodes, business processes with several instances are instantiated at the same time to manage the sequence.

We are not considering swimlanes for any transformation as it has no influence on the behavior of the system and (see Fig. 2) for the core artifacts identified above.

Concept	Visual Representation
Executable Node	
Edge	
Initial Node	
Final Node	
Decision/Merge Node	
Join/Fork Node	

Fig. 2. UML AD core artifacts

4   Framework

We have noted core terms used in UML-AD that needs formalization. To define the ontology of the core terms used in UML-AD, we have chosen a class of temporal logic that considers a point, interval and both point and interval as primitives known as point interval logic (PIL) (see Fig. 3). The choice of logic is mainly due to the reason that it facilitates in providing consistency based on explicit axioms supported by a proof theory.

We have adopted an axiomatic system [27] explained in subsections 4.1-4.3 for readers' convenience. Which is comprised of core building blocks like UML AD's core terms serves as an enumeration of the abstract process model. We would formally define them as explicit axioms i.e. consistent semantics, based on PIL.

Graphical Representation

Qualitative Relations	Point X to Point Y	Interval X to Interval Y	Point X to Interval Y	Interval X to Point Y
X Equals Y	X ● / Y ●	X → / Y →	N/A	N/A
X Before Y	X● Y●	X→ Y→	X● Y→	X→ Y●
X After Y	X● Y●	X→ Y→	X● Y→	X→ Y●
X Meets Y	N/A	X● Y→	N/A	X→● Y
X Met-by Y	NA	Y→ X→	Y→ X●	Y● X→
X Overlaps Y	N/A	X→ Y	N/A	N/A
X Overlapped-by Y	N/A	X→ Y→	N/A	N/A
X Starts Y	N/A	X→ Y→	X● Y→	N/A
X Started-by Y	N/A	X→ Y→	N/A	X→ Y●
X During Y	N/A	X→ Y→	X● Y→	N/A
X Contains Y	N/A	X→ Y→	N/A	X→ Y●
X Finishes Y	N/A	X→ Y→	X● Y→	N/A
X Finished-by Y	N/A	X→ Y→	N/A	X→ Y●

Fig. 3. Point Interval Relations

4.1 Abstract Process Model

An abstract process model provides an enumeration of core concepts notated here as an atomic process, process/sub-process, special atomic process and temporal constraints. An atomic process can be referred to a task/action associated with "moment" of [28], which is a non-divisible interval. Process and sub-process are referred to business process, sub-process, patient flow/ sub-patient flow respectively associated with breakable intervals. The special atomic process is referred to an event associated with the point. The corresponding temporal relationship between them is used to represent the constraints and flow. Axioms are provided defining the ontology of these concepts i.e. formal semantics.

Definition 1 - Abstract Process Model: To define abstract process model we would be using two relations 'In' [29] based on interval algebra stating that an activity can either starts, finishes another activity or occurs during the preceding activity i.e.

$$Starts \lor During \lor Finishes \qquad \text{(R1)}$$

However, a relation 'Part' accommodates both interval and point [30], and add equals relation to the existing three temporal relations i.e.

$$Equals \lor Starts \lor During \lor Finishes \qquad \text{(R2)}$$

A triad represents the abstract process model that is comprised of collection of process symbols $a_1, a_2, \ldots a_n$, and notated as 'A' with corresponding temporal occurrences $t_1, t_2, \ldots t_n$, notated as 'T' bearing some duration $D(t_1), D(t_2), \ldots, D(t_n)$. The temporal objects such as interval, moment and point used to define the ontology of the components of the abstract process model expressed as *(A, T, D (T))*. A predicate 'Occurring with the assistance of relation R2 given above used to define an abstract process model that may occur on any of the temporal element i.e. point, moment or interval:

$$Occurring(A,T,D(T)) \Rightarrow \forall t_1 \ (Part(t_1,T) \land Occurring(a_1,t_1,D(t_1))) \qquad \text{(Axiom 1)}$$

