# Handling Global and Local Time and Energy Constraints of Sequence Diagrams

Vinícius Camargo Andrade
*Department of Computer Science*
Federal Technology University - Paraná
Ponta Grossa, Brazil
vcandrade@utfpr.edu.br

Letícia Mara Peres
*Department of Computer Science*
Federal University of Paraná, UFPR
Curitiba, Brazil
lmperes@inf.ufpr.br

Marcos Didonet Del Fabro
*Department of Computer Science*
Federal University of Paraná, UFPR
Curitiba, Brazil
marcos.ddf@inf.ufpr.br

***Abstract*—The Unified Modeling Language (UML) has been widely adopted for modeling different sorts of applications. Despite having several kinds of diagrams, they were not designed verifying the execution of real-time embedded systems with time and energy constraints. There are UML profiles that capture this information, but it is necessary to rely on a separated validation framework. The main approach to fill this gap is to translate UML the models into representations such as Petri nets. However, existing works have little support for addressing energy and time constraints at the same time. This paper presents a technique for transforming UML sequence diagrams with energy and time constraints into timed Petri net models. These Petri net models are then used as input into software verification tools like Tina and GTT.**

***Keywords—Sequence Diagram; Time Petri Nets; Meta-Modeling; Model Transformation; Embedded Software***

## I. INTRODUCTION

The execution time of Real-Time Embedded (RTE) systems needs to follow firm time constraints. It may be necessary to halt for a moment in order to follow the defined time constraints. For this reason, the utilization of methods to predict or simulate the execution time is of major importance [1]. Not less important, energy constraints may also be used when developing embedded systems. For example, when the hardware manufacturers need to produce complex products with low power consumption [2].

The OMG (Object Management Group) has created a Unified Modeling Language (UML) profile in order to assist into RTE modeling task. This profile is called MARTE (Modeling and Analysis of Real Time and Embedded systems) [3]. MARTE has constructs adapted to the design of real-time embedded systems. This profile is used in different kinds of applications, such as performance analysis; scheduling; or resource allocation. Main goal of existing works is to validate the time and energy constraints of a given RTE application. The most common approach is to translate an informal model into a formal model of modeling such as UML Sequence Diagrams (SD) in Petri nets which is used as input to engines such as Tina or GDD. However, most part of existing works do not handle both time and energy constraints; or the transformation is done manually [2].

We present in this paper an approach that checks the consistency of UML Sequence Diagrams with energy and time constraints. We translate the Sequence Diagrams into a Petri Net model supporting both kinds of constraints. We used ATL (AtlanMod Transformation Language) [4], a well-known transformation language. This model is further translated into a Timed Petri net (TPN) solver [5], which performs the checking. Thus, the main contributions are the following: we provided minor extensions to Sequence Diagrams and to an existing Petri Net metamodels. We defined a set of transformation rules between SDs and Petri Net. An experiment was undertaken applying the approach to a single application.

This article is organized as follows. The backgrounds are defined in the Section 2. Section 3 presents the extensions applied to the input and output metamodels, as well as the proposed transformations. Section 4 shows an experimental evaluation and presents the comparison between the transformations carried out. The related works are presented in Section 5. Finally, Section 6 shows conclusions of the paper.

## II. BACKGROUND

We discuss in this section, UML sequence diagram and timed Petri net concepts.

### A. Sequence Diagrams (SD)

Models are real-world abstractions and are mainly used to document software systems. In Model Driven Engineering approaches the models may be organized in a three level architecture, the so called terminal model, meta-model and meta-meta-model [4, 6].

A terminal model is a representation of a given system, often called instances. The language which enables defining terminal models is depicted in metamodels. The relation conformance enforces a terminal model and has only instances of elements of a metamodel. Finally, meta-meta-models define the core concepts for a giving modelling architecture, and they are self descriptive. In our work, we use definitions from [7].

The UML Sequence Diagram metamodel defines interactions between actors and classes or operations initiated by them [8].

### B. Timed Petri Net (TPN)

Timed Petri Net (TPN) [9] is a variation of a Petri Net [10]. It is a graphic and mathematical model for describing systems with concurrent, asynchronous, distributed, parallel, non deterministic and stochastic time characteristics. It is applied in distinct fields, such as manufacture, distributed computing and real-time computing.

