Verification of Synchronization-Related Properties for UML-MARTE RTES Models with a Set of Time Constraints Dedicated Formal Semantic

Ning Ge, Marc Pantel

Verification of Synchronization-Related Properties for UML-MARTE RTES Models with a Set of Time Constraints Dedicated Formal Semantic

Ning Ge and Marc Pantel
University of Toulouse, IRIT
2 rue Charles Camichel, BP 7122, 31071 Toulouse cedex 7, France
Email: firstname.lastname@enseeiht.fr

***Abstract*—Critical Real-Time Embedded Systems (RTES) have strong requirement with respect to system's reliability. In Model-Driven Engineering (MDE), verification at early phases of the system lifecycle is an important issue, especially for time constraints in UML-MARTE RTES model. In order to assess that the time requirements are met by the behavior models, the key challenging problem is to transform these time constraints from the UML-MARTE model to computable formal semantics that provide time properties verification. Moreover, to allow the application of this formal semantic to real industrial use cases, the performance of verification should scale well. In this paper, we present a set of time constraint dedicated semantics under the framework for UML-MARTE RTES model's time requirement assessment. We focus on how to specify a set of synchronization-related constraints between system's tasks relying on a formal semantics and to accomplish verification by an efficient observer-based model checking method using Time Petri Nets. We analyse the method's computational complexity and demonstrate the method's scalability by illustrating some performance results.**

***Keywords*-MDE; UML; MARTE; real-time embedded system; time requirements; formal semantic; verification; task synchronization**

I. Introduction

As the complexity of the real-time embedded systems increases rapidly, the requirement of system's reliability are getting more and more pregnant. Model-driven Engineering (MDE) allows an early integration of feasibility analysis during design phase, and enables a rapid iterative design-verification cycle. The core issue of MDE for RTES is how to rapidly validate and verify that the system's behavior matches the specification, especially for the temporal aspect.

UML (Unified Modeling Language) [1] and its profile MARTE (Modeling and Analysis of Real Time and Embedded systems) [2] are standardized modeling language that are getting widely accepted by industrial designers to cope with the development of complex RTES.

Our research focuses on the use of MDE and formal methods in order to improve the development of safety critical systems. This work focus on the verification of real time properties for RTES the UML-MARTE modeling languages. The challenging problem in the work presented in this paper is how to encode the expected temporal properties in the mapping from concrete-level model to abstract-level formal semantic, and how to implement a scalable verification toolset relying on the translated formal semantics.

In this paper, we introduce an approach relying on the framework of the UML-MARTE Model Checker that was presented in [3], to formalize synchronization-related constraints for UML-MARTE models as a set of computable formal semantics that preserves the time consistency, and propose an observer-based model checking method using Time Petri Net (TPN) [4]. TPN has been selected as verification support, not only because of the maturity of both its theory and related toolset TINA [5], but also as it provides a powerful capacity to express temporal semantic.

We aim to, in both finite and infinite time scope, verify major tasks' synchronization-related constraints at RTES system-level. The main contributions in this paper are:

- Give out a formal definition of time constraints dedicated semantic at system level for both finite and infinite time scope:
  - Two tasks are coincident
  - Synchronization of two tasks
  - Mutual exclusion of two tasks' execution
  - One task is another one's sub-occurrence
  - One task is preceding another
  - One task is causal with respect to another
- Define a computable time property expression at TPN level for both finite and infinite time scope, and build the mapping from system-level properties to these computable properties.
- Demonstrate the approach is applicable and scalable for industrial applications, by analysing the computational complexity and illustrating the performance results.

The paper is organized as follows: Section II presents a brief overview of UML-MARTE model checker. Section III describes the proposed methodology. Section IV presents the formal definition of synchronization-related constraints. Section V introduces decomposition method concerning computable time property set at TPN level. Section VI discusses the computational complexity and illustrates a performance report to demonstrate the method's scalability. Section VII compares our work with related works. Section VIII gives some concluding remarks.

