A Framework Dedicated to Time Properties Verification for UML-MARTE Specifications *

Ning Ge, Marc Pantel and Xavier Crégut
University of Toulouse, IRIT/INPT, France
{ning.ge, marc.pantel, xavier.cregut}@enseeiht.fr

Abstract

In this paper, we present a framework dedicated to time properties verification for UML-MARTE specifications. This framework relies on a property-driven transformation from UML architecture and behaviour models to executable and verifiable Time Petri Nets (TPN) models. Meanwhile, it translates the time properties into a set of property patterns, corresponding to TPN observers. The observer-based model checking approach is then performed on the produced TPN. This verification framework can assess time properties like upper bound for loops and buffers, Best/Worst-Case Response Time (B/WCRT), Best/Worst-Case Execution Time (B/WCET), Best/Worst-Case Traversal Time (B/WCTT), schedulability, and event/task-level time constraints.

1 Introduction

Model-driven Engineering allows an early integration of feasibility analysis in the design phase, and enables a rapid iterative design-prototype cycle. The core issue is how to rapidly validate and verify that the system's behaviour matches the specifications, especially for the time aspect. To our knowledge, no formal specification of the whole UML-MARTE language have been currently provided. And, even if this was the case, the analysis of such an expressive language cannot scale with the current formal verification technologies like model checking if it relies on the same transformation for the various kind of properties.

In this paper, we introduce a framework dedicated to time properties verification of UML-MARTE specifications, which relies on property-driven transformation to Time Petri Nets (TPN). Our work follows the same approach as [4] to design a time property verification toolset for UML-MARTE specifications. In this purpose, we propose the following 2 approaches: *Transformation from UML-MARTE models to executable TPN models; Translation and verification of time properties using observer based model checking approach.* This verification framework can assess time properties like upper bound for loops and buffers, Best/Worst-Case Response Time (B/WCRT), Best/Worst-Case Execution Time (B/WCET), Best/Worst-Case Traversal Time (B/WCTT), schedulability, and event/task-level time constraint in both logical and chronometric aspects (synchronization, coincidence, exclusion, precedence, sub-occurrence, causality).

This paper is structured as follows: Section 2 discusses the related works; Section 3 introduces a case study; Section 4 presents UML-MARTE verification framework and the associated approaches; conclusions and further work are discussed in Section 5.

2 Related Works

Some related works are also aimed to verify the properties of UML specifications. [1] and [14] respectively use Esterel and Promela as verification language. Although their time constraint

*This work was funded by the French ministries of Industry and Research and the Midi-Pyrénées regional authorities through the ITEA2 OPEES and FUI Projet P projects

specification language, CCSL, covers logical and chronometric constraints in UML, their verification approach only supports logical time constraints so far to our understanding. [5] translates UML diagrams into Maude language and verifies deadlock property using LTL. This work does not handle other time properties. In terms of performance, this work indicates that they still need to test on larger examples. [8] transforms UML to timed automatons, then translates them to concrete programs for SPIN model checker. It verifies the consistency between different system descriptions. This work does not concern the time aspect. [9] presents MARTE2MAST, a tool that analyses the schedulability using MAST. It supports analysis using simulation tools and static analysis techniques. As the simulation is not exhaustive, it cannot prove the correctness in all possible cases. [13] describes a search-based UML-MARTE analysis method for starvation and deadlock detection. It uses genetic algorithms to search through the state space. As the genetic search method cannot ensure the building of the full state space including all the final states, this method cannot guarantee that the whole space is exhaustively searched. It can detect errors but cannot prove their absence.

3 Case Study

A classical asynchronous Real-Time System (RTS) model is described in Fig. 1. According to the general asynchronous message-driven pattern, the *Sender* will regularly distribute data to the two receivers *Calculator A* and *Calculator B* through the *Router*. The receivers provide redundant control service. They will do some computation after receiving the data. The redundant controller requires that the output of the two calculators must be available at the same time in each working cycle; otherwise, the servo of the corresponding actuator cannot correctly unify the redundant command. In this case, the designer needs to verify *the coincidence of computation tasks between calculators A and B*. As it is impossible to respect a strict simultaneous timing with an explicit local synchronisation, a time tolerance is defined. Once the two time instants fall into the same time window (size of window equals to tolerance), they are considered as coincident. We use this case to illustrate our approaches in the following parts.