A TPN, R is given by a tuple (P, T, Pre, Post, $M_0$, I, E), where: P is a finite set of places; T is a finite set of transitions; Pre: (P x T) → is a input function for transitions, which represents the weight of the transition input arcs; Post: (P x T) → is an incidence function of

We would like to thank CAPES and CNPq (Universal A 14/2011) for partially funding this research.



DOI 10.1109/UKSim.2018.00025

transitions output, which represents the weight of arcs of transitions outputs; $M_0$: P → is the initial marking; I: T → ( +x (ℚ+ ∪ {∞})) is the function which associates a firing time interval at each transition, being + the set of non-negative rational numbers; E: T → ( +x (ℚ+ ∪ {∞})) is the function which associates a numeric interval at each transition, being + the set of non-negative rational numbers.

A timed Petri net possesses an interval associated to the transition which can be named as T[X, Y], where: T is the name of transition; X is the minimum value of the constraint; Y is the maximum value of the constraint.

## III. Time and Energy Validation in Sequence Diagrams

In this section, we present our approach to translate SDs into Petri Nets and to validate it in TPN. An overview of solution is illustrated in Figure 1. Step 1 refers to development of Sequence Diagrams. Our transformation acts in step 2. This step is developed in three stages: stage 1, adapts SD and TPN metamodels; stage 2, defines transformations between the model elements; and stage 3, shows the result obtained from the transformation. And, at last, step 3, refers to the time analysis performed in the generated TPN.

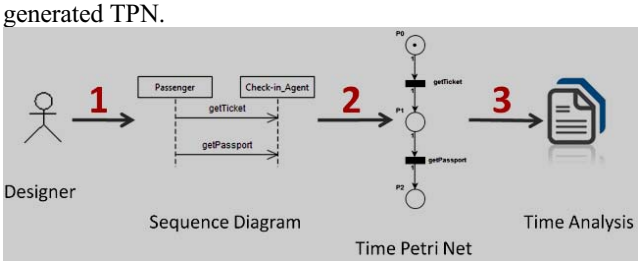



Fig. 1. Overview of the work performed.

### A. Input and Output Metamodels

Different existing works using SD metamodels are analyzed: [7, 11, 12, 13, 14], and TPN metamodels: [13, 15, 16, 17] have specifications with no native support for time and energy constraints. In this context, we adapt both metamodels in order for them to contemplate such restrictions. We emphasize that once both metamodels have been defined and adapted, they are used to perform the transformations, that is, it is not necessary to rework the adaptations for each new transformation. In this subsection we describe the adaptations performed.

We define time and energy constraints specified in seven distinct structures on a UML SD: the exchange of a specific message; the complete execution time of an action; in the execution of the internal elements of a combined fragment: *Option*, *Alternatives*, *Parallel*, *Loop* and *Resource Usage*.

Figure 2 illustrates the original SD metamodel [14] and the adaptations concerning the constraints associated to the message (in gray).

Time constraints are represented by the class *ExecTime*, which contains the minimum and maximum attributes related to time. The minimum and maximum energy constraints are represented by the attributes of the class *Energy*. Both classes inherit the abstract class Resource which is linked to the class *Message*.

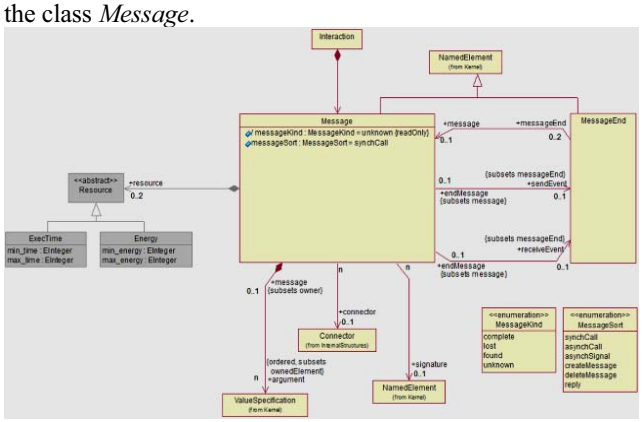



Fig. 2. Adaptation performed in order to represent the constraints in a specific message [14].

Other adjustments made in elements of UML SD are similar to the adaptation illustrated in Figure 2. Alteration occurs in the association between the abstract class *Resource* and the class which refers to the element that will contain the time and energy constraints.