Definition 2 - Atomic (Process): We assume that the collection of process symbols are indivisible components i.e. atomic, of time that may be either moment or point. If the occurring activities are unbreakable then we notate them as atomic process as expressed below:

$$Occurring(A,T,D(T)) \Rightarrow \neg \exists t_1 \ (In \ (t_1,T) \land Occurring(a_1,t_1,D(t_1))) \qquad \text{(Axiom 2)}$$

The definition provided here establishes that an atomic process is indivisible with some positive duration. In real-life, domains such as business healthcare use terminologies to model a process for instance task or action registration of a patient. These terms are unbreakable activities and the definition provided here is general enough to subsume all of them. 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.

Definition 3 - Special Atomic (Process): Also, the Axiom 2 may refer to activities that have no or zero duration and known as temporal point. In real-life domains such as business healthcare use terminologies for instance as an event (describing a time stamp), patient's diagnostics start and finish time.

Definition 4 - Business Process (Process): To define a business process P, we consider it is occurring over a time interval which may be divisible by having some temporal relationships and can be expressed as

$$P = (A,R(A)) \qquad \text{(Axiom 3)}$$

Here, we assume that 'A' is comprised of atomic processes such as $a_1, a_2, \ldots, a_n$ occurring over a breakable interval that may comprised of two or more moments. We consider an example of a breakable interval 'I' having two moments 'i_1' and 'i_2' and can be expressed as $I = i_1 \oplus i_2$. In real-life, domains such as business and healthcare domains use terminologies for example business process and patient's admission etc. In addition, collection of the atomic processes present in a business process using a conjunction of temporal relations R (A).

Definition 5 - Derived Temporal Constraints: We assume that the derived temporal constraints notated as DTC constitute of 13 relations i.e. R(A), of interval algebra and all the possibly derived relations from them (see Fig. 2). These constraints are used to control the flow of the processes in the model and are given as:

$$R\,(A) \vDash DTC \qquad\qquad \text{(Axiom 4)}$$

Deduced temporal constraints provide inference mechanism and cover all possible relationships between two actions/activities and provides a formal semantics to the control nodes of UML-AD.

Definition 6 - Sub Process: To define sub-process here we consider that a P_1 is a sub process of a parent process P, if and only if

$$R\,(A_1) \subseteq DTC_1 \qquad\qquad \text{(Axiom 5)}$$

We can infer from Axiom 4 that any temporal relation(s) present between two or more atomic processes of a sub-process P_1 can be derived using the derived temporal constraints (definition 5) such that $DTC \vDash R(A_1)$. In real-life, domain such as business and health care use terminologies to represent group of activities that are part of its parent process for example sub-process and patient's diagnostics etc.

4.2 Verification

To provide the structural properties, soundness and completeness are considered in verifying abstract process model presented here. Soundness refers to the correctness and inferences derived from given relationships may be referred to completeness using the resolution algorithm [31].

Definition 7 – An Abstract Process model is sound if we can infer from R(A) that any relation R_1 (A_1) can be proved from derived temporal constraints such that

$$R\,(A) \vdash DTC \qquad\qquad \text{(Axiom 6)}$$

Definition 8 – An abstract Process model is complete if any temporal constraint of DTC is a logical consequent of a given set of temporal relations available in R(A) i.e. Axiom 4. Due to space limitation corresponding theorems are not provided here and interested readers are referred to [32].

4.3 Validation

We have defined an abstract process model (theory) and to validate this abstract process model there must exist a corresponding concrete realization as its real-life interpretation. Any application of real-life can be selected to transform in such a way that the provided axioms are true propositions to construct a consistent abstract process model. To define instances of the abstract process model and its core elements define above, we require a mapping mechanism that will interpret abstract process model to its corresponding concrete model. This also ensures that the terms defined here can be used to provide formal meaning to UML AD terms (real-life model).

