Benefits of using Petri nets for cyber-physical systems' controllers development in the classroom

Luis Gomes[1,2,3,*], Anikó Costa[1,2,3]

[1]NOVA University Lisbon, NOVA School of Science and Technology, Portugal
[2]Center of Technology and Systems (CTS), UNINOVA, Portugal
[3]Intelligent Systems Associate Laboratory (LASI), Portugal

Abstract

Cyber-Physical Systems (CPS) are increasingly becoming widespread in our daily lives. The development of embedded controllers, as a major constituent of CPS, needs to cope with the impact of the sustained complexity increase verified during the last decades. Model-driven development strategies supported by graphical formalisms having precise semantics and interactive tools allowing model editing and composition, simulation, verification, and automatic code generation, can provide a highly efficient way to achieve rapid prototyping and reliable implementations. These challenges have a strong impact when defining academic curriculum and selecting pedagogical approaches to teach related topics. Petri nets can provide this type of support, as they are amenable to provide support for the major characteristics presented in this type of controllers, namely concurrency and parallelism, conflict and resource sharing modeling, as well as support to modularity and composability. In this paper, a line of examples based on the analysis of a parking lot controller will be used illustrating on how the referred modeling challenges can be faced by students when using Petri nets to describe the behavior of associated embedded controllers. Several types of exercises are presented addressing different levels of complexity, considering different configurations of the parking lot infrastructure. This line of examples can be used at different types of courses, ranging from those focusing at system level modeling to those focusing on specific implementation strategies, where several classes of autonomous and non-autonomous Petri nets can be used.

Keywords

Embedded controllers, Modularity, Composability, Conflict resolution, Petri nets

1. Introduction

Cyber-Physical Systems (CPS), sometimes seen as a new generation of networked embedded systems, have been growing significantly in terms of visibility and importance. Among their characteristics, one can mention dependability, functioning in a reliable, safe, and secure manner, as argued in [1], which identifies research gaps and future directions for the different phases of the industrial system development life cycle. In parallel with other components, one of the CPS constituent parts is the controller part. Model-driven Development (MDD) approaches have an important role to play within their design methodologies. In order to cope with the above referred constraints it is of paramount importance to rely on a modeling formalism equipped with a precise execution semantics adequate to support a design automation approach supported by computational tools and covering all development phases, from specification to automatic execution code generation, and including tools for simulation and verification of behavioral properties.

Taking into consideration the main characteristics of most of the embedded controllers used in cyber-physical systems, Petri nets can be seen as the perfect candidate to be selected for the role of supporting modeling formalism. In particular, as the design of embedded controllers is intended, a non-autonomous class of Petri nets supporting explicit dependencies on its execution semantics to input and output signals and events is of interest, as will also allow their full integration in model-driven development approaches supporting automatic generation of execution code without writing a line of code.

PeNGE 2024 - Petri Net games, examples and quizzes for education, contest and fun, Geneva, Switzerland, June 25, 2024
*Corresponding author.
lugo@fct.unl.pt (L. Gomes); akc@fct.unl.pt (A. Costa)
0000-0003-4299-8270 (L. Gomes); 0000-0001-8147-028X (A. Costa)

In this paper the emphasis is put on the modeling and development tasks of the embedded controller to be carried out in the classroom. For that, a specific type of embedded controllers targeting parking lot access control is used. The advantages of selecting this class of applications are two-fold: being very well know by the students contributing to keep them involved and grab their interest, and exhibit different types of behavioral dependencies which can find in Petri nets a proper modeling formalism to use.

The paper is structured as follows. In the next section, the proposed challenge is presented, introducing a type of controlled systems to be modeled and analysed. In Section 3 a few relevant classes of Petri nets used for the referred tasks are identified. In Section 4 a discussion on the modeling strategies to be used is presented, followed by the Section 5 where a brief discussion is produced. Section 6 concludes.

2. The challenge

Students are challenged to tackle a practical problem involving a real-world scenario, considering a system's controller equipped with several physical sensors and actuators. Their task involves modeling both the behavior of the system and its actuators.

The proposed line of exercises relies on the analysis of a parking lot controller, including controlling its entrance and exit gates alongside monitoring parking capacity. Figure 1 shows the simplest parking lot configuration, with one entrance and one exit, where each entrance/exit has a presence detector on the floor to detect arrival of a car and a button to be activated by the driver (to get a ticket at the entrance, or to insert ticket after payment at the exit).

