UML Models for Dependability Analysis of Real-Time Systems

Nawal Addouche, Christian Antoine, Jacky Montmain

UML Models for Dependability Analysis of Real-Time Systems*

N. Addouche, C. Antoine and J. Montmain
URC- CEA/EMA
Laboratoire de Génie Informatique et d'Ingénierie de Production
Site EERIE de L'Ecole des Mines d'Alès
Parc Scientifique Georges Besse – 30035 Nîmes Cedex 1 – France
nawal.addouche@ema.fr, christian.antoine@ema.fr, jacky.montmain@ema.fr

Abstract - *In this paper, we present the UML profile called DAMRTS (Dependability Analysis Models for Real-Time Systems) representing an extension to the reference metamodels of the OMG profile for "Schedulability Performance and Time" (SPT). The aim is to provide concepts that enable to specify a real-time system with stochastic and probabilistic information allowing dependability analysis. A behavioural UML models are also proposed with a formal semantics designed for probabilistic model checking. An extension of statecharts semantics is developed with probabilities and real-time requirements, resulting in probabilistic timed automata (PTAs) as semantics models. These models are used to verify probabilistic temporal properties related to the dependability of real-time systems.*

Keywords: Real-time systems, UML, dependability analysis, probabilistic model Checking.

1 Introduction

The use of UML for developing real-time systems is widely adopted in industry. Other steps are also important in the development of real-time systems. Performance and dependability evaluations are based on separate models: queuing files, stochastic Petri nets [4], stochastic process algebras [2] and Markov processes are generally used. The formal verification is done on system models developed in other formalisms. In particular, for verifying temporal properties, timed automata [1] or timed Petri nets are used. In the case of the probabilistic temporal properties, formalisms like probabilistic timed automata or continuous time Markov chains are used [6]. However, these mathematical models are too fine grained to be directly specified by a real-time system designer. In order to have UML accepted by the real-time development community, the OMG group has proposed a profile called "Schedulability, Performance and Time" [10] for real-time systems. In this profile, some supports are introduced in UML to capture a maximum of real-time requirements and to perform the real-time development tasks directly on UML models. Beside the usual analysis and design stages, scheduling analysis, performance evaluation and formal verification of critical properties are included. However, the two last activities are partially covered because "quality of service" requirements are introduced without a clear indication about the formal verification of this type of properties. Adapted tools to formal verification or performance evaluation on these UML models are not yet available. The profile is too general as it covers all real-time problems both soft and hard. For all these reasons, a new profile is proposed to analyse and verify dependability properties of real-time systems. Our proposal has the following aims:

- To be compliant with the standard OMG's SPT profile,
- To give a UML quantitative models of real-time system : the model must cover functioning and malfunctioning with probabilistic aspects,
- To propose a formalisation of behavioural UML models with probabilistic timed automata in order to verify dependability properties. A probabilistic model checker requires specification of these properties with a suitable temporal logic.

2 Modelling process

Before defining proposed profile, it is important to follow a modelling process which allows collect of qualitative and quantitative information. This one is needed to dependability analysis. Modelling steps are defined as follow:

- Resources identification: among different real-time systems, we focused on automated systems of production. We must identify the set of sensors and effectors (all components in contact with raw material such robots, conveyor belt, etc.) which compose the controlled system. We also consider the set of PLC (Programmable Logic Controller) composing the controller system. These components are modelled as system resources. Real-time data are also identified (e.g. response time, deadlines, duration, etc.),

- Malfunctions analysis: quantitative information related to resource dependability is collected (e.g. reliability, maintainability, etc.). It is generally represented by rates related to a probabilistic distribution [11]. Particular qualitative information is also taken into account. It concerns a set of undesirable events being able to occur on system components. For each one of them the causes are defined with one or several logical combinations of components faults. This type of information is available in a dependability analysis based on faults trees (a model of faults combinations leading to undesired events) [11]. We introduce it in the dynamic UML models,

- Static modelling: the static aspect of system is modelled with a class diagram. Classes represent resources system. Real-time and dependability features collected in first steps are associated to *resource* classes,

- Dynamic modelling: it is based on collaboration diagram and extended UML statecharts defined in section 3.2.2. Malfunctions are introduced in these models using dependability information collected in dependability analysis step,

- Properties identification: After modelling real-time system, dependability analysis consists to verify some properties on the model. Those are defined in the profile SPT as required quality of services (QoS). In our profile they are not represented in UML model. After translating UML models to a formal model, properties are specified in a suitable temporal logic and verified with probabilistic model checking [6].