Amongst several existing TPN metamodels, we have chosen to adapt the solution proposed in [17], since it already contains time constraints, but do not energy constraints. The metamodel corresponding to the TPN is represented in Figure 3. The element *Transition* presented in gray was the only one modified. It already had the attributes *min_time* and *max_time*, making it necessary to add energy constraint, represented by attributes *min_energy* and *max_energy*.

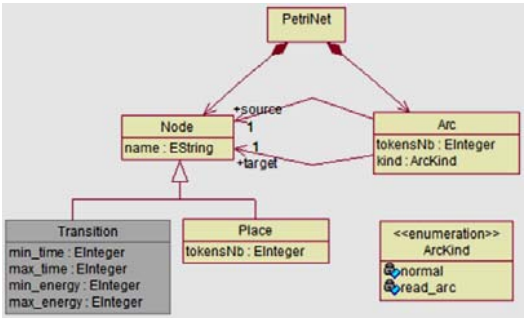


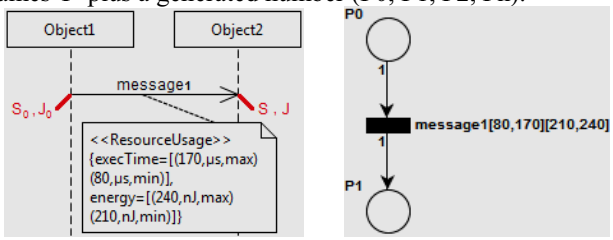

Fig. 3. Adaptation performed to represented the time and energy constraints in a TPN. Adapted from [17].

### B. Sequence Diagram to Petri Nets Transformation

In this section, we describe the transformation rules between the input elements of an SD into the output elements of a timed Petri Net. Different specifications were analysed to choose the elements used in the transformation. The selected SD elements are: *message*, *execution specification* and *combined fragment*. We use simple artificial input and output instances to illustrate each transformation. The implemented transformation combines all these rules and can be used to transform complex SDs into TPNs.

*1) Message:* In Figure 4 (a) is an exchange of messages with time and energy constraints associated to a message.

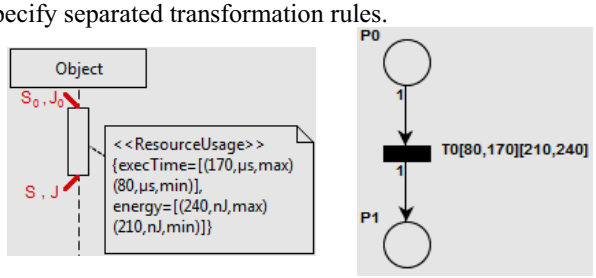
When firing a message, initial values time ($S_0$) and energy ($J_0$) are assigned. Once the recipient object receives the message, the values of final time (S) and final energy (J) are assigned. The exchange of message is transformed into a time Petri sub-net (Figure 4 (b)) having respectively, place, arc, transition, arc and place. The transition receives the same name and message constraints being exchanged between the objects, on the other hand, the places receives names 'P' plus a generated number (P0, P1, P2, Pn).




a) b)

Fig. 4. UMLSD2TPN applied to the exchange of messages containing time and energy constraints.

*2) Execution Specification:* Figure 5 (a) presents a resource usage restriction applied into the *execution specification* element. The initial values of time ($S_0$) and energy ($J_0$) are assigned at the start of the execution. The final (S) and (J) are assigned at the end of the execution. While the input element is different, the output Petri Net elements (Figure 5(b)) follow the same format than the translation of simple messages. Still, it is necessary to specify separated transformation rules.

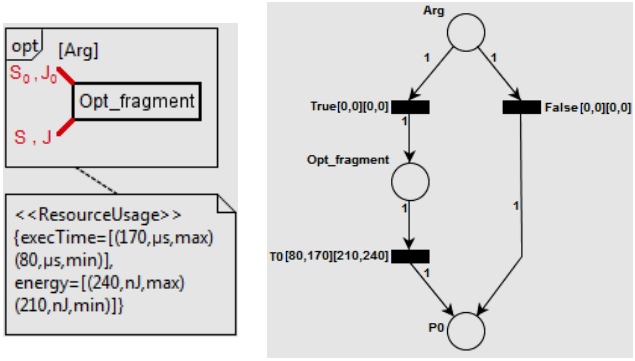



a) b)

Fig. 5. UMLSD2TPN applied to the element Execution Specification containing time and energy constraints.