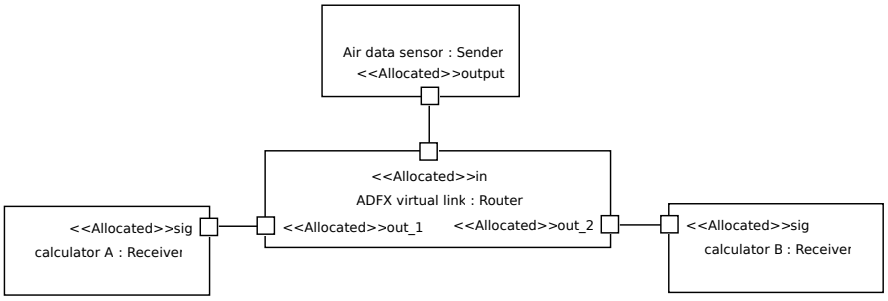

Figure 1: Case Study System Model

4 UML-MARTE Verification Framework

4.1 Overview

The framework is shown in Fig. 2. It verifies whether the UML-MARTE specifications satisfy the expected Time Properties. *System Models* consists of both *Behaviour Model* and

Architectural Model. They are translated into *TPN* models through *Behaviour/Structure Transformation*. *Time Properties* are translated into *Time Property Patterns* by *Time Property Transformation*. The model checking is performed on the generated *Tag Pattern TPN* models and the corresponding *LTL/CTL/Marking Formulas* by using *TPN Model Checker*. The verification is based on the observers added in the TPN, which are used to observe the value of one *Property Pattern*. Finally, *Verification Result Computation* is performed to combine the *Property Pattern Results*, then the target *Time Property Verification Result* is available. We present the transformation and verification approaches in the following parts of this section.

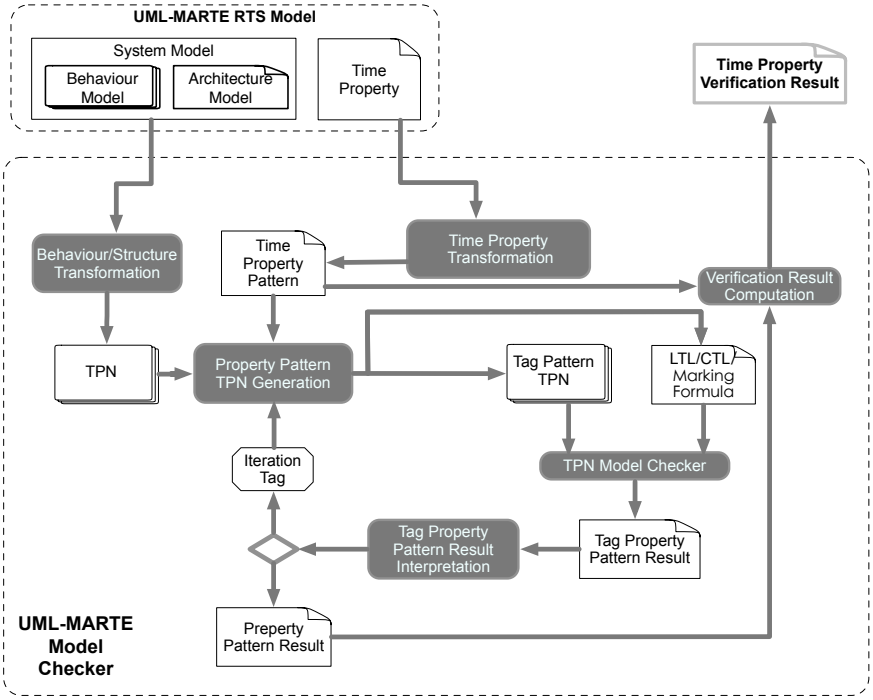

Figure 2: UML-MARTE Model Checker

4.2 Transformation from UML-MARTE to TPN

UML [11] and its domain specific extensions are becoming widely used notations to cope with the design of complex automotive real-time embedded applications. However, the semi-formal nature of its semantics makes UML limited for verification. In particular, it has a partial executable semantics that does not ease the use of model checking technologies. MARTE [12] is a UML profile standardized by the OMG. This profile adds capabilities to UML for model-driven development of real-time embedded systems. MARTE provides foundations for model-based description of real time and embedded systems.

In order to give a precise semantics to UML-MARTE specifications, we use a model transformation approach to build completely formal and verifiable models. Petri Nets are powerful models for describing system's behaviour and for verifying the properties. TPN [10] allows expressing and verifying time properties under both logical and chronometric time models. It is supported by TINA toolbox [2]. Compared to Petri Nets, the transitions in TPN are extended

with a time constraint that controls the firing time. For example in Fig. 3, transition T_1 is attached with time constraint [19,27]. When the token arrives at place P_1, the local timer of T_1 starts. Between 19 and 27 time units, T_1 can be fired.