3 The DAMRTS profile

A profile is composed by a set of UML diagrams selected in the UML metamodel. The extension mechanisms (stereotypes, tagged values, constraints) make it possible to add a new elements specific to the application domain [9]. The UML profile SPT specialises the UML reference metamodel in a specific metamodel dedicated to domain of real-time systems analysis. The DAMRTS profile is a specific profile designed for dependability analysis of a real-time system (see figure1). It is based on concepts defined in the profile SPT with new stereotypes. Those are added to the metamodel in order to introduce particular dependability information. The malfunctions considered as undesirable events and their possible causes are modelled with stereotypes.

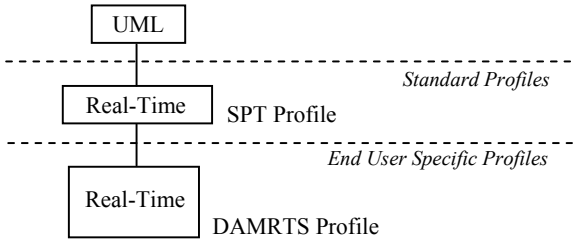

Figure1. Standard and specific profiles

3.1 Static view

The static aspect of the system is described by a class diagram. It is based on general resources model proposed in the profile SPT. The resources model provides quantitative information to UML specifications. It mainly represents a client/server model with quality of service offered by these resources and required by the client. The quality of service can represents performances (response time, deadline, etc) and/or dependability (reliability, availability, etc) features relating to the real-time systems. In DAMRTS profile, static aspect of the system is given by a class diagram representing both controller and controlled systems. Each component is modelled with a *resource* class.

3.1.1 Resource type

A part of general resources model presented in [10] deals with different categorisations of resources. They are based either on purpose or on activeness or on protection. A given resource instance may belong to more than one type, although it can be classified by at most one type from each category (See categorisations of resources in section 4.1.6 of [10]). In DAMRTS profile, two types of resource are taken into account. Based on purpose, resource related to a controlled system component (sensors and effectors) is classified as device. The second one representing a PLC is classified as processor resource.

3.1.2 Quality of service

In general resource modelling of the profile SPT, a generic notion of a “QoS characteristic” is presented, but there is no particular correspondent stereotype. QoS characteristic can be presented as different ways (either as tagged values or as stereotypes) [10]. In proposed profile, the QoS is represented as attributes when it is about actions of resource classes (e.g. duration of actions, response time

for a call action, etc.). It is also represented as a tagged value when it is about general QoS, like reliability and maintainability of resources (see figure 3). The modelled QoS are those offered by resources and not required by the client. They can take as value a nonnegative real related to dense time [5] or rates [11]. The rates represent constants related to probabilistic distributions. In our case, there are associated to random events (e.g. undesired or lost signals, failures of effectors, etc.).

3.1.3 Indicator and Cause stereotypes

Indicator Class. Each *resource* class is associated to a *Indicator* class. One or several variables (booleans or integers) are defined as attributes of *Indicator* class. These variables are related to dynamic aspect of the *resource* class objects and represent their degraded or failure states. When evaluation of variable became true, it means that the object is in a malfunction state.

Cause Class. Attributes of *Cause class* is defined in terms of the corresponding *Indicator* class attributes. They represent one or several logical expressions composed by an elementary logical conditions linked by conjunctive and disjunctive connectors. Variables used in the elementary conditions are those defined in Indicator classes. Actions of Cause stereotype consist to evaluate the logical expressions. When one of them is true, it indicates that associated failure became true.

3.2 Dynamic view

To represent dynamic aspects of the system, extended UML statecharts and collaboration diagrams are used. Combination of these two diagrams allows representing all system interactions. Indeed, the collaboration model describes external interactions between objects whereas UML statechart diagrams represent how an instance of a class reacts to an event occurrence. The proposed statecharts allow expressing events with probabilities and the real-time constraints of actions.