*3) Combined Fragment:* is composed by other simpler elements. There are different kinds of combined fragments. It is noteworthy that the original specification does not support constraints. Thus, we use the extended SD metamodels.

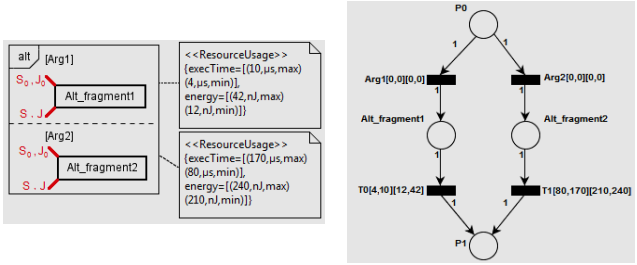
*a) Option (opt):* the *Opt_fragment* is executed if a condition depicted in *Arg* is true. If the condition is false, nothing is executed (Figure 6 (a)). The respective timed Petri subnet is shown in Figure 6 (b). This subnet starts with a place that receives the name of the condition associated with this place, with two transitions, True and False. Both have zeroed constraints. The time and energy constraints are assigned to the last transition after the execution of *Opt_fragment*.




a) b)

Fig. 6. UMLSD2TPN applied to the opt decision structure.

*b) Alternatives (alt):* This structure represents two or more alternative flows and is illustrated in Figure 7 (a). It needs to have an interaction fragment called ELSE, so if none of previous conditions are met, something is run. This combined fragment has a constraint associated with each interaction operand. The respective timed Petri subnet is shown in Figure 7 (b). and is initiated by a place P0 attached to N transitions, where N is the number of interaction operands. Each transition receives the name of the condition and zeroed constraints, as they only represent a choice of flow. These transitions are associated with a place which represents the executions of these internal operands.

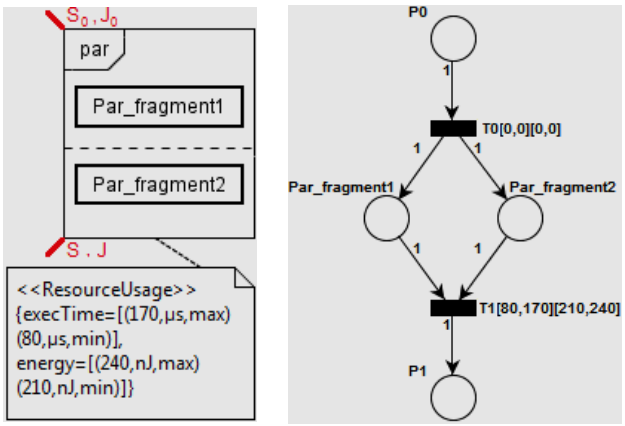



a) b)

Fig. 7. UMLSD2TPN applied to complex decision structure.

*c) Parallel (par):* Defines two or more interaction operands are executed in parallel, lacking conditions. Figure 8 (a) shows a parallel execution of elements *Par_fragment1* and *Par_fragment2*, with a global time and energy constraint associated. The respective timed Petri subnet is shown in Figure 8 (b). The core idea is similar to the previous transformation rules: the initial transitions have zeroed constraints and the last transition contains the value of the time and energy constraints. It may have a multiple number of parallel executions.

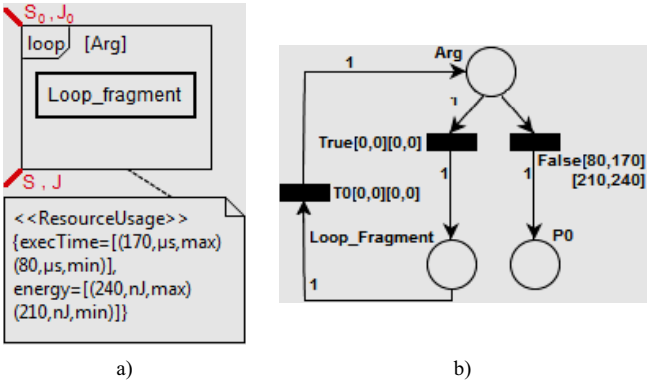
*d) Loop (loop):* Figure 9 (a) shows the corresponding SD. While the condition *Arg* is satisfied, the elements in the combined fragment are executed repeatedly. The timed Petri subnet is illustrated in Figure 9 (b). The initial place gets its name from the input condition in the combined fragment. This place is associated with two transitions: True and False, which has the constraints zeroed because it presents only flows




a) b)

Fig. 8. UMLSD2TPN applied to parallel execution.