Definition 9-Interpretation Function ϕ: To instantiate an abstract process model and its components, we need to establish that there exists a corresponding instance using an interpretation symbol ϕ.

Theorem 1-Interpretation: For any interpretation 'p' of the formal model presented here, there exists a corresponding unique instance p_R.

Definition 10- Abstract Process Model (Instance): We will use the interpretation ϕ to define an instance of the formal model presented here that may be expressed as $\phi(A, T, D(T)) \rightarrow (A_R, T_R, D(T(A_R)))$ and in real-life, time elements have duration that may be expressed as

$$D(T(A_R)) \in \mathbb{R}^+ \qquad \text{(Axiom 7)}$$

The instantiation of the formal model provided her make sure that there exists a consistent system.

Definition 11-AtomicProcess (Instance): Using theorem 1, for any atomic process 'a' there exists a unique instance of it represented as a_R. This definition may refer to real-world action/task instances occurring over a time moment. It bears some positive duration expressed as

$$D(t(a_R)) > 0 \qquad \text{(Axiom 8)}$$

Definition 12-Special Atomic Process (Instance): The instance of the special atomic process may refer to events of real-life having no or zero duration. Using duration assignment D (t), we can determine the length of the occurring temporal elements that is:

$$D(t(a_R)) = 0 \qquad \text{(Axiom 9)}$$

Definition 13-Business Process (Instance): To define an instance $P_R(A_R, R(A_R))$ of a business process P can be expressed as $\phi(A) \rightarrow A_R$ and $\phi(R(A)) \rightarrow R(A_R)$. A_R represents real-life atomic processes set and $R(A_R)$ establishes relationships between them.

$$\forall\ (t_i,\ t_j)\ (R\ (a_i,\ a_j) \rightarrow R(\phi(a_i),\phi(a_j))\ \in R(A_R) \qquad \text{(Axiom 10)}$$

The instance $P_R\ =\ (A_R,\ R(A_R))$ of a business process comply with the temporal constraints established in the abstract process model.

Definition 14-Sub-Process (Instance): To define a unique interpretation of sub-process $P_1 = (A_1, R\ (A_1))$, there must exists two or more than two unique atomic process instances in the respective of instantiation expressed as $P_{R1} = (A_{R1}, R\ (A_{R1}))$ such that $\phi(A_1, R\ (A_1)) \rightarrow (A_{R1}, R\ (A_{R1})$ can be expressed below

$$\exists\ t_1,\ t_2 \in T_1\ (R_1\ (a_1,\ a_2) \rightarrow R_1\ (\phi(a_1,\ a_2)) \in R\ (A_{R1}) \qquad \text{(Axiom 11)}$$

After providing formal definitions to the core terms now we will use a graphical tool called point graph to present visually the abstract model

4.4 Visual Representation

Point Graph (PG) is a graphical technique but formal in its nature, to represent the temporal statements based on simple node and edge notation. PG is supported by point interval logic (PIL) (see Fig. 3) and two nodes may represent an interval that can be graphically represented as before or precedes relation. For convenience. PG is defined below.

Definition 15 – Point Graph (PG): A graphical tool PG is defined as a tuple comprised of (V, E_A, D, and T). V represents collection of nodes to show closed interval. E_A is a conjunction of edges between two nodes to represent temporal relation 'before' i.e. E, and precedes i.e. $E_{\leq}$. Where D is used to represent the duration between nodes and each node represents on its own a time stamp i.e. T.

PG is also supported by three algorithms to support the control flow known as unification, branch folding and join folding. These algorithms not only assist in the completeness of a path using absolute and relative information available that facilitates in constructing a deadlock free business process model. For interested readers please see {33}.

After providing formal semantics to UML AD core terms it is possible now to transform them into PG that is given in next section.