3.2.1 UML Collaboration diagram

A collaboration diagram consists of objects and associations that describe how the objects communicate. It represents the structural organisation of objects which exchanges messages. In the DAMRTS profile two types of messages: signals and orders. The first ones are sent from objects of Sensor classes to objects of Controller class. The second ones are sent from instances of Controller class to instances of Effector classes (see figure 4).

3.2.2 Extended UML Statecharts

In UML, each class has an optional statechart which describes the behaviour of its instances (the objects). This statechart receives events from other statecharts and reacts to them. The reactions include sending of the new events to other objects and executing of internal methods on the object. The communications between components of the system are modelled as events. Exchanged signals and orders as well as random events (e.g. undesired and lost signals) are represented as events associated to a discrete probability distribution. The syntax of UML statecharts, defined in the standard UML [8] is extended. The operational semantics of UML statechart, given in [3] is also extended with real-time and probabilistic aspects. A new type of dependability information excerpted from faults trees analysis is also introduced.

Syntax. This section describes the informal interpretation of extended UML statecharts. Graphical representation of UML statecharts is based on a set of nodes and a set of edges. An edge is presented by the following syntax*:*
Edge: = Event [Guard] / Action.
Event: = Event name (Probability).
Guard: = Boolean Expression.
Action: = Operation name (Arguments) [Duration, deadline].
Event represents either received signals or random events with their associated probability,
Guard is a boolean expression of predicates defined on the set of attributes of the corresponding *Cause* classes. These attributes represent AND-composition and OR-composition states of different objects. The compositions are excerpted from a faults tree analysis of the system.
Action expresses operation execution or sending messages to other objects. They are not instantaneous but have duration or deadline. Transitions between states are probabilistic. When two transitions are enabled, the choice is nondeterministic.

Semantics. Before presenting the extended semantics, some notations are fixed: the power set of a set E is denoted by $\mathbf{P}(E)$. A probability space is denoted (Ω, E, P), where Ω is the set of possible outcomes of the probabilistic experiments, $E \subseteq \mathbf{P}(\Omega)$ is a σ-algebra of measurable sets and *P:* $E \to [0,1]$ is a probability measure. A discrete probability space is denoted $(\Omega, \mathbf{P}(\Omega), P)$. A system consists of a finite collection of communicating statecharts. In the following, we assume a given finite collection of extended statecharts, denoted by $\{ESC_1, \ldots, ESC_n\}$.

A single extended UML statechart ESC_i consists of the following elements:

- A finite set $Nodes_i$ of nodes with a tree structure, described by the function $children_i : Nodes_i \to \mathbf{P}(Nodes_i)$.

- A finite set $Events_i$ of events. A discrete probability distribution is associated to events,

- A set $Guards_i$ of guard expressions. Guard expressions are boolean combinations. We use the symbol $\perp_g$ to denote "no guard required", $\perp_g \notin Guards_i$.

- A set $Actions_i$ of actions. The symbol $\perp_a$ is used to denote "no action required", $\perp_a \notin Actions_i$. A set of clocks is defined on actions. We consider that actions are executed in their correspondent nodes. So, these clocks dictates when UML statechart may remain in a node, letting time pass,

In semantics proposed in [3], probability is defined on actions. Events are considered as instantaneous. In our case, we introduce some changes which consist to associate probabilities to events and not for actions.

Let a finite set E-Edges$_i$ of Extended edges. A E-edge is a tuple (X, e, g, a, Y, P) where $X \subseteq Nodes_i$ is a non-empty set of source state nodes, $g \in Guards_i \cup \{\perp_g\}$, $a \in Actions_i \cup \{\perp_a\}$ and P is a probability measure in the discrete probability space:

$$(\mathbf{P}(Events) \times \mathbf{P}(Nodes_i) \setminus \{\phi\}, P).$$

$E \subseteq Events_i$ is a set of events and $Y \subseteq Nodes_i$ is a set of target nodes. The set E-Edges$_i$ is defined as: $\{(X, e, g, a, Y) \mid \exists X, g, a, P : \nu(X, g, a, P) = (X, g, a, P) \in$ E-Edges$_i \wedge P(\{(E, Y)\}) > 0\}$

Let a finite set of clocks K . C is a function assigning to each element $a_i^j \in Actions_i$ a clock $k_i^j \in K$, $j \in N$; $C : Actions_i \to K$.