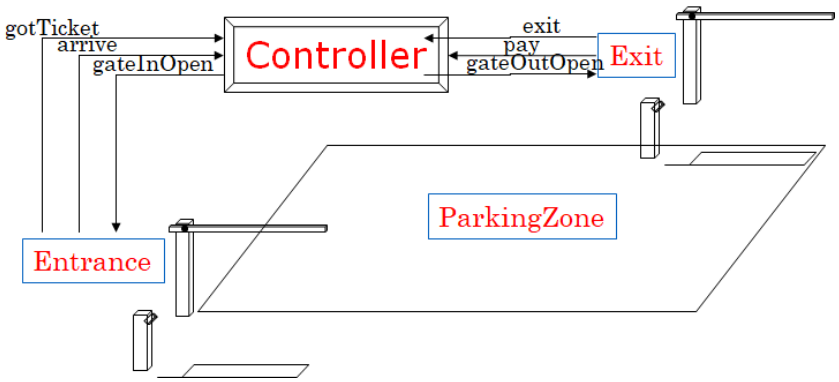


Figure 1: A simple parking lot with one entrance and one exit.

At this point, simultaneous activities at entrance and exit needs to be considered (locality on evaluation and concurrency modeling) in a similar way as in a producer-consumer system. Resource availability modeling is also present, as the number of free parking places can be seen as resources that are competitively shared by incoming users.

However, starting with this simplest parking lot configuration, several complications can be considered, namely:

- including several entrances, where some conflicts may occur, e.g when the number of cars trying to enter the parking lot is higher than the free places available inside.
- including also several exits as well as several parking areas or floors considering some modular structure of the parking lot (and where modular composition of the Petri nets models can be used).
- including additional sensors at entrances and exits to improve detection of the driver behaviour, e.g. leaving the entrance after trying to enter and goes reverse afterwards.

In the referred complications, several abstraction levels can be adopted, ranging from a generic description of the system behavior, to a fine detailed control actions allowing direct association with actuation of gates.

Also different types of goals can be addressed, ranging from high-level description and analysis of the system behavior, to generation of operational models to be directly deployed in the parking lot controller, including simulation of the system's behaviour considering randomly arriving and exiting of cars, among others.

From the above, it is clear that the modeling of several key characteristics of the system can be adequately accommodated through Petri nets modeling (locality, concurrency, conflicts, resources, and so on).

3. On the Petri net classes of interest

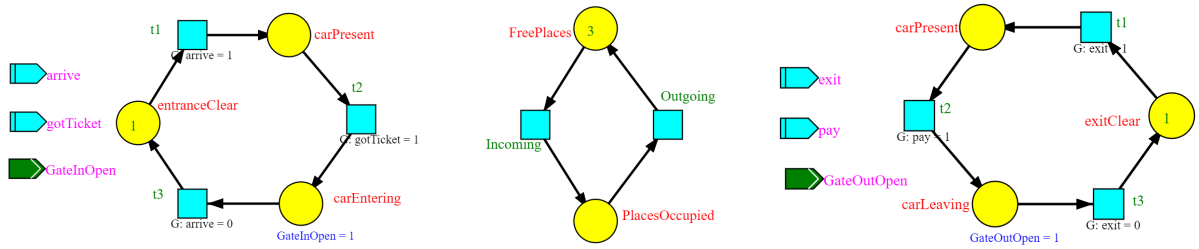
From the above, depending on the specific goals included on specific courses where students are enrolled, different types of Petri nets classes can be considered, including autonomous classes, such as elementary nets [2], place-transition nets [3], and coloured Petri nets [4], as well as non-autonomous classes, such as interpreted and synchronized Petri nets [5] [6] [7].

In this last group, the Input-Output Place-Transition nets (IOPT-nets) [8] [9] [10], and their associated tools framework (IOPT-Tools) [11], are of special interest. IOPT nets and IOPT-Tools allow explicit modeling of inputs and output signals dependencies, as well as automatic generation of execution code, which can be directly deployed in some popular boards, such as Arduino, Raspberry Pi, and FPGA-based (using C and VHDL code).

In the following sections, without loss of adequacy for the usage of the other referred classes, IOPT-nets will be used, illustrating possible lines of conduct.

4. Practicing basic modeling strategies

Coming back to the simplest parking lot configuration illustrated in Figure 1, one possible approach to follow may rely on identification of sub-models associated with the different parts of the system. In this case, one may build separated models associated with the entrance, the parking area, and the exit, as illustrated in Figure 2, where explicit reference on input signals dependencies as well as output actuators are included.