II. Overview of UML-MARTE Model Checker

The architecture of the model checker for UML-MARTE Model is shown in Fig. 1.

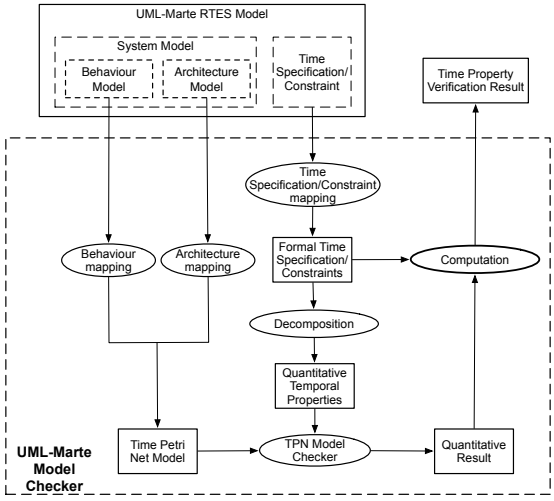

Figure 1. UML-MARTE Model Checker

The objective of the UML-MARTE model checker is to verify whether a <*UML-MARTE Model*> satisfies the designed <*Time Specifications*> or respects the required <*Time Constraints*>.

It takes the <*System Model*> and <*Time Specification/-Constraint*> as input. The <*System Model*> consists of two concerns: <*Behavior Model*> and <*Architectural Model*>. The former defines how the system will act and response to its outside world while the latter describes the interconnection relation between sub-components of the system. The system <*Specification/Constraint*> consists of two categories: functional and non-functional. For RTES, the former concerns whether system's output value is consistent with what it is designed for and whether the output is generated at desired time, while the latter focus on the performance requirements, sustainability, etc. The current OMG standard has not defined a complete specification framework for verification purpose, so in practice, we add our own profile to describe properties at task level. System and specification/constraint models are respectively transformed to associated semantic components at TPN level through <*Behavior/Structure Mapping*> and <*Time Specification/-Constraint Mapping*> approaches. All the transformations are performed automatically and the computable formal model is hidden to the end user.

Formal verification is performed with <*TPN Model*> and <*Generated Quantitative Timing Property*> generated by <*Decomposition*> method, in which a dedicated iterative observer-based model checking method and the TINA toolset are applied. Finally, <*Computation*> is performed with <*Quantitative Results*> and <*Formal Time Specification/Constraint*> to get the target <*Time Property Verification Result*>.

We have proposed an efficient approach for each core issue in the whole framework to accomplish the final verification purpose. In this paper, we focus on the approaches <*Time Specification/Constraint Mapping*> and <*Decomposition*>.

III. Methodology Description

The time property we want to verify at UML-MARTE level are the synchronization-related constraints between system's tasks. The exact meaning of these constraints may change in different context, so in order to have a generalized, complete and standard semantic, we investigate firstly some common time properties in industrial RTES domain and then synthesize them in a standard way to provide a formal definition. In order to follow the OMG specification framework, we choose the same basic semantic elements as CCSL's [6] but extended its semantic to cover chronometric time model.

Three temporal aspects are taken into account when defining the semantic of the time properties: it should be indifferent for mono-clock and multi-clock system, for discrete and dense time concept, and for finite and infinite time scope. According to our study so far, for some properties, if we want to use the same encoding for both finite and infinite time scope, some properties like exclusion, sub-occurence and precedent cannot be computed in finite time. A compromise is made in this paper: in order to make all time constraints dedicated semantic verifiable, for some of the properties, we use a different semantic in finite time scope and infinite time scope.

Another important factor is that, although the notion of synchronization should enforce things to occur simultaneously, in real industrial RTES, the strict simultaneous property is not be always expected to be perfectly achieved. The design requirement is usually associated with temporal tolerance. In order to take into account this more realistic fact, we introduce this tolerance for all time constraints dedicated semantic in this paper. The temporal tolerance is denoted by δ ($\delta \in \mathbb{R}^+$).