V is a function assigning to k_i^j , a values in the dense time domain; $V : k_i^j \to \mathbf{R}^{\geq 0}$.

We define a function: $Actions_i \to D$ which assigns a value $d_i^j \in \mathbf{R}^+$ to each action $a_i^j \in A_i$.

Given an E-edge (x_i, e, g, a, P, y_i) where $x_i \in X$ and $y_i \in Y$. We associate the clock k_i^j to the action a_i^j , when the E-edge is enable; the node y_i becomes active and remains as long as $d_i \geq K_i^j \rangle 0$.

4 Example: Assembly chain of micro-motors

This example presents an automated chain assembly of electrical micro-motors. It is excerpted from a European project named PABADIS (Plant Automation BAsed on DIstributed Systems). This one deals with a flexible and a reconfigurable system designed for production of different types of micro-motors. Figure 2 represents the controlled system.

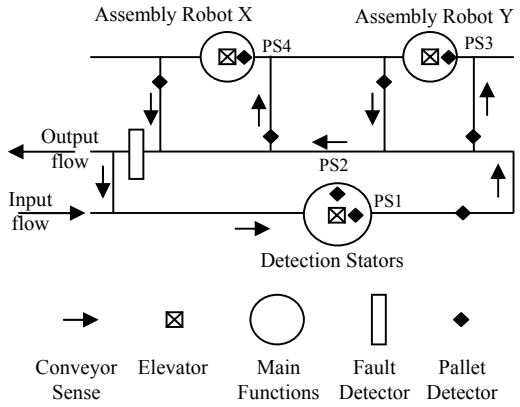

Figure 2. Assembly chain

Micro-motors consist to stators and rotors. The first are transported to assembly robots, on pallets via a conveyor system and seconds are available into stocks near each robot. A set of pallets containing stators moves along the conveyor belt. These are detected by pallet sensors PSi at different levels of the conveyor system. When assembly of micro-motors is completed, the pallets then move into a fault detection station where a camera detects the possible assembly faults. Set of PLC (Programmable Logic Controller) and PC composes the control system.

Let us consider an undesirable event such an *Assembly fault*. Among the causes of this mode, there are undesired signals sent by the sensors, undesired orders sent by the controller and material failures of elevator. To simplify model and make it easy to understand, some components (conveyor belt, stators detector, etc) are implicitly omitted in the following models.

4.1 Static UML model

Figure 3 describes the static aspect of the assembly chain example. A set of classes which represent the controlled system and the system controller are linked with associations. Stochastic and real-time attributes are given. Such defined in the profile DAMRTS, the proposed QoS is given either as tagged values or as operations attributes. For example, *Robot* Class represents the assembly robots. The operations *Ass* () and *Stop* () are called by *Controller* class. Their proposed execution times are associated as attributes. C*ontroller* Class reacts to signals coming from *Sensor* class or *Default detector* class by sending orders (O.Go up, O.Go down, O.Ass, O.Stop) to *Elevator* and *Robot* classes. The deadline to send these Orders represents the operations attributes. *Robot* class is associated to the stereotypes *Indicator* and *Cause*. They represent respectively the failure state "Faulty Assembly" and its cause.

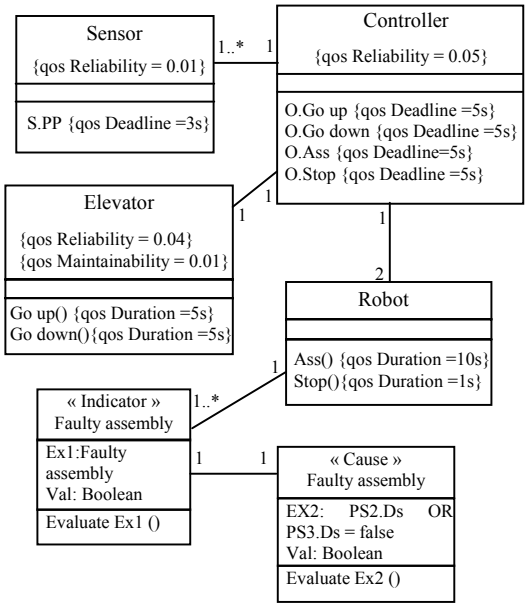

Figure 3. Class diagram

4.2 Collaboration diagram

Interactions between objects of classes are presented in figure 4. Exchanged messages describe signals sent from *Sensors* (S.PPi for *sensor* i) and *Fault detector* objects (S.Fault) to *controller*. They also represent orders sent from *Controller* to *Robot* and *Elevator* objects. Our example presents a distributed system such that several controllers (PLC) interact to control the system functioning. To simplify, we represent in the collaboration diagram one *controller* object.