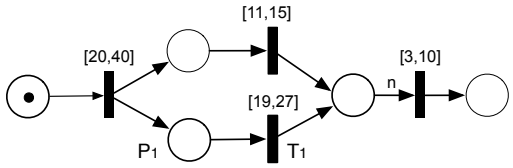

Figure 3: Time Petri Net Example

This transformation covers UML architecture models (using the full Composite Structure Diagram) and behaviour models (using the full State Machine Diagram and a major subset of Activity Diagram). It is property-driven in order to limit the state space during model checking. Property-driven means that the transformation of some UML elements can be different depending on the assessed property; meanwhile the transformation does not conserve all the information in UML, but only those concerning the property verification. Another issue is raised from TPN theoretical limits. As model checking and reachability are undecidable in TPN when using stopwatch [3], the transformation method should avoid using stopwatches. The transformation rules for UML Activity Diagram can be consulted in [7].

4.3 Translation of Time Properties into Time Property Patterns

Time properties are expressed using MARTE. These expressions cannot be directly verified by TPN model checkers. Thus, they should be translated into TPN-compatible analyzable formalism. The proposed method translates them into a set of verifiable quantitative property patterns. Each property pattern's value can be computed by iterative use of the model checker relying on dichotomy search. To make our method practical for end users, we focus on the time properties at both event and task levels. For the time constraints, we introduce the concept of *time tolerance*, because two simultaneous events cannot be measured without errors in the real world.

In our case study, the property is the coincidence between two tasks. The coincidence between two tasks is determinated by the coincidence between the Event_{start} and the Event_{end} of tasks. For the n^{th} occurrence of task X and Y, if the two Event_{start} are coincident and the two Event_{end} are coincident within time tolerant δ, task X and Y are coincident. Formally, task coincidence is translated into the following 3 equivalent assertions.

$Coincidence(X, Y, \delta) \equiv$

$$\forall t \in \mathbb{R}_+ : (|O(X_s^t) - O(Y_s^t)| < 2) \wedge (|O(X_e^t) - O(Y_e^t)| < 2) \quad (1)$$

$$\forall t \in \mathbb{R}_+ : (|T(X_s^t) - T(Y_s^t)| < \delta) \wedge (|T(X_e^t) - T(Y_e^t)| < \delta) \quad (2)$$

$$\forall i \in \mathbb{N}^* : (T(X_e^i) + \delta < T(Y_s^{i+1})) \wedge (T(Y_e^i) + \delta < T(X_s^{i+1})) \quad (3)$$

In the above assertions, X represents task; X_a the inner event a of task X, particularly X_s for start event, X_e for end event; X_a^i the i^{th} occurrence of inner event of task X; X_a^t the occurrence of X_a which is the nearest (forward or backward) to the time instant t; $T(X_a^i)$ the occurring time instant of X_a^i; $T(X_a^t)$ the occurring time instant of X_a^t; $O(X_a^t)$ the occurrence count for X_a at time t; and δ is the time tolerance for coincidence. There are 4 time property

patterns in the assertions (Table 1). The task-level property is then represented by a set of event-level quantitative property patterns.

Table 1: Time Property Patterns

Time Property Pattern	Definition
X_s^{i+k}	Representation of event X_s^{i+k}
$\|O(X_a^t) - O(Y_a^t)\| < \delta$	Occurrence difference between events X_a^t and Y_a^t
$\|T(X_a^t) - T(Y_a^t)\| < \delta$	Relative T_{max} between events X_a^t and Y_a^t
$T(X_e^i) + \delta < T(Y_s^{i+1})$	Relative T_{min} between events X_a^t and Y_b^t

4.4 Verification of Time Property Patterns

To assess the time property, a TPN observer is added into the original TPN, and then TINA is used to verify the observer-dedicated LTL/CTL/Marking formulas on the TPN. Our approach guarantees that each assertion's verification is independent in terms of reachability graph generation, so a parallel computation is possible.

We choose the property patterns $|T(a^t) - T(b^t)| < \delta$ to illustrate the approach. A TPN observer (Fig. 4) is added to the original TPN. The middle transition will always instantly

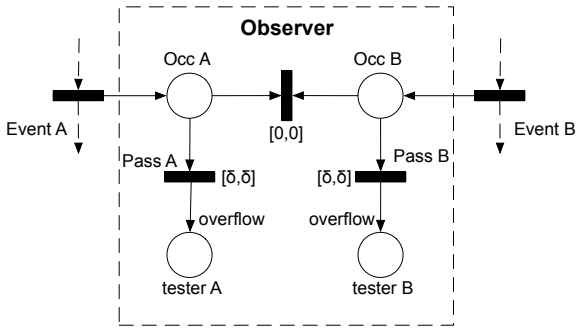

Figure 4: $|T(A^t) - T(b^t)| < \delta$ Pattern TPN Observer

neutralize the tokens from the places *Occ A* and *Occ B* except when one token waits for a time longer than δ that leads to the firing of one *Pass* transition. To guarantee the termination of model checking, the pattern is extended by adding a large overflow number on the tester's incoming arc. Places *tester A* and *tester B* are used to detect this exception. In the generated reachability graph, it only requires to verify if *tester A* or *tester B* has marking. The formula is: $\Diamond(testerA = 1) \vee \Diamond(testerB = 1)$.