IV. Formal Definition of Time Constraints Dedicated Semantic

A. Preliminary definitions

Task A task is considered as the smallest computable unit in RTES, which will consume time and modify resources (consumes and produces). Its general function is to compute its outputs from its inputs. A task could be executed finitely or infinitely many times according to the system design. The terms *task*, *operation* and *action* in UML are equivalent.

Presence The presence of a task is the duration $[t_s, t_e]$ for task's execution where t_s and and t_e are its start and end time.

Occurrence Occurrence is an instant concept, so the occurrence of a task is not significant, but need to precise the associated inner event (start and end).

In order to simplify the presentation afterwards, we define the expressions that defines these concepts in Table I.

Table I
SYMBOL OF SYNCHRONIZATION-RELATED FORMAL DEFINITION

Symbol	Definition
X	Task X
X^i	The i^{th} presence of task X
X_a	The inner event[1] a of task X
X_a^i	The i^{th} occurrence of X_a
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
$O(X)$	The max possible presence of task X.
$O(X_a)$	The max possible occurrence for X_a.
$O(X_a^t)$	The occurrence count for X_a at time t

B. Coincidence

Task X and Y are coincident iff. the n^{th} occurrence of X occurs simultaneously with the n^{th} occurrence of Y while $n \in \mathbb{N}$. With a temporal tolerance introduced, it is possible that an interleave exists between i^{th} occurrence of X and j^{th} occurrence of Y when $i \neq j$, which violates the coincidence definition. So constraints for consequent occurrences must be added. The coincidence schema is shown in Figure 2.

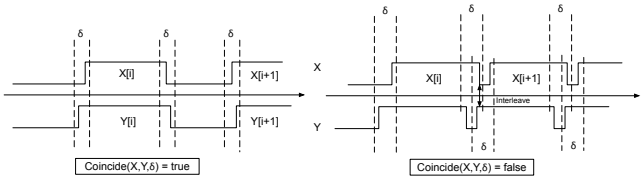

Figure 2. Coincidence

Definition 1 : Coincidence - Finite Time Scope

$C_{ft}(X, Y, \delta) \equiv$

$O(X_s) = O(Y_s)$
$O(X_e) = O(Y_e)$
$\forall i \in [1, O(X_s)] : |T(X_s^i) - T(Y_s^i)| < \delta$
$\forall i \in [1, O(X_e)] : |T(X_e^i) - T(Y_e^i)| < \delta$
$\forall i \in [1, O(X_e) - 1] : T(X_e^i) + \delta < T(Y_s^{i+1})$
$\forall i \in [1, O(Y_e) - 1] : T(Y_e^i) + \delta < T(X_s^{i+1})$

Definition 2 : Coincidence - Infinite Time Scope

$C_{ift}(X, Y, \delta) \equiv$

$\forall t \in \mathbb{R}_+ : |O(X_s^t) - O(Y_s^t)| < 2$
$\forall t \in \mathbb{R}_+ : |O(X_e^t) - O(Y_e^t)| < 2$
$\forall t \in \mathbb{R}_+ : |T(X_s^t) - T(Y_s^t)| < \delta$
$\forall t \in \mathbb{R}_+ : |T(X_e^t) - T(Y_e^t)| < \delta$
$\forall i \in \mathbb{N}^* : T(X_e^i) + \delta < T(Y_s^{i+1})$
$\forall i \in \mathbb{N}^* : T(Y_e^i) + \delta < T(X_s^{i+1})$

[1]The inner event in this paper could be the start of task (X_s) or the end of task (X_e).

C. Synchronization

Logical synchronization is a reduced coincidence relation without restricting a simultaneously execution. The only concern is that the execution order must persist.