5　Transformation

The specification of UML-AD does not define a mapping to any executable language, but the syntax should make the mapping possible. A methodical approach is adopted by initially identifying the core constructs in section 3 and corresponding formal definitions provided in section 4. This will enable us to perform a mapping between them. This methodical approach will also assist in the transformation of UML-AD into a formal representation. The resultant transformation is general enough to clearly construct a consistent business process model. Additionally, PG as a graphical tool will

assist in representing enhanced qualitative and quantitative information to construct a complete and error-free process model.

5.1 Executable Node (Action)

In UML-AD, executable node is graphically represented as rectangle (round-cornered). The definition provided for action in section 3 provides a logical basis for comparison with point graph (PG). The comparison shows that executable node is like a vertex i.e. atomic process, used in PG. In PG a pair of a rounded rectangle nodes represents an interval bounded by its start and end points expressed as sv_1 and ev_1. The vertex is labeled with earliest/late/latest time expression to show lower and upper bound of an action. Here we have seen the similarity in the graphical structure of both UML AD and PG, but PG provides added information and thus Executable node i.e. action, has been transformed (see Fig. 4).

5.2 Edge

In UML AD edge can be represented as a solid arrow between actions/activities. Whereas an edge in PG is used to represent the duration, direction, and flow of an action in a process. An edge in-between represents a qualitative relation which is '<' (before or meets) between the two vertices ($sv_1 < ev_1$). It may carry the duration label of the action/activity using a length function 'D'. UML AD Edge is similar to Edge in PG, however, PG provides more than a simple edge describing flow thus it has been transformed (see Fig. 4).

5.3 Initial/Final Node

To represent the beginning and a finishing of an activity, UML-AD provides two control flow constructs initial node and final node. The former construct is graphically represented as fully blackout circle, and the latter is graphically represented as target i.e. a solid circle inside a circle. PG offers the same facility denoted as Source and sink nodes. Two rounded rectangle vertices represent them and respective transformation (see Fig. 4).

5.4 Decision/Merge Node

In UML AD, a diamond is used to represent the branching/merge flow with a guard which is a condition. A token from inflow edge is transported to one of the several outflow edges i.e. mutually exclusion, that fulfils the condition (guards). Also, the same diamond construct can be used to express other conditional behaviors to express merging of the inflows resulting in one outflow but with no synchronization. Modelers have the discretion to choose the constructs so mainly they chose to represent a merge to conclude a branching behavior. PG provides simple to use node and edge to graphically represent an activity to express conditional behavior based on their temporal occurrence and labeled accordingly to establish the choice made. PG offers similar but enhanced reasoning with qualitative and quantitative information of a conditional flow and therefore decision/merge transformed into PG (see Fig. 4).

5.5 Fork/Join Node

In UML-AD, the concurrency is graphically represented as fully black-out bar which can be used either horizontally or vertically. In PG, branch(join) folding and unification algorithms are provided to support the concurrent behavior. The application of these algorithms ensures that a consistent model is constructed supported by an inference mechanism.. For example, using any of the aforementioned algorithm will perform an analysis on the PG representing a process with corresponding lengths to make a choice. UML AD's Fork/Join can be transformed into PG's branch(join) folding (see Fig.4).
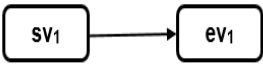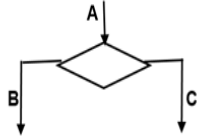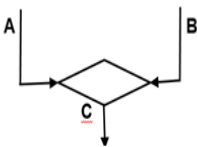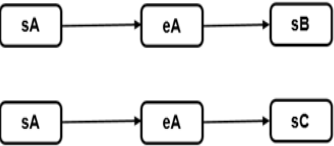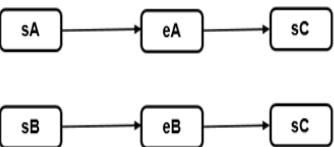
Fig. 4. Transformation of UML AD into PG

6 Application

A case study is being conducted in improving Kings' college hospital trust's patient flow modeling and a problem statement is considered here for the application of the framework.