(a) Entrance. (b) Parking Area. (c) Exit.

Figure 2: Sub-models to be used in compositions.

The modeling strategy may initially adopt a kind of simplified (or naif) approach, where when a car comes to the entrance/exit area will eventually enter/leave (as illustrated in Figure 2(a) and (c)).

Staying with the simpler parking lot configuration (one entrance and one exit), the next step will compose the models producing the parking lot controller model, relying on the net addition operation between models [12] [13]. For the case at hand, a synchronous composition is selected, where fusion of groups of transitions as illustrated in Figure 3 is used, allowing reaching the model of Figure 4.

Further complications may consider different drivers' behavior where after being present at the entrance/exit, the driver goes reverse and leave the entrance/exit area (even before gate opening). In this line of complications, one additional ground detector can also be considered to be installed after the gate in order to confirm that the car actually cross the gate, or goes reverse not crossing the gate (as well as to avoid closing the gate while the car is entering). In this case, sub-models need to be

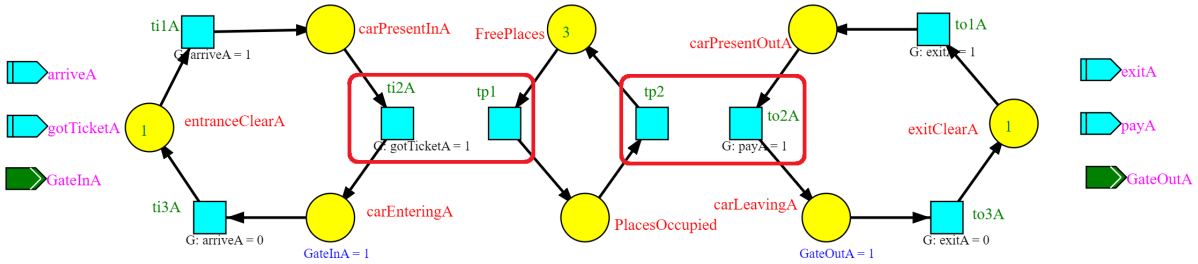


Figure 3: Synchronous composition of the sub-models for a parking lot with 1 entrance and 1 exit, identifying the fusion sets to be used.

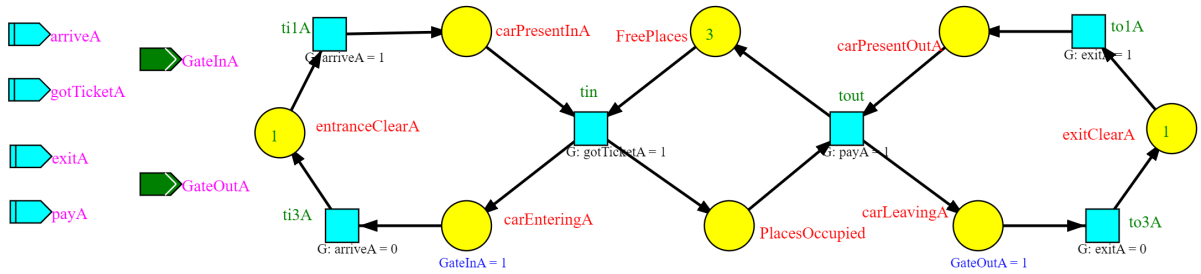


Figure 4: The parking lot controller model.

more elaborated, as instead of considering an atomic action associated with the fusion of transitions at entrance and at the parking area, it will be necessary to consider a two-phase operation, where the first phase checks availability of a free place to park and reserve the place (removing a token from the *FreePlaces* place), but only create a token during the second phase in the *PlacesOccupied* place after actually crosses the gate afterwards. If driver goes reverse, the token is regenerated at *FreePlaces* place.

Going further with composition of sub-models, one may consider to build the model of a parking lot with two entrances and two exits, considering the disjoint addition of two entrance models, two exit models, and one parking area model, and four fusion sets of transitions to be merged (synchronous composition), as presented in Figure 5, resulting in the model of Figure 6.