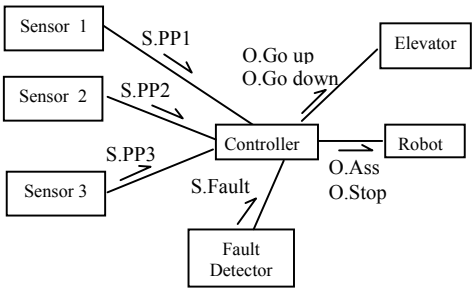

Figure 4. Collaboration diagram

4.3 Statecharts diagram

Robot and *Controller* objects behaviour are respectively modelled as given in figure 5 and figure 6. In *Robot* statechart, Orders are modelled as events. In the example: "O.Ass (0,10) [PS1.Ds OR PS2.Ds]/ Assembly()[10s]", guard expresses that the edge is enabled if one of sensors PS2 or PS3 is in degraded state *Ds*. The probability of sent an assembly order when one of sensors are in degraded state is evaluate to 0,10. The execution of the operation of assembly *Ass* () lasts 10s. The guard "S.Active AND E.Active" represents the condition to leave the state *Emergency stop*: sensors and elevator must be in the state *Active* of their respective UML statecharts. When probability is not represented, it means it is equal to 1.

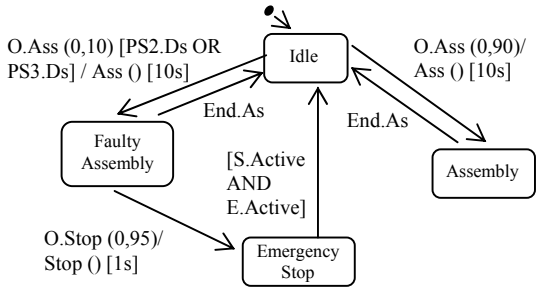

Figure 5. Robot statechart

Among the malfunctions of controller, sending of undesired orders or lost orders are modelled in *Controller* statechart as random events, e.g. "Fault (0.05)/ Order [5s]". The received sensor signals are presented as events and the sending of orders as actions with their associated deadlines. The edge "After 3s[SPP2=True]/ O.Go down().RT=5s", expresses that when PS2 detects a pallet (S.PP2=true), order to go down from controller to elevator must be sent after 3 seconds.

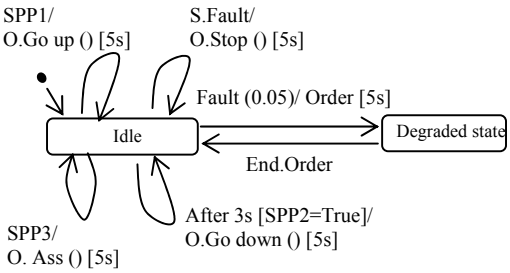

Figure 6. Controller statechart

5 Translating UML models to PTAs

Timed automata are automata extended with clocks, positive real valued variables which increase uniformly with time, and whose nodes and edges are labelled with clocks constraints, respectively called *invariants* and *guards*. The invariant dictate when the automaton may remain in a node, letting time pass, and guards when the corresponding edge can be taken [1]. Probabilistic timed automata are a variant of timed automata extended with discrete probability distributions [6]. This type of automata has been chosen for formalising extended UML statecharts because it takes into account dense time, nondeterminism and probabilistic choice as defined in the extended UML statecharts. They are also amenable to model check probabilistic temporal properties. In figure 7, we give probabilistic timed automaton describing a sub-system of the assembly chain example: the robot behaviour reacting to controller orders. The probabilities used in the example should in practice be obtained from statistical analysis of observed behaviour.