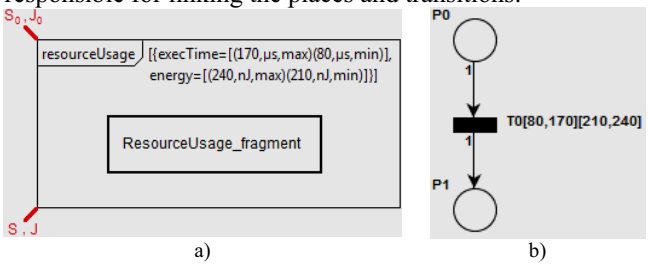
*Loop_Fragment*, from transition T0, returning to the place of origin. The False transition contains constraints imposed on SD.




a) b)

Fig. 9. UMLSD2TPN applied to the loop.

*e) Resource Usage (resourceUsage):* this combined fragment is illustrated in Figure 10 (a). This element enables the definition of constraints into groups of elements, not only to particular elements as in the previous rules. This means we can define global time and energy constraints into a block of execution. The Petri subnet related to the combined fragment Resource Usage is illustrated in Figure 10 (b) and is formed by only two places (P0 and P1), a transition, which is named T0, containing the constraints of the combined fragment of UML SD, and two arcs, responsible for linking the places and transitions.




a) b)

Fig. 10. UMLSD2TPN applied to the combined fragment resource usage.

## IV. Results and Discussion

In this section, we present transformations and time analysis of models carried out in a case study. To execute the transformations, we use ATL transformation language, the metamodels were implemented using EMF metamodels and the instances are EMF models. To illustrate, we use a UML sequence diagram for a pulse oximeter system [2], shown in Figure 11.

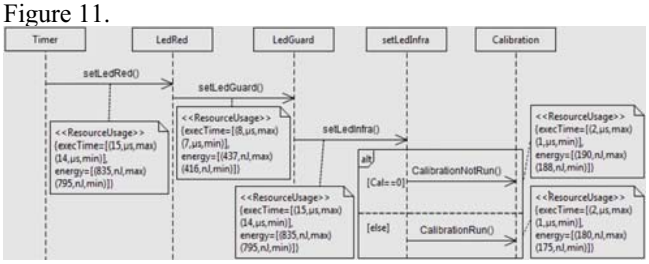



Fig. 11. Sequence diagram of a pulse oximeter [2].

This transformation combines different elements, as messages and combined fragments of the alternative type, to produce a unique and valid Petri net. The resulting timed Petri Net is illustrated in Figure 12.

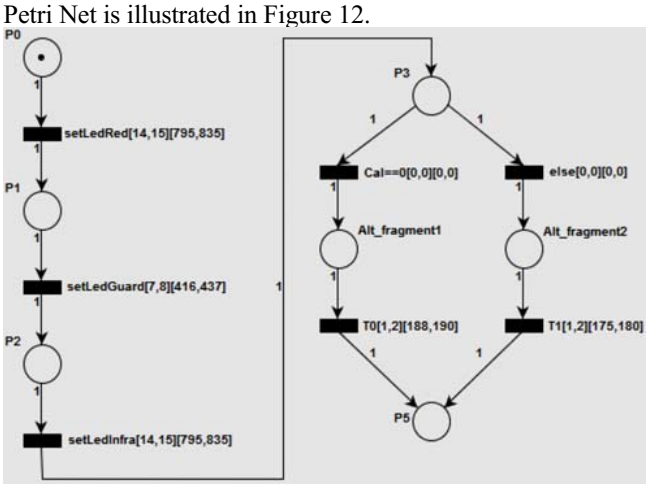



Fig. 12. Timed Petri subnet referring to pulse oximeter.

### A. Time Analysis

In this section we present the results of the time analysis. We use a class graph, generated by GTT tool [5, 18], that shows all execution alternatives of the corresponding TPN. This graph represents the possible firings of this TPN. Each node represents a class $C_{k\_n} \in C$. Each arc is a transition fired in the class $C_{k\_n}$ that generates a successor class $C_{(k+1)\_n}$. where $k$ is the level and n is an identifier. It contains as well the tagging vector of the class with global time. The following parameters in the node represent the name of the transition $t_i$; its time on *rk* $(t_i)$; its adjustment coefficient $ac_k(t_i)$; and its global time $g_k(t_i)$.