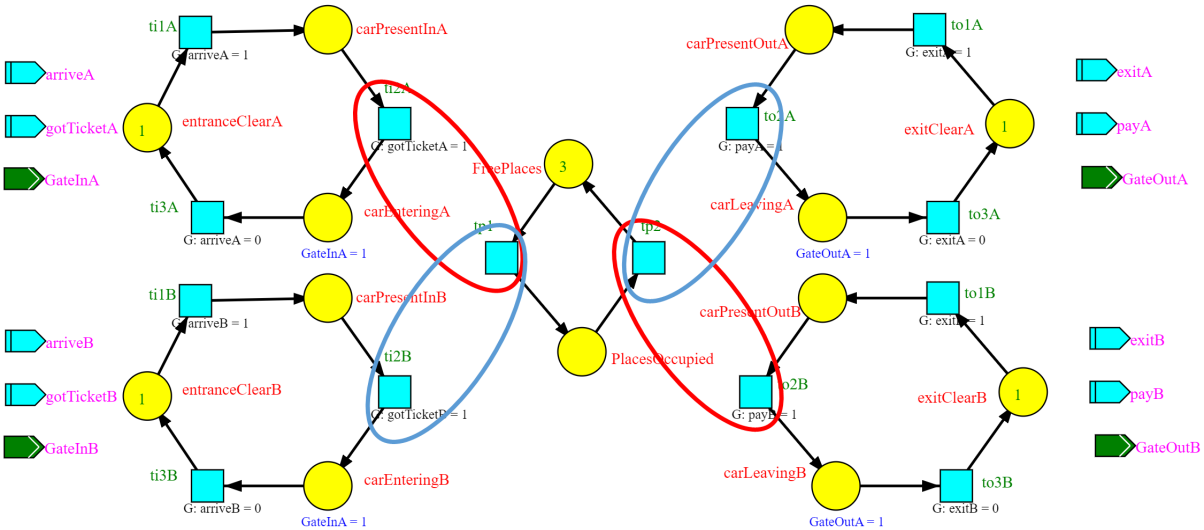


Figure 5: Synchronous composition of the sub-models for a parking lot with 2 entrances and 2 exits, identifying the fusion sets to be used.

Other complication that can be considered for large parking lot infrastructures is the existence of different floors or parking areas, while continue to monitoring the movement of vehicles between

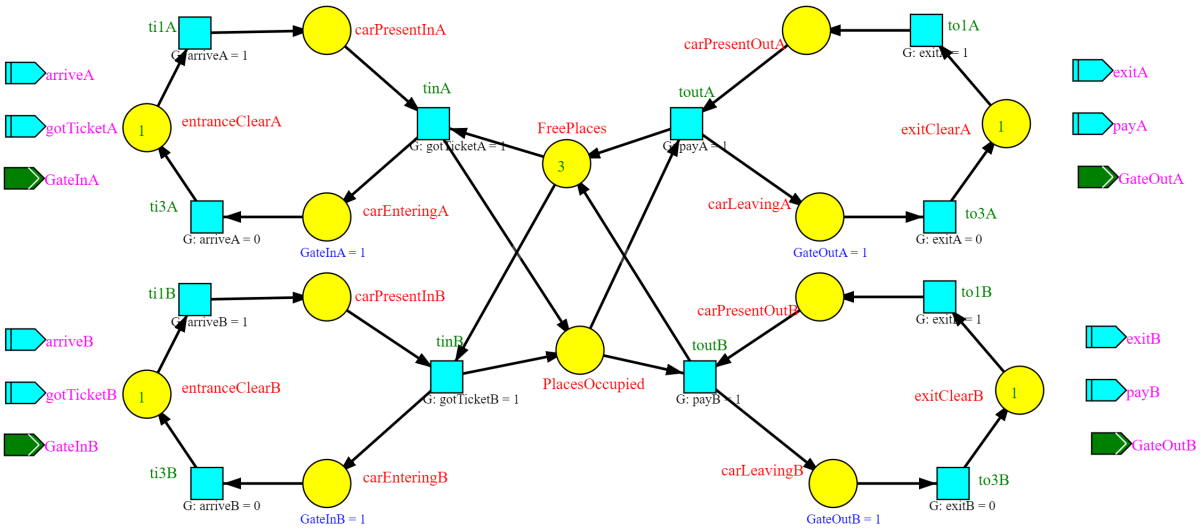


Figure 6: The parking lot controller model considering 2 entrances and 2 exits.

those areas in order to provide information to the parking lot's users where they can find places to park. Considering different parking areas or different floors at the parking lot, one can replicate the model of Figure 2(b) for each area (while keeping another replica of Figure 2(b) to control the overall number of free and occupied places). In order to detect the movement of cars between those different parking zones it will be necessary to consider a model depending on some presence sensors, such the one presented at Figure 7, which can be (asynchronously) composed with the remaining models (in this case, each parking zone will have an instance of the parking area of Figure 2(b), and the composition of the model of Figure 7 will consider merging the places at the left hand side of Figure 7 with the places of the instance of Figure 2(b) of Zone A, and merging the places at the right hand side of Figure 7 with the places of the instance of Figure 2(b) of Zone B). Potential complications to be considered include unidirectional passages or bidirectional passages, allowing vehicles to go or not in reverse while in the transit area.