6.1 Problem description

A trauma patient can come to accident and emergency (A&E) either via ambulance, or walk-in or brought in someone. This would take three parallel paths which are:

i. The trauma patient with minor injuries walked into A & E triage and in general, patient is seen by a specialist nurse followed by a consultation with a consultant. In case the patient requires further investigation then the patient is transferred to the ward. A number of tests such as MRI, CT Scan etc. may be carried out during the stay in the ward. This would lead to a treatment and ultimately the patient is discharged.

ii. The trauma patient who has driven into A&E by someone with minor major injury could be seen directly by a consultant especially if the hospital has been notified prior such as via 111. In general, the reported patient has records in the system transferred from 111 and could be referred to the high dependent unit (HDU). The patient could there either die or get better to be transferred to a general ward and thereafter discharged.

iii. A trauma patient brought in to A&E via ambulance with a major injury. The patient condition is critical and requires an urgent attention by a consultant. The patient could need an intervention and sent to the theatre for an emergency operation. After treatment, the patient would normally be discharged.

6.2 Solution

There are various combinations and permutation of getting access to consultants, nurses, diagnostics, theatres, wards, critical care etc. which UML AD lacks to reason and represent. To represent enhanced qualitative information, we have used a set of PIL statements. Table 1 exhibits quantitative and extended qualitative information of the patient flow (see Fig. 5). In table 1, natural language has been used to express atomic processes involved using PIL statements.

Table 1. PIL Representation with extended Reasoning

| Process Symbol	Natural Language Description	Quantitative Information	Qualitative Relationships
A	Patient seen by a consultant with minor trauma	9	A meet B
B	Transferred to ward for diagnosis & treatment	5	-
C	Patient seen by a consultant with minor major trauma	7	C meet D
D	Patient transferred to HDU for diagnosis & treatment	7	C precede B
E	Patient with major trauma sent to CCU	14	eD precede eE
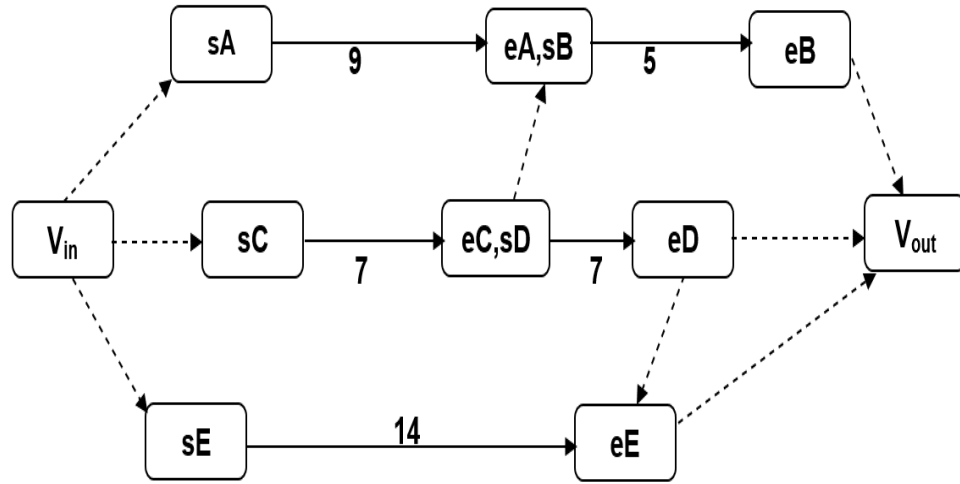
Fig. 5. Trauma Patient Pathways modeled using PG

In real-life the pathways are interchangeable, a patient can move from one pathway to another. For example, if a patient is reported with the minor major trauma and the patient is scheduled to be transferred to HDU, considering either patient's condition improves or HDU is no longer required and hence can move to the minor trauma patient pathway. Some of the judgments are subjective to human and machine factors and because of this, a patient is sent to CCU wrongly instead of the HDU.