Definition 3: Synchronization - Finite Time Scope

$Syn_{ft}(X, Y, \delta) \equiv$

$O(X_s) = O(Y_s)$
$O(X_e) = O(Y_e)$
$\forall i \in [1, O(X_e) - 1] : T(X_e^i) + \delta < T(Y_s^{i+1})$
$\forall i \in [1, O(Y_e) - 1] : T(X_e^i) + \delta < T(X_s^{i+1})$

Definition 4 : Synchronization - Infinite Time Scope

$Syn_{ift}(X, Y, \delta) \equiv$

$\forall t \in \mathbb{R}_+ : |O(X_s^t) - O(Y_s^t)| < 2$
$\forall t \in \mathbb{R}_+ : |O(X_e^t) - O(Y_e^t)| < 2$
$\forall i \in \mathbb{N}^* : T(X_e^i) + \delta < T(Y_s^{i+1})$
$\forall i \in \mathbb{N}^* : T(Y_e^i) + \delta < T(X_s^{i+1})$

D. Exclusion

Task X and Y are in exclusion relation iff. Not any presence of X occurs simultaneously with any presence of Y. This is not an intuitive definition for synchronous behavior, however it could be considered as another form of coincidence with some time offset. It needs equally to add the constraints for interleave situation. The schema of exclusion is shown in Figure 3.

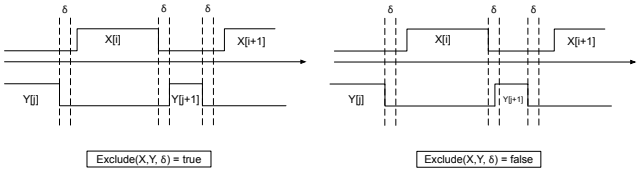

Figure 3. Exclusion

Definition 5 : Exclusion - Finite Time Scope

$E_{ft}(X, Y, \delta) \equiv$

$\forall i \in [1, O(X_s)], \forall j \in [1, O(Y_s)] :$
$T(X_s^i) + \delta < T(Y_s^j) \Rightarrow (T(X_e^i) + \delta < T(Y_s^j)) \wedge (T(Y_e^j) + \delta < T(X_s^{i+1}))$
$T(X_e^i) + \delta < T(Y_s^j) \Rightarrow T(Y_e^j) + \delta < T(X_s^{i+1})$
$T(X_s^i) + \delta < T(Y_e^j) \Rightarrow T(X_e^i) + \delta < T(Y_s^j)$
$T(X_e^i) + \delta < T(Y_e^j) \Rightarrow (T(X_e^i) + \delta < T(Y_s^j)) \wedge (T(Y_e^j) + \delta < T(X_s^{i+1}))$

The original semantics can not be applied in infinite time scope. A computable version reduces the semantics that the

execution of two tasks is overlapped and for any task there is no more than one continuous occurrence.

Definition 6 : Exclusion - Infinite Time Scope

$$
\begin{aligned}
&E_{ift}(X,Y,\delta) \equiv \\
&\quad \forall t \in \mathbb{R}_+ : |O(X_s^t) - O(Y_s^t)| < 2 \\
&\quad \forall t \in \mathbb{R}_+ : |O(X_e^t) - O(Y_e^t)| < 2 \\
&\quad \forall i \in \mathbb{N}^* : \\
&\quad T(X_s^i) + \delta < T(Y_s^i) \Rightarrow (T(X_e^i) + \delta < T(Y_s^i)) \wedge \\
&\qquad\qquad\qquad\qquad (T(Y_e^i) + \delta < T(X_s^{i+1})) \\
&\quad T(X_e^i) + \delta < T(Y_s^i) \Rightarrow T(Y_e^i) + \delta < T(X_s^{i+1}) \\
&\quad T(X_s^i) + \delta < T(Y_e^i) \Rightarrow T(X_e^i) + \delta < T(Y_s^i) \\
&\quad T(X_e^i) + \delta < T(Y_e^i) \Rightarrow (T(X_e^i) + \delta < T(Y_s^i)) \wedge \\
&\qquad\qquad\qquad\qquad (T(Y_e^i) + \delta < T(X_s^{i+1}))
\end{aligned}
$$

E. Sub-occurrence

Task Y is a Sub-occurrence of X iff.tThe i^{th} occurrence of X and the j^{th} occurrence of Y occur simultaneously, where always $j \leqslant i$. The schema of sub-occurrence is shown in Figure 4.