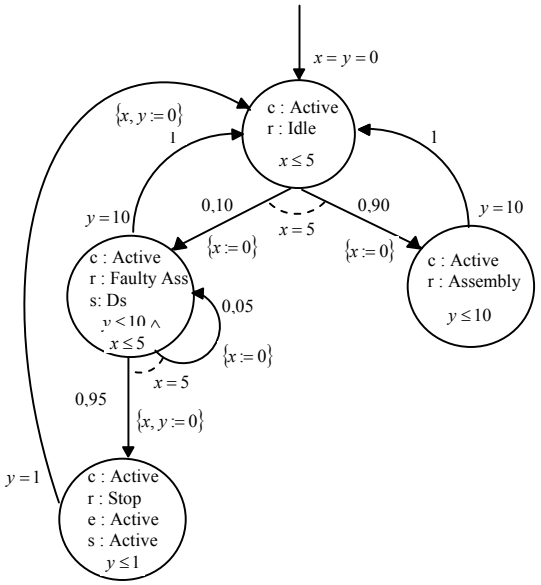

Figure7. Probabilistic timed automaton

In the probabilistic timed automaton, the sub-system consists to robot, controller and two clocks *x* and *y*. Atomic propositions, related to probabilistic timed automata of elevator and sensor, are included in nodes. "s:Ds, e:Active and s:Active" express guards of UML statecharts in figure 5. In initial state, both clocks x and y set to 0. The controller send assembly order to robot in 5 time units. Then, the robot effect the assembly tasks with probability 0,90. After assembly takes 10 time units, the robot becomes idle. When one sensor is in degraded state, then robot perform a fault assembly with probability 0,10. When an order stop is send by the controller, the robot stop. It becomes idle when sensors and elevator are in active state.

6 Conclusion

The paper presents the profile DAMRTS designed for dependability analysis of real-time systems. The static model is based on the metamodels of the OMG group profile called "Schedulability, Performance and Time". Some concepts such as *Indicator* and *Cause* stereotypes are defined in the metamodel in order to introduce new dependability information excerpted from dependability analysis based on faults trees. The dynamic models are based on collaboration and extended UML statecharts such proposed in our modelling process.

The UML statecharts are extended by introduction of probabilities and real-time aspects using clocks, an example of assembly chain with real-time features is well given. We proposed an extended semantics of UML statecharts related to probabilistic timed automata. The UML statecharts are then easily convertible to probabilistic timed automata. The aim is to verify formally probabilistic temporal properties related to the dependability of real-time systems.

In future works, the formal model will be integrated in the probabilistic model checker PRISM [7]. The probabilistic properties will be specified in PTCTL (Probabilistic Timed Computation Tree Logic). This one is suitable when the formal model is described with probabilistic timed automata.

References

[1] R. Alur and D. L. Dill, "A theory of timed automata", Theoretical Computer Science, 126(2):183-235, 1994.

[2] C. Canevet and Al, "Performance modelling with UML and stochastic process algebra", In Proceedings of the Eighteenth Annual UK Performance Engineering Workshop, July 2002.

[3] D.N. Jansen, H. Hermanns and J-P Kaoten, "A Probabilistic Extension of UML Statecharts: Specification and Verification", FTRTFT 02, Oldenburg, Germany, 2002, pages 355-374.

[4] P.King, R.Pooley, "Using UML to derive stochastic Petri nets models", In UKPEW'99. Proceedings of the 15th UK Performance Engineering Workshop, the University of Bristol, July 1999.

[5], H.Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Kluwer Academic Publishers, 1999.

[6] M.kwiatkowska, "Model Checking for Probability and Time : From Theory to Practice", In LICS 03, IEEE Computer Society Press, June 2003, pages 351-360.

[7] M.kwiatkowska, G.Norman and D.Parker, "Prism:Probabilistic Model Checker", In Proc.TOOLS 2002, volume 2324 of LNCS, 2002, pages 200-204.

[8] OMG. "Unified Modeling Language Specification" v.1.5, OMG Document Formal / 03-03-01, March 2003.

[9] OMG. "White Paper on the Profile mechanism v.1.0", Analysis and Design Platform Task Force, OMG Document ad/99-04-07, April 1999.

[10] B.Selic, A.Moore. "Response to the OMG RFP for Schedulability, Performance and Time": Revised submission ", OMG document ad/2001-06-14.

[11] A. Villemeur, *Sûreté de fonctionnement des systèmes industriels*, Ed Eyrolles, 1988.