Figure 13 illustrates the class graph corresponding to our example. The starting point of this class graph is given by the node *C0_0* which has a relative time *R = [14, 15]* if firing *setLedRed* transition. The next node is represented by *C1_1*

class which has global time *G = [14, 15]* and relative time *R = [7, 8]* if *setLedGuard* is fired. After *setLetInfra* firing, *C3_3* class has global time *G = [35, 38]*. From this node it is possible to fire for transitions *CalIgual0* or *else*. Firing *CalIgual0* transition, follows transition *T0*, finishing in *C5_6* class with global time *G = [36, 40]*. On the other hand, following the sequence of firings per *else* transition (from *C3_3* class), this is accompanied by *T1* transition and final global time *G = [36, 40]* in class *C5_7*.

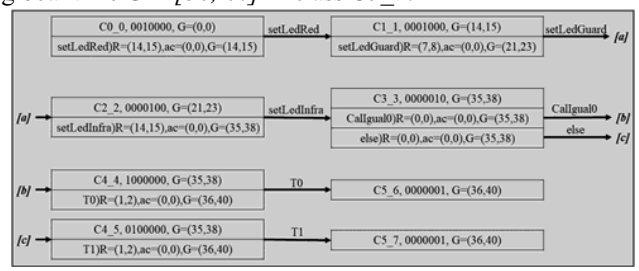



Fig. 13. Graph of 5 levels classes, of TPN from Figure 12 representing time.

### B. *Transformations*

Table 1 compares six SD and TPN models used in experiments. As comparison characteristics, we used the total number of: messages (Msg), execution specification (Exec. Spec.), combined fragments (Comb. Frag.), places and transition (Trans.) of each model.

The first model is related to basic processes of an Automated Teller Machine (ATM) [19]. The second model [20] is a framework responsible for simulating the behavior of real-time embedded systems. The third model illustrates the steps of a check-in agent for an airline [21]. A pulse-oximeter system [2] is the fourth model. The model of [22, 23] refers to a process of a security breach by an attacker. Finally, the sixth example is an elevator system proposed by [21].

TABLE I. COMPARISON OF TRANSFORMATIONS

| Model | Msg | Exec. Spec. | Comb. Frag. | Places | Trans. |
|---|---|---|---|---|---|
| ATM | 14 | 3 | 1 | 14 | 13 |
| Framework | 13 | 2 | 1 | 12 | 11 |
| Check-in | 10 | 0 | 1 | 12 | 10 |
| Pulse-oximeter | 5 | 0 | 1 | 7 | 7 |
| Security breach | 12 | 2 | 1 | 14 | 14 |
| elevator system | 8 | 0 | 2 | 8 | 9 |

## V. RELATED WORK

Several metamodels of timed Petri Net (TPN) are found in the literature. The metamodel proposed by [13] comprises a set of Places, Transition and Arcs (InputArc and OutputArc) having the function of interconnecting Places and Transitions. On the other hand, the Petri net is formed by a set of Marking, which attaches a number of marks for each place.

Regarding the works of transformation models in the literature, many of these works propose manual methods based on graphical structures of the UML and Petri net (PN). The work from [24] presents a transformation of the Activity Diagram of UML2 in a Petri net and colored Petri net through the Fundamental Modeling Concept Petri Net diagram (FMC-PND). In [25], the authors address the transformation of models using Triple Graph Grammar (TGG) [26]. Kerkouche [27] propose a model transformation of the state diagram and the collaboration diagram into a colored Petri net using graphs. Their work aims to make a connection between formal and informal notation for purposes of analysis and simulation techniques. Du [28] addresses the mapping of a class diagram and SD into a colored Petri net based on the technique of Edged Graph Grammar (EGG) [29]. Soares [30] uses ETL (Epsilon Transformation Language) technology to implement rules for converting UML sequence diagrams into colored Petri net.

Ameedeen [31], show a transformation of SD into a Petri net using Query/View/Transformation (QVT) [32]. Ribeiro [21] performs a transformation of models involving SD in a colored Petri net. Despite the fact that these works perform a model transformation in an automated way, they do not consider constraints in their models.