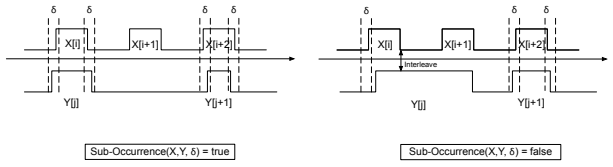

Figure 4. Suboccurrence

Definition 7: Sub-occurrence - Finite Time Scope

$$
\begin{aligned}
&S_{ft}(X,Y,\delta) \equiv \\
&\quad O(X_s) \geqslant O(Y_s) \\
&\quad O(X_e) \geqslant O(Y_e) \\
&\quad \forall j \in [1, O(Y_s)], \exists i \in [j, O(X_s)] : \\
&\quad (|T(X_s^i) - T(Y_s^j)| < \delta) \wedge (|T(X_e^i) - T(Y_e^j)| < \delta) \wedge \\
&\quad (T(X_e^{i-1}) + \delta < T(Y_s^j)) \wedge (T(Y_e^j) + \delta < T(X_s^{i+1}))
\end{aligned}
$$

The original semantic can not be applied in infinite time scope. A computable version reduces the semantic that the faster one's occurrence is always $k (k \in \mathbb{N}^*)$ times multiple of the slower one's.

Definition 8 : Sub-occurrence - Infinite Time Scope

$$
\begin{aligned}
&S_{ift}(X,Y,\delta,k) \equiv \\
&\quad \forall t \in \mathbb{R}_+ : |O(X_s^t)/k - O(Y_s^t)| < 2 \\
&\quad \forall t \in \mathbb{R}_+ : |O(X_e^t)/k - O(Y_e^t)| < 2 \\
&\quad \forall i \in \mathbb{N}^* : \\
&\quad (|T(X_s^{i \cdot k}) - T(Y_s^i)| < \delta) \wedge \\
&\quad (|T(X_e^{i \cdot k}) - T(Y_e^i)| < \delta) \wedge \\
&\quad (T(X_e^{i \cdot k}) + \delta < T(Y_s^{i+1})) \wedge \\
&\quad (T(Y_e^i) + \delta < T(X_s^{i \cdot k+1}))
\end{aligned}
$$

F. Precedence

Task X precedes Y iff. at any time, the occurrence of X is larger than or equal to the occurrence of Y. This implies X_s^i must precede Y_s^i, however it is not necessary to also have X_e^i precedes Y_s^i in all context. So a supplementary parameter to clarify the strict level of specification is maintained. From $\mathcal{L}_1$ to $\mathcal{L}_3$, it represents respectively "less strict", "strict" and "very strict".

Definition 9: Precedence - Finite Time Scope

$$
\begin{aligned}
&P_{ft}(X,Y,\delta,\mathcal{L}_1) \equiv \\
&\quad \forall i \in [1, O(X_s)] : T(X_s^i) + \delta < T(Y_s^i) \\
&P_{ft}(X,Y,\delta,\mathcal{L}_2) \equiv \\
&\quad \forall i \in [1, O(X_s)] : \\
&\quad (T(X_s^i) + \delta < T(Y_s^i)) \wedge (T(X_e^i) + \delta < T(Y_e^i)) \\
&P_{ft}(X,Y,\delta,\mathcal{L}_3) \equiv \\
&\quad \forall i \in [1, O(X_s)] : T(X_e^i) + \delta < T(Y_s^i)
\end{aligned}
$$

The original semantic cannot be applied in infinite time scope. A computable version reduces the semantic so that it is the same to the causality definition in infinite time scope.