There are two statements that are derived from the given scenario i.e. relations: C precedes B and eD precedes eE (see Fig. 5). These derived relations cannot be modeled using UML AD as there is no inference mechanism. However, PIL provides an inference mechanism and by using FindPath lower bound and upper bound algorithms can establish undirected paths that lead to deriving relationships between two undirected nodes. UML activity Diagram lacks in providing such extensive qualitative and quantitative representation of patient pathways which is desirable.

7 Conclusion and Future Work

Using a methodological approach, the state-of-the-art framework developed to enable a modeler to construct a consistent model facilitating enhanced reasoning. The transformation provided in this paper considered as the major contribution.

The development of the framework includes identifying the core artifacts of UML-AD and formally defining them. This provides a logical foundation and requires its application. The interpretation of the formal model achieved by formally defining corresponding instances. This ensures the verification and validation of the formal system developed. Additionally, we have compared and transformed the UML-AD core artifacts into PG. PG has formal translation into point interval logic (PIL).

Combined with extended qualitative relations that can be derived (see Fig. 3) and quantitative information that may be available, a detailed and precise model can be constructed. That is instrumental for NHS in dealing with huge delays in A&E that is

resulting in patients' fear of not receiving the right care at the right time. So far, no modeling technique or method has addressed the desired issue and this research can be used widely in addressing the bottlenecks NHS is facing. The system developed can also facilitates scheduling and optimization of patient flows but the intent here is to provide a transformation so it has not been included. A case study conducted at Kings' College Hospital (NHS trust) applied the framework to improve their patient flows and overall performance.

As part of the continued work on the project, a transformation of business process modeling notation (BPMN), (another OMG standard for business industry) into the formal system presented here that will be carried out in the future to show the novelty of the approach.

References

1. Object Management Group (OMG). Unified Modeling Language (UML), OMG, 2015 http://www.omg.org/spec/UML/2.5/, format/2015-03-01, last accessed 2018/04/04.
2. Nitto, Elisabetta Di, Luigi Lavazza, Marco Schiavoni, Emma Tracanella, and Michele Trombetta. Deriving executable process descriptions from UML. In *Proceedings of the 24th International Conference on Software Engineering*, pp. 155-165. ACM, (2002).
3. Van der Aalst, W.M.P., Process mining: a research agenda, Computers in Industry, Vol. 53, pp. 231-44, (2004a).
4. Van der Aalst, W.M.P., Workflow mining: discovering process models from event logs, 2004, IEEE Transactions on Knowledge and Data Engineering, Vol. 16 No. 9, pp. 1128-42, (2004b).
5. Dumas, Marlon, and Arthur HM Ter Hofstede. "UML activity diagrams as a workflow specification language." *International conference on the unified modeling language*. Springer, Berlin, Heidelberg, (2001).
6. Russell, N., van der Aalst, W.M., Ter Hofstede, A.H. and Wohed, P. On the suitability of UML 2.0 activity diagrams for business process modeling. In *Proceedings of the 3rd Asia-Pacific conference on Conceptual modeling-Volume 53* (pp. 95-104). Australian Computer Society, Inc. (2006).
7. Sarshar, K., & Loos, P. Modeling the Resource Perspective of Business Processes by UML Activity Diagram and Object Petri Net. In *Enterprise Modeling and Computing with UML* (pp. 203-229). IGI Global, (2007).
8. Wohed, P., van der Aalst, W.M., Dumas, M., ter Hofstede, A.H. and Russell, N. Pattern-based analysis of UML activity diagrams. *Beta, Research School for Operations Management and Logistics, Eindhoven*, (2005).
9. Zaidi, A.K. and Levis, A.H. TEMPER: a temporal programmer for time-sensitive control of discrete event systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *31*(6), pp.485-496, (2001).
10. Vergidis, K., Tiwari, A. and Majeed, B. Business process analysis and optimization: Beyond reengineering, 2008. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(1), pp.69-82 (2008).
11. Zakarian, A. Analysis of process models: A fuzzy logic approach. *The International Journal of Advanced Manufacturing Technology*, *17*(6), pp.444-452 (2001).
12. Valiris, G. and Glykas, M. Critical review of existing BPR methodologies: the need for a holistic approach. *Business process management journal*, *5*(1), pp.65-86, (1999).

