Indian J.Sci.Res. 17(2): 662-665, 2018 ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

COMPARISION OF MODELING TECHNIQUES

[1]S. Yamuna Rani, [2]Dr.Sumagna Patnaik
[1]Department of Computer Science, JNTU, Hyderabad
[2] Department of MCA,JB Institute of Engineering and Technology,Moinabad, Hyderabad

Abstract: This paper deals with different criteria of modeling techniques. An efficient management of the flow of work items between different people is an important prerequisite for a successful software engineering project. To make the software design process better different modeling techniques are used. Here different UML models are compared namely Sequence diagram, Activity diagram, State chart diagram ,Collaboration diagram and Petri Nets. The main goal of this paper is to show the common features and the main differences between above said models .a case study of hospital management is used to illustrate the concept.

I. Introduction

An efficient management of the flow of work items between different people is an important prerequisite for a successful software engineering project. To make the software design process better different modeling techniques are used. The UML specifies a modeling language that incorporates the object oriented community's consensus on core modeling concepts. However UML lacks a precise semantics that hinders error detection in the early stages of system development. Even worse ,there is no clear definition of consistency criteria among various UML notations.

Many graphical notations in UML only have informal English definitions[UML 99]and thus are error prone and cannot be formally analyzed. With the goal to better understand UML and to reveal potential problems in the current definitions of UML and to eventually formally analyze UML specification and designs, many researchers are currently trying to formally define the meanings of UML notation. Although there are many existing works on applying formal methods to UML specific notations is still rare. UML models are powerful enough to specify software system model visually and efficiently. They are lacking a formal semantics and difficult to apply, directly, mathematical techniques on UML model for system validation.

Petri Net is a software system modeling, which provides concurrency, synchronization and resource sharing behavior of a system. There are many theoretical results assoiciated with Petri Nets for analysis of such issues detection and performance analysis. And Petri nets are not easy for develops to draw, especially complex systems. Having difficulty to model objects and relation between objects.

" Effective modeling of complex concurrent systems requires formalism that can capture essential properties such as non determinism, synchronization and parallelism. Petri net offer a clean formalism for highly reusable and modular systems, but lacks general concurrency features. There have been a number of attempts to combine Petri Nets with object oriented concepts to profit from strengths of both the approaches." -John anil sadhana.

II Modeling techniques

Sequence Diagram

Sequence diagrams are based on message sequence charts. Objects are shown as boxes with dashed lines extending vertically below them. As these lines move down the page, they represent the passing of time. Arrows across the page show the sequences of messages passed between objects as time passes. Periods of activity by an object may be shown as a bar, called an activation, overlying its dashed life line. Each sequence diagram represents one or more routes through the unfolding of a use case (high level) or of an operation in a class (low level). If a single route is shown, one particular set of conditions is being assumed. Such a set of conditions is termed a scenario.

Collaboration Diagram:

Here time is not represented implicitly. Instead messages are numbered explicitly and the visual emphasis is on showing which objects communicate with which others. Messages are numbered to show the order in which they should happen. Causal nesting can be shown as a decimal numbering scheme. Partial orders, representing concurrency, can be shown by using names instead of numbers at the appropriate level.

State Chart Diagrams

UML defines state diagrams, which allow a class to be defined in terms of the states it can be and the events which cause it to move between states. They are essentially Harel Statecharts. These derive from conventional state transition diagrams, with the additional features that:

a state may indicate that the object is engaged in some activity, transitions between states can be due to messages

[1]Corresponding Author

or to changes in certain conditions or to a combination of these states can be nested within super-states.

Activity Diagram

These are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control.

Petrinets

Petri nets are a formalism which has been developed to describe the concurrent behaviour of interacting systems. A system is represented graphically using **places**, **transitions**, **arcs**, and **tokens**. Places, graphically represented by circles, stand for the sub-states that parts of the system can be in. A token, indicated by a spot, is present whenever that part of the system is in the corresponding sub-state. The place is said to be marked. Transitions are represented by solid bars. Transitions have input arcs which link places to transitions, and output arcs which link transitions to places. There are no arcs directly from one place to another or from one transition to another. A transition is enabled if all of the places attached to its input arcs have a token present. A transition which is enabled *may be* there, causing one token to be removed from each of its input places, and one token to be can't be there. Mutual exclusion can be demonstrated by proving that two places in different components are never simultaneously occupied by a token. Traditional Petri nets do not measure time in any way. A number of extensions have added timing information, sometimes to allow more powerful theorems to be proved about the set of states that can be reached.

III. Comparision

In this section we survey existing ideas for exploiting UML designs for performance modeling