G. Causality

Causality is similar to Precedence, except that it requires the maximum possible occurrence of X equals to that of Y, because each occurrence/execution of X causes each correspondent occurrence of Y.

Definition 10 : Causality - Finite Time Scope

$$
\begin{aligned}
&C_{ft}(X,Y,\delta,\mathcal{L}_1) \equiv \\
&\quad O(X) = O(Y),\ P_{ft}(X,Y,\delta,\mathcal{L}_1) \\
&C_{ft}(X,Y,\delta,\mathcal{L}_2) \equiv \\
&\quad O(X) = O(Y),\ P_{ft}(X,Y,\delta,\mathcal{L}_2) \\
&C_{ft}(X,Y,\delta,\mathcal{L}_3) \equiv \\
&\quad O(X) = O(Y),\ P_{ft}(X,Y,\delta,\mathcal{L}_3)
\end{aligned}
$$

Definition 11 : Causality - Infinite Time Scope

$$
\begin{aligned}
&C_{ift}(X,Y,\delta,\mathcal{L}_1) \equiv \\
&\quad \forall t \in \mathbb{R}_+ : |O(X_s^t) - O(Y_s^t)| < 2 \\
&\quad \forall t \in \mathbb{R}_+ : |O(X_e^t) - O(Y_e^t)| < 2 \\
&\quad \forall i \in \mathbb{N}^* : T(X_s^i) + \delta < T(Y_s^i) \\
&C_{ift}(X,Y,\delta,\mathcal{L}_2) \equiv \\
&\quad \forall t \in \mathbb{R}_+ : |O(X_s^t) - O(Y_s^t)| < 2 \\
&\quad \forall t \in \mathbb{R}_+ : |O(X_e^t) - O(Y_e^t)| < 2 \\
&\quad \forall i \in \mathbb{N}^* : \\
&\quad (T(X_s^i) + \delta < T(Y_s^i)) \wedge (T(X_e^i) + \delta < T(Y_e^i)) \\
&C_{ift}(X,Y,\delta,\mathcal{L}_3) \equiv \\
&\quad \forall t \in \mathbb{R}_+ : |O(X_s^t) - O(Y_s^t)| < 2 \\
&\quad \forall t \in \mathbb{R}_+ : |O(X_e^t) - O(Y_e^t)| < 2 \\
&\quad \forall i \in \mathbb{N}^* : T(X_e^i) + \delta < T(Y_s^i)
\end{aligned}
$$

V. Decomposition: Low-Level Computable Property Set

All the above definitions can be decomposed into a set of properties (Table II and Table III) which are computable by TPN model checker.

In general, our method will add some supplementary TPN elements to the original TPN generated from the model to compute whether these properties are guaranteed. The detailed computation method for each of these elements

Table II
FINITE TIME SCOPE COMPUTABLE PROPERTY SET

Low-level Property	Formal Definition
Max Occurrence Count	$\forall i \in \mathbb{N}^* : O(X_s^i) < constant$
Min interval	$\forall i,j \in \mathbb{N}^* : T(E_1^i) - T(E_2^j) > \delta$
Max interval	$\forall i,j \in \mathbb{N}^* : T(E_1^i) - T(E_2^j) < \delta$

Table III
INFINITE TIME SCOPE COMPUTABLE PROPERTY SET

Low-level Property	Formal Definition
Occurrence difference	$\forall t \in \mathbb{R}_+, k \in \mathbb{N}^* :$ $\lvert O(X_s^t)/k - O(Y_s^t)\rvert < \delta$
Min interval	$\forall i \in \mathbb{N}^*, k \in \mathbb{N}^*, b \in \mathbb{N}, j = i \cdot k + b :$ $\lvert T(E_1^i) - T(E_2^j)\rvert > \delta$
Max interval	$\forall i \in \mathbb{N}^*, k \in \mathbb{N}^*, b \in \mathbb{N}, j = i \cdot k + b :$ $\lvert T(E_1^i) - T(E_2^j)\rvert < \delta$

is described in another article. In this paper we give out the computational complexity of unfolding one TPN as the computation unit. All these computable properties need only one time TPN state space generation to know whether it is true or false. An obvious advantage of this approach is that as each computation unit is independent to the others, a parallel computation platform is then adaptable for this approach to deploy.