13. Van der Aalst, W.M.P. and Van Hee, K.M. Business process redesign: a Petri-net-based approach. *Computers in industry*, *29*(1), pp.15-26, (1996).
14. Chishti, I., A grounding of business process modeling based on temporal logic. In *Information Society (i-Society), International Conference* (pp. 266-273) IEEE, USA (2014).
15. Phalp, K., & Shepperd, M. Quantitative analysis of static models of processes. *Journal of Systems and Software*, *52*(2-3), 105-112, (2000).
16. Aguilar-Saven, R.S. Business process modelling: Review and framework. *International Journal of production economics*, *90*(2), pp.129-149, (2004).
17. Hofacker, I. and Vetschera, R. Algorithmical approaches to business process design. *Computers & Operations Research*, *28*(13), pp.1253-1275, (2001).
18. Powell, S.G., Schwaninger, M. and Trimble, C. Measurement and control of business processes. *System Dynamics Review*, *17*(1), p.63-91, (2001).
19. Valiris, G. and Glykas, M. Business analysis metrics for business process redesign. Business Process Management Journal, 10(4), pp.445-480, (2004).
20. Cheikhrouhou, S., Kallel, S., Guermouche, N. and Jmaiel, M. The temporal perspective in business process modeling: a survey and research challenges. *Service Oriented Computing and Applications*, *9*(1), pp.75-85, (2015).
21. Baresi, L., & Pezze, M. On formalizing UML with high-level Petri nets. In *Concurrent object-oriented programming and petri nets* (pp. 276-304). Springer, Berlin, Heidelberg, (2001).
22. Kristensen, M. R., Jørgensen, J. B., Thomsen, P. G., & Jørgensen, S. B. Efficient sensitivity computation for nonlinear model predictive control. *IFAC Proceedings Volumes*, *37*(13), 567-572, (2004).
23. Petriu, D. C., & Shen, H. Applying the UML performance profile: Graph grammar-based derivation of LQN models from UML specifications (2002, April). In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation* (pp. 159-177). Springer, Berlin, Heidelberg, (2002).
24. Korherr, B. *Business Process Modelling-Languages, Goals, and Variabilities* (Doctoral dissertation, Vienna University of Technology, (2008).
25. Van der Aalst, W.M.P, Arthur H.M.T.H., and Mathias W. Business process management: A survey. *Business process management*. Springer Berlin Heidelberg, 1-12, (2003).
26. Bell, A. E. Death by UML fever: self-diagnosis and early treatment are crucial in the fight against UML fever", ACM Queue, Vol. 2(1), pp. 72-80, (2004).
27. Chishti, I. Towards a general framework for business process modeling. *Infonomics Society*, *5*(3), 443-453, (2014).
28. Allen, J. and Hayes, J. Moments and Points in an Interval-based Temporal based Logic, *Computational Intelligence*, 5(4), 225-238 (1989).
29. Allen, J. Towards a General Theory of Action and Time, Artificial Intelligence, 23, 123-154, (1984).
30. Ma, J., Knight, B., & Nissan, E. Temporal representation of state transitions. *AI EDAM*, *13*(2), 67-78, (1999)..
31. Konar, A. Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain", CRC press, (1999).
32. Chishti, I., Basukoski, A. and Chaussalet, T.J. *Modeling and Optimizing Patient Flows.* In: 8th Annual International Conference on ICT: Big Data, Cloud & Security, GSTF, Singapore, (2017).
33. Zaidi, A.K. and Wagenhals, L.W. Planning temporal events using point–interval logic. *Mathematical and computer modelling*, *43*(9), pp.1229-1253, (2006).