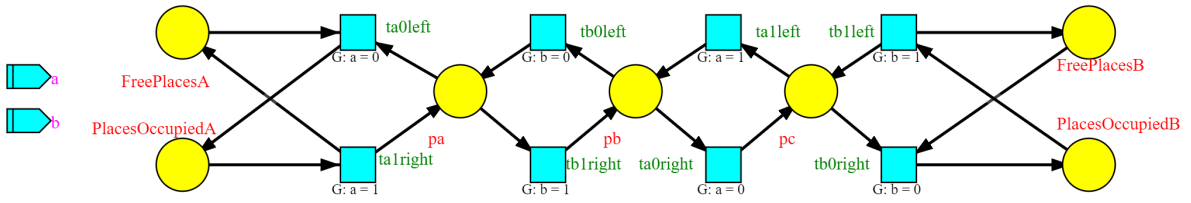


Figure 7: Modeling moves between different parking zones.

Coming back to the model of Figure 6, some effective conflicts may occur. In particular when only one place is free inside the parking lot and two cars are present at both entrances willing to enter, two transitions at the entrances are in conflict. This situation can be used to introduce the concept of arbiters, responsible to a-priori manage the conflict (for some types of controllers, where a non-deterministic behavior is not allowed, this type of arbiters may be necessary to consider).

Figure 8 presents two possible types of arbiters to a-priori fix conflicts, where the transitions at the top and at the bottom of the models will be fused with the in-conflict transitions of the entrances. Important to refer that using the model on the left of Figure 8 will still introduce a conflict (between one of the two transitions of the arbiter and one of the two transitions initially in-conflict), which can be fixed introducing priorities among the in-conflict transitions (giving low priority to the transitions of the arbiter). This can be used to introduce the concept of priorities associated with transitions.

On the other hand, if the right hand model of Figure 8 is used instead, no priorities are necessary to

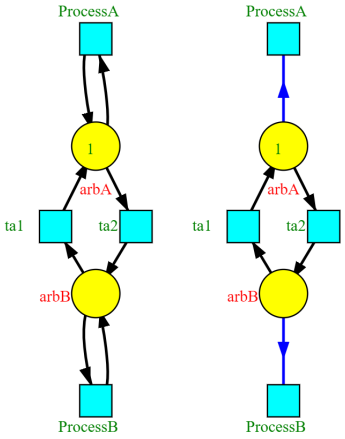


Figure 8: A-priori fixing conflicts using specific arbiters.

be introduced, but it will be necessary to consider test arcs, resulting in the model of Figure 9.