VI. COMPUTATIONAL COMPLEXITY & PERFORMANCE ANALYSIS

A. Computational complexity

We present here a complete complexity analysis (Table IV) for the computation cost of all the mentioned high-level time constraint properties, including both finite and infinite time range scope. In the finite time range scope, computational complexity depends on the complexity of system's design. For simplicity of its presentation, we use low-level computable property as the computation unit, and denote A for upper bound of max occurrence of task in the finite time range, which varies in accordance to system's design. We denote $B = A \cdot \log_2 A$. In the infinite time range scope, the computational complexity is independent of the system's design, which leads that its complexity is a constant.

B. Performance Analysis

We present the computational performance for each low-level computable property, then the computational performance of time constraints can be deduced by using the upper complexity table. Two aspects concerning performance issue are taken into account: to assess that the property is true, and to prove it false.

Our method is relatively independent of the system size and fast enough to assess that some low-level computable properties are false. In table V, we categorize these low-level properties by this character of independence.

Table IV
COMPUTATIONAL COMPLEXITY OF TIME CONSTRAINT DEFINITION

Property	finite	infinite
Coincidence	$6A + 2B$	6
Exclusion	$6A^2$	8
Sub-occurrence	$7A^2 + 2B$	6
Precedence (less strict)	A	3
Precedence (strict)	$2A$	4
Precedence (very strict)	A	3
Causality (less strict)	$A + 2B$	3
Causality (strict)	$2A + 2B$	4
Causality (very strict)	$A + 2B$	3
Synchronization	$2A + 4B$	4

Table V
DEPENDENCE OF LOW-LEVEL PROPERTY

Low-level Property	Proof: true	Proof: false
Min Interval (finite)	*dependent*	*dependent*
Max Interval (finite)	*dependent*	*dependent*
Max Occurrence Count (finite)	*dependent*	*independent*
Occurrence Difference (infinite)	*dependent*	*independent*
Min Interval (infinite)	*dependent*	*independent*
Max Interval (infinite)	*dependent*	*independent*

On the other hand, to prove the property is true depends on the system feature, so it is important to measure the performance influence to the original TPN by the added TPN elements. It is computed by comparing the state space generation time of the original TPN and the TPN modified by our method. In our test scope, it demonstrates that the over-cost of the observer is very slight, which implies if the original TPN can terminate its state space generation in an acceptable time range, the time constraint is also computable.

In order to make this performance result be significant to demonstrate that the method is applicable for pragmatic RTES, we randomly generate the system which scales from 2 to 10 parallel threads, in which each thread dispose of 10-100 periodic tasks. As we need to compare both finite and infinite scenarios, the upper bound of these periodic tasks' occurrence in finite scenario is set to 100.

As shown in Figure 5, 6 and 7, the performance varies in scope of 40% variation, which means it is very stable.

VII. RELATED WORKS

Several formal specifications of timing constraint exist. CCSL standardizes clock constraint semantic within the UML/MARTE framework to formally express causal and temporal constraints between the previously defined clocks and proposes a process to model time specification. It defines a complete set for clock constraints, which can be seen as driven by instantaneous events. However, as it focus on the

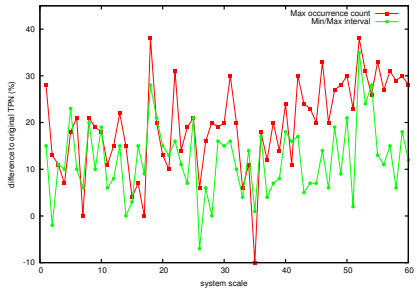

Figure 5. Performance Influence: Finite System Measured in Finite Time Range Scope