Still in the work of [31], conversion of SD to Petri net is unidirectional transformation process, that is, it is not possible to perform the reverse conversion. In this context, [33] propose rules to transformation of Petri net models to UML sequence diagram.

In [34], is propose a verification framework that can assess time properties like upper bounds 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 synchronization-related properties. This verification occurs on a timed Petri net that comes from a transformation of a UML/MARTE sequence diagram.

The work of [2] is the closest to ours. It transforms a model of source SD into a target model TPN and it also contemplates the constraints in his models. However, in this work the transformation is performed with manual mapping. On the other hand, our work uses ATL, a declarative and imperative language of model transformation, its organization of rules possesses mechanisms of modulation and reuse, organizational structure and source orientation, also it supports any model that can be defined in a metamodel Meta-Object Facility (MOF) [35].

## VI. CONCLUSIONS

This paper presents a model transformation between UML Sequence Diagrams and Timed Petri Net, both with time and energy constraints. For this, the sequence diagram was extended with the MARTE profile allowing the representation of constraints which could not previously be represented.

For the accomplishment of transformations, were collected in the literature metamodels of Sequence Diagram

and Timed Petri Net. However the existing metamodels did not consider the time and energy restrictions. In this context, we made adaptations in these metamodels to be able to carry out the proposed transformation.

With the metamodels defined and adapted, we implemented a set of rules to carry out the transformation automatically. In addition, the tool UMLSD2TPN was implemented to convert the generated timed Petri net into an input format for formal verification tools such as TINA [36] and GTT [5].

## REFERENCES


[1] J. A. Stankovic. "Misconceptions about real-time computing: A serious problem for next-generation systems." Computer 21.10. 1988 : 10-19.

[2] E. Andrade, P. Maciel, G. Callou, B. Nogueira, C. Araújo. “Mapping uml sequence diagram to time petri net for requirement validation of embedded real-time systems with energy constraints”. Proceedings of the 2009 ACM symposium on Applied Computing, SAC '09, pages 377-381, New York, NY, USA, 2009. ACM.

[3] Object Management Group OMG. “Meta object facility (mof) 2.0 uml profile for marte”, June 2011.

[4] LINA & INRIA ATLAS group. “Atl: Atlas transformation language”, 2005.

[5] L. M. Peres, L. A. Künzle and E. Todt. “Applying global time Petri net analysis on the embedded software context”. Sba Controle & Automação, 2011, vol.22, no.6, p.610-619. ISSN 0103-1759.

[6] T. Gherbi, D. Meslati and I. Borne. "MDE between Promises and Challenges”. In Computer Modeling and Simulation, UKSIM'09. 11th International Conference. IEEE, no. 152-155, March 2009.

[7] H. Shen, A. Virani and J. Niu. “Formalize uml 2 sequence diagrams”. High-Assurance Systems Engineering, IEEE International Symposium on, 0:437 - 440, 2008.

[8] C. Larman. “UML and patterns: an introduction to object-oriented analysis and design and the unified process”. Prentice Hall, 2nd ed, p. 656, 2002.

[9] B. Berthomieu and M. Diaz. “Modeling and verification of time dependent systems using time Petri nets”. IEEE Trans. Softw. Eng., 17(3):259-273, March 1991.

[10] T. Murata. “Petri nets: Properties, analysis and applications”. Proceedings of the IEEE, v. 77, n. 4, p. 541–580, 1989.

[11] V. Garousi, L. Briand and Y. Labiche. “Control flow analysis of uml 2.0 sequence diagrams”. Technical Report, Proc. of European Conference on Model Driven Architecture-Foundations and Applications (ECMDA-FA), 2005.

[12] M. Thongmak and P. Muenchaisri. “Design of rules for transforming uml sequence diagrams into java code”. Asia-Pacific Software Engineering Conference, 0:485, 2002.

[13] M. A. Ameedeen, B. Bordbar and R. Anane. “Model interoperability via model driven development”. J. Comput. Syst. Sci., 77(2):332 - 347, 2011.

[14] Object Management Group OMG. Omg unifed modeling languagetm (omg uml), superstructure. 2010.

[15] B. Combemale, X. Crégut, P. Garoche and X. Thirioux. “Essay on semantics de nition in mde - an instrumented approach for model verification JSW”, 4(9):943-958, 2009.