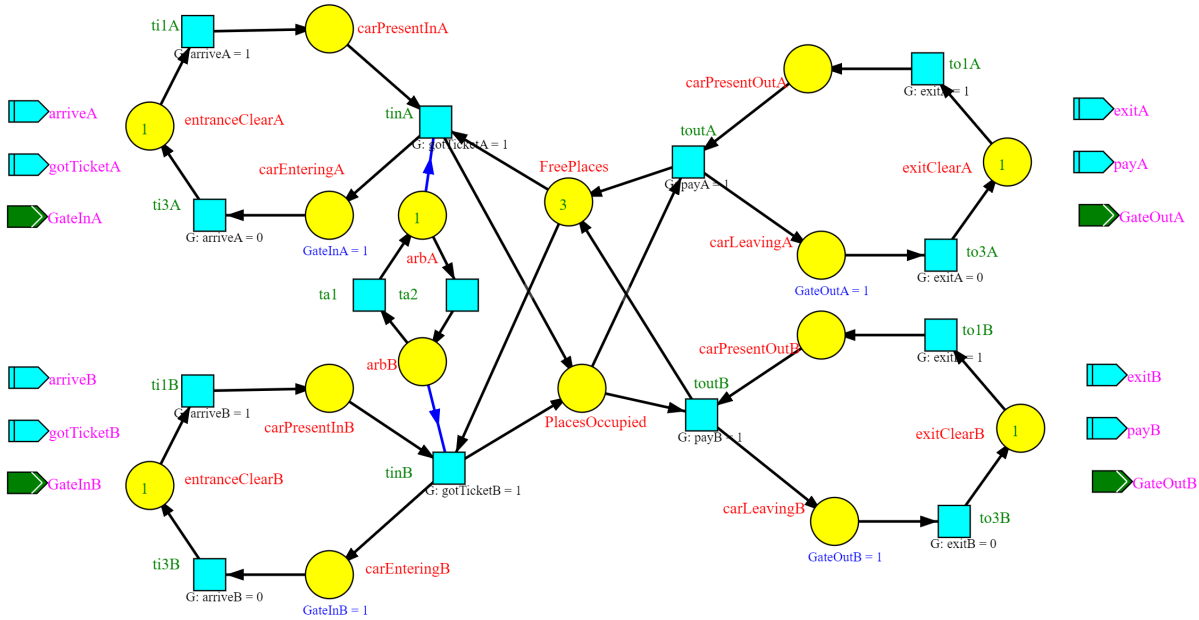


Figure 9: The parking lot controller model considering 2 entrances and 2 exits including a arbiter for conflict management.

5. Brief discussion

Other than the modeling activities referred in the previous sections, where the students were challenged to build the behavioral model of a parking lot controller taking advantage of composability of Petri net models, from the pedagogical point of view, other aspects important in controllers' development are emphasized.

One of them is related with the benefits of adoption of a model-driven development approach, where the system model can be used during all phases of development, ranging from specification to implementation, including simulation and verification of properties.

For that end, a set of web-based computational tools, the IOPT-Tools framework [11], freely available at http://gres.uninova.pt/IOPT-Tools/, is used to support the whole phases of development. IOPT-Tools have been in use at several universities supporting experimentation in several engineering courses

[14, 15].

Some of the tools available within the IOPT-Tools provide support for simulation of the model, where students can validate their solutions. The simulator tool uses an interactive token-player interface, as well as a timing diagram (similar to those obtained with data analysers), which provides a visual representation of the evolution of the marking and transition firing, as well as of the input and output signals of the controller. In this sense, students can analyse the behavior of the controller from two different points of view, namely from a model-centric view, or from the input-output interface view.

The validation through the simulation of the model is complemented with a set of tools allowing property verification through the construction of the reachability graph representing the associated state space of the model and a querying system providing answers to specific questions on boundedness and reachability aspects, among others.

Also the physical implementation of the controllers is addressed as the IOPT-Tools provides automatic code generators for software-centric implementations based on C, as well as for hardware-based implementations based on VHDL. Generated code can be directly deployed (without writing additional lines of code) in popular low-cost boards used in the lab such as Arduino, ESP-32, LattePanda, Raspberry Pi and other Linux platforms, as well as on several FPGA boards from AMD-Xilinx. Current practice at our University is to loan a FPGA board to students to allow extensive and autonomously experimentation by them on the whole phases of development.

Overall, students enjoy having a web-based tool for development and a board for autonomous experimentation always available and are motivated to participate actively in the lab activities.

6. Conclusions and future work

In this paper, a non-autonomous Petri net class based on Place-Transition nets and including priorities and test arcs was used (the class of IOPT-nets). However, as referred in Section 3, several classes of Petri nets can be equally used to explore exercises in this challenge framework.

As a matter of fact, most of the goals (namely introducing concurrency, locality on evaluation, conflict modeling, and so on) can also be achieved with autonomous Petri nets, namely Place-Transition nets, or other types of autonomous classes. Introducing high-level characteristics on the net, such as in Coloured Petri nets, will allow introducing new challenges, as different types of vehicles.

Also a final challenge (not presented in this paper) can address distributed execution of the global model, relying on the decomposition of the model into a set of sub-models and allowing their execution by a set of distributed controllers (one per entrance and exit, for instance), where the interaction between the different sub-controllers is achieved asynchronously (and overall the whole system can be seen as a Globally-Asynchronous Locally-Synchronous system).

Acknowledgments

This work was financed by Portuguese Agency FCT – Fundação para a Ciência e Tecnologia, in the framework of project UIDB/00066/2020.

References

[1] R. Sinha, S. Patil, L. Gomes, V. Vyatkin, A survey of static formal methods for building dependable industrial automation systems, IEEE Transactions on Industrial Informatics 15 (2019) 3772–3783. doi:10.1109/TII.2019.2908665.

[2] G. Rozenberg, J. Engelfriet, Elementary net systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 