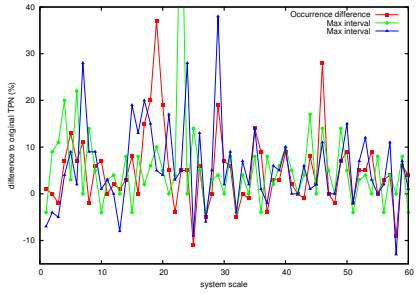

Figure 6. Performance Influence: Finite System Measured in Infinite Time Range Scope

low-level event concept, there is still a gap in UML/MARTE for task-level's temporal specification and their verification.

Concerning the verification method, CCSL transforms UML model to SyncCharts, and uses Esterel assertions to express clock constraints. This language has a well-defined notion of instant, and at each reaction, any signal has a unique status. This is not the case with non-strictly synchronous languages [7]. It is thus less applicable in a real RTES scenario, as the designers need always a time duration to tolerate. Another issue is that, in [8], the author gives also an intuitive interpretation on Time Petri Net for a chosen sub-set of CCSL constraints. However, the intuitive interpretations are not quite suitable for computation. Compared to the work of CCSL, we avoid these problems for both specification and verification aspect in our work.

VIII. Conclusion

We have first given an overview of our framework enabling model-checking for UML/MARTE in RTES's MDE approach. Then we focus on the formal semantic for time constraint at task-level, which extends the capacity to define the verification requirement for pragmatic RTES. It also extends these specifications to cover infinite time range scenarios, which makes the verification of periodic system to be more practical. Further, the correspondent verification method is presented to demonstrate that these temporal properties' verification based on both finite and infinite time range are computable. Finally, it is shown that our method

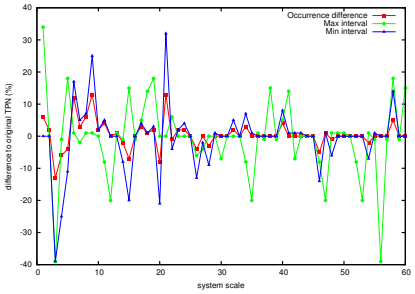

Figure 7. Performance Influence: Infinite System Measured in Infinite Time Range Scope

is not too expensive, which adds value to have it applied in real RTES's verification.

The usage of this framework would shorten the RTES development time by accelerating the V&V cycle at model level. It is also presented that all the computation units for verification is independent to each other and then can be engaged natively in parallel computation environment. Our future work is then to apply this approach on real industrial applications.

References

[1] *OMG Unified Modeling Language^{TM} (OMG UML), Superstructure*, Object Management Group, Inc., Feb. 2009.

[2] *UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems Version 1.0*, Object Management Group, Inc., Nov. 2009.

[3] N. Ge and M. Pantel, "Time properties dedicated semantics for uml-marte safety critical real-time system verification," July 2012, submitted to 8th European Conference on Modelling Foundations and Applications (ECMFA). [Online]. Available: http://hal.archives-ouvertes.fr/hal-00675778

[4] P. Merlin and D. Farber, "Recoverability of communication protocols–implications of a theoretical study," *Communications, IEEE Transactions on*, vol. 24, no. 9, pp. 1036 – 1043, sep 1976.

[5] 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*, vol. 42, no. 14, pp. 2741–2756, 2004.

[6] M. Peraldi-Frati and J. DeAntoni, "Scheduling multi clock real time systems: From requirements to implementation," in *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2011 14th IEEE International Symposium on*, march 2011, pp. 50 –57.

[7] C. André, "Verification of clock constraints: CCSL Observers in Esterel," INRIA, Rapport de recherche RR-7211, Feb. 2010.

[8] F. Mallet and C. Andre, "On the semantics of uml/marte clock constraints," in *Object/Component/Service-Oriented Real-Time Distributed Computing, 2009. ISORC '09. IEEE International Symposium on*, march 2009, pp. 305 –312.