[16] E. Breton and J. Bézivin. “Towards an understanding of model executability”. Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001, FOIS '01, pages 70-80, New York, NY, USA, 2001. ACM.

[17] B. Combemale, X. Crégut, P. Garoche, X. Thirioux and F. Vernadat. “A Property-Driven Approach to Formal Verification of Process Models”. Enterprise Information System IX, volume 12, pages 286-300. LNBIP, Springer, 2008.

[18] L. M. Peres. Proposta de um Método de Verificação por Tempo Global com Redes de Petri no Desenvolvimento de Software Embarcado e em Tempo Real. Tese de Doutorado, Universidade Federal do Paraná, 2010.

[19] J. Whittle and J. Schumann. “Generating Statechart Designs from Scenarios”. In Proceedings of the 2000 International Conference on Software Engineering (ICSE 2000), Limerick, Ireland, 2000, pp. 314-323.

[20] M. A. Wehrmeister; J. G. Packer and L. M. Ceron. "Framework to Simulate the Behavior of Embedded Real-Time Systems Specified in UML Models." Computing System Engineering (SBESC), 2011 Brazilian Symposium on. IEEE, 2011.

[21] O. Ribeiro. “Animation-based validation of reactive software systems using behavioural models”. PhD thesis, Escola de Engenharia, Universidade do Minho, Minho, Portugal, 2009.

[22] K. Anastasakis, B. Bordbar, G. Georg, and I. Ray, “UML2Alloy: a Challenging Model Transformation”. In ACM/IEEE 10th international conference on Model Driven Engineering Languages and Systems, 2007, pp. 436-450.

[23] G. Georg, I. Ray, K. Anastasakis, B. Bordbar, M. Toahchoodee, and S. H. Houmb3, “An Aspect-Oriented Methodology for Developing Secure Applications”. Submitted to Information and Software Technoology, 2008.

[24] T. Staines. “Intuitive mapping of uml 2 activity diagrams into fundamental modeling concept Petri net diagrams and colored petri nets”. Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, ECBS '08, pages 191-200, Washington, DC, USA, 2008. IEEE Computer Society.

[25] A. Staines. “A triple graph grammar mapping of uml 2 activities into Petri nets”. International Journal Of Computers, 4(1):27, nov de 2009.

[26] E. Kindler, R. Wagner, “Triple Graph Grammers: Concepts, Extensions, Implementations and Application Scenarios”, Technical Report Tr-ri-284, University of Paderborn, Paderborn, Germany, 2007.

[27] E. Kerkouche, A. Chaoui, E. Bourennane, O. Labbani. “A uml and colored petri nets integrated modeling and analysis approach using graph transformation”. Journal of Object Technology, 9(4):25-43, July 2010.

[28] Z. Du, Y. Yang, J. Xu, e J. Wang. “Mapping uml models to colored petri nets models based on edged graph grammar”. Journal of Computational Information Systems, 11(7):3838-3845, November 2011.

[29] Z. Xiao-Qin , H. Xiu-Qing , Z. Yang. “An edge-based context-sensitive graph grammar formalism”. Journal of Software, 2008, 19(8):1893-1901.

[30] J. A. C. Soares. “Automatic Model Transformation from UML Sequence Diagrams to Coloured Petri Nets”. 2017.

[31] M. Ameedeen, B. Bordbar, R. Anane. “A model driven approach to the analysis of timeliness properties”. Pages 221-236, 2009.

[32] Object Management Group. “Qvt. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification." Final Adopted Specification (November 2005). 2008.

[33] T. W. Jie, and M. A. Ameedeen. "A Model Driven method to represent Free Choice Petri Nets as Sequence Diagram." Software Engineering and Computer Systems (ICSECS), 2015 4th International Conference on. IEEE, 2015.

[34] N. Ge, and M. Pantel. "Time properties verification framework for uml-marte safety critical real-time systems." Modelling Foundations and Applications. Springer Berlin Heidelberg, 2012. 352-367.

[35] Object Management Group. “Meta Object Facility (MOF) 2.0 Core Specification”. 2003.

[36] B. Berthomieu, F. Vernadat. “Time Petri nets analysis with TINA”, 3rd Int. Conf. on the Quantitative Evaluation of Systems (QEST’2006), Riverside (USA), 11-14 Septembre 2006, p. 123-124.