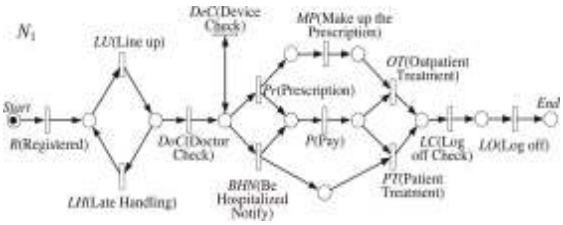
NAME OF THE DIAGRAM	ADVANTAGES	DISADVANTAGES	APPLICABLE
Sequence Diagram	• Expressive and less compact • It shows incomplete or inconsistent	• forced to extend to the right when adding new objects; consumes horizontal	• Business process modeling • Requirement specification • Communication
	representation	space •	modelling
Collaboration diagram	• Strips outline basic conceptual ideas	• Here timing is not shown as sequence diagram	• Model flow of control • Shows object flow of control
Activity Diagram	• It seems to very similar to petrinets • They model open ,reactive systems	• Though it is similar like Petri Net it does not have formal semantics	• Modeling workflows • Modeling business requirements
State chart diagram	• Squentiality and concurrency both are similar like petrinets	• Not every behavior is modelled	• Modeling dynamic aspects of thesystem • Tomodel the lifetime of a rective system
Petri nets	• Concurrent execution • Distributed systems • Non determinism • Simplicity • Visual Representation	• It is not easy for developers to draw petrinet,especially for complex systems • Lack of abstract modeling,that is designer is always forced to draw with the smallest details (place/transition)	• Modelling logistic systems • Databases • Designing real time digital controllers of embedded systems.

Example

Conversion from UML models can be achieved with simple transformation approaches.

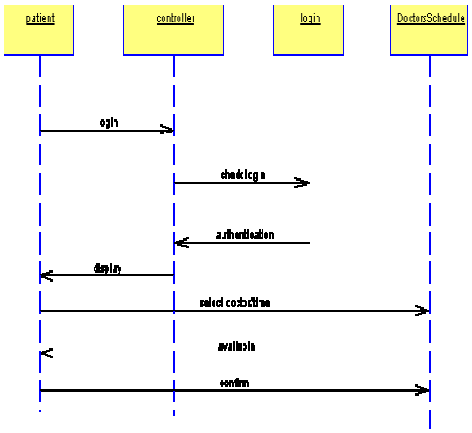
An example explained how the hospital registration is performed in different modeling techniques .

Petri net:

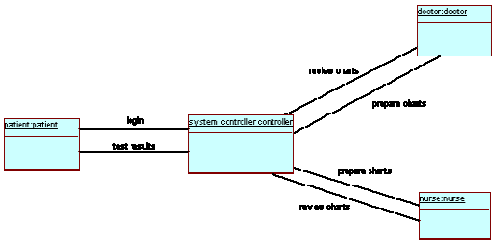


The process model of the Patient Registration System

Sequence Diagram

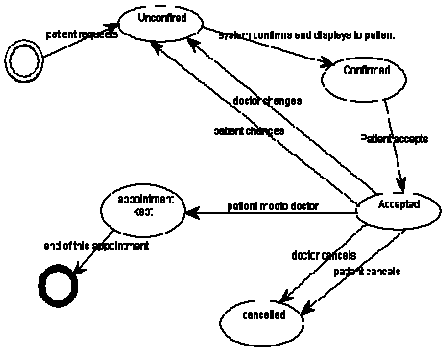


Collaboration Diagram



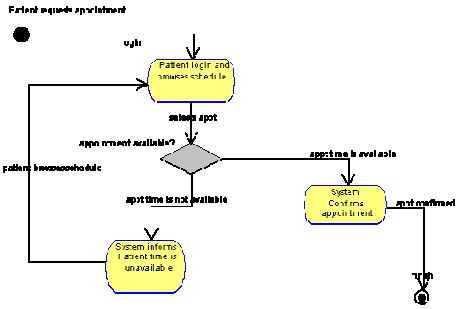
This collaboration diagram represents the flow of information within the "maintain charts" use case. The doctors and nurses can prepare inputs to the medical records and charts. Lab test results can be viewed by the patient.

State Chart Diagram



The appointment class goes to various transitions in state as the patient and/or doctor selects changes or cancels the appointment. When a patient requests an appointment, and unconfirmed appointment record is created. The record becomes confirmed if it is available, and it is presented to the patient. If the patient accepts the appointment, the record enters an accepted state. At any time the doctor or patient can cancel the appointment at which time the record enters and cancelled state. The state transition reached an end point whenever the appointment is kept.

Activity--Schedule Appointment [Activity]



This Activity Diagram represents operation viewpoint and flow of information between activities for the "make appointment" use case. A decision diamond provides the branching based on whether the time slot is available. The loop back show how alternative appointment times are selected.

V. Conclusion

This paper has presented some common aspects of different UML modeling techniques. All the advantages and disadvantages different methodsare tabulated.Finally an example for patient registration drawn with Petrinet ,Sequence Diagram,Collaboration Diagram,State chart and activity diagram.

The future work can be carried out as converting all UML models to different variants Petrinet Models and can be analysed.

References

[1] Xinhong Hei, Lining Chang, Weigang Ma, Jinli Gao, Guo Xie."Automatic Transformation from UML Statechart to Petri Nets for Safety Analysis and Verification"

[2] Chuanyi Li a , b , Jidong Ge a , b , ∗, Liguo Huang c , Haiyang Hu b , d , Budan Wu b , Hao Hu a , Bin Luo a" Software cybernetics in BPM: Modeling software behavior as feedback for evolution by a novel discovery method based on augmented event logs "

[3] Tony Spiteri Staines ,"Transforming UML Sequence Diagrams into Petri Nets"

[4] Peter King and Rob Pooley "Derivation of Petri Net Performance Models from UML Specifications of Communications Software"

[5] Simona Bernardi, Susanna Donatelli "From UML Sequence Diagrams and Statecharts toanalysable Petri Net model