Once it is known how to verify $|T(a^t) - T(b^t)| < \delta$, it is possible to change δ to compute a near optimal tolerance. If $|T(a^t) - T(b^t)| < \delta + 1$ is verified as true, but false for $|T(a^t) - T(b^t)| < \delta$, then the near optimal tolerance is $\delta + 1$. In order to improve the computation efficiency, a dichotomy search is used to reduce the complexity from $O(N)$ to $O(\log N)$ using divide and multiply by two instead of add or subtract one.

5 Conclusion and Future Works

This paper introduces a time property specific UML-MARTE specifications and verification framework relying on the translation to TPN. We illustrate the UML-MARTE transformation

method and time property verification method by a case study. The performance evaluation of the given case study can be consulted in [6].

In the future, we will focus on extending this framework. On the technical side, we will optimize the TPN models by finding some reducible structural patterns without influencing the property. On the methodological side, we will experiment with other kinds of properties, like the functional property, to improve the *Property-driven* approach to DSML (Domain Specific Modelling Language) model verification that started in the TOPCASED project.

References

[1] Charles André and Frédéric Mallet. Specification and verification of time requirements with ccsl and esterel. In *Proceedings of the 2009 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems*, LCTES '09, pages 167–176, New York, NY, USA, 2009.

[2] B. Berthomieu, P.-O. Ribet, and F. Vernadat. The tool tina - construction of abstract state spaces for petri nets and time petri nets. *International Journal of Production Research*, 42(14):2741–2756, 2004.

[3] Bernard Berthomieu, Didier Lime, Olivier Roux, and François Vernadat. Reachability problems and abstract state spaces for time petri nets with stopwatches. *Discrete Event Dynamic Systems*, 17:133–158, 2007.

[4] Benoit Combemale, Xavier Crégut, Pierre-Loic Garoche, Xavier Thirioux, and Francois Vernadat. A property-driven approach to formal verification of process models. In Jorge Cardoso, José Cordeiro, Joaquim Filipe, and Vitor Pedrosa, editors, *Enterprise Information System IX*, volume 12, pages 286–300. LNBIP, Springer, 2008.

[5] Patrice Gagnon, Farid Mokhati, and Mourad Badri. Applying model checking to concurrent uml models. *Journal of Object Technology*, 7(1):59–84, January 2008.

[6] Ning Ge and Marc Pantel. Time properties verification framework for uml-marte safety critical real-time systems. In *Proceedings of 8th European Conference on Modelling Foundations and Applications (ECMFA2012)*, July 2012.

[7] Ning Ge, Marc Pantel, and Xavier Crégut. Time properties dedicated transformation from uml-marte activity to time petri net. In *Proceedings of 5th International workshop UML and Formal Methods (UML&FM'2012)*, August 2012.

[8] Alexander Knapp and Jochen Wuttke. Model checking of uml 2.0 interactions. In *Proceedings of the 2006 international conference on Models in software engineering*, MoDELS'06, pages 42–51. Springer-Verlag, 2006.

[9] Julio L. Medina and Álvaro García Cuesta. From composable design models to schedulability analysis with uml and the uml profile for marte. *SIGBED Rev.*, 8(1):64–68, March 2011.

[10] P. Merlin and D. Farber. Recoverability of communication protocols–implications of a theoretical study. *Communications, IEEE Transactions on*, 24(9):1036 – 1043, sep 1976.

[11] Object Management Group, Inc. *OMG Unified Modeling Language^{TM}, Superstructure*, February 2009.

[12] Object Management Group, Inc. *UML profile for MARTE: modeling and analysis of real-time embedded systems version 1.0*, November 2009.

[13] M. Shousha, L. Briand, and Y. Labiche. A uml/marte model analysis method for uncovering scenarios leading to starvation and deadlocks in concurrent systems. *Software Engineering, IEEE Transactions on*, PP(99):1, 2010.

[14] Ling Yin, Frédéric Mallet, and Jing Liu. Verification of MARTE/CCSL Time Requirements in Promela/SPIN. In *IEEE ICECCS 2011 - 16th IEEE International Conference on Engineering of Complex Computer Systems*, Las Vegas, États-Unis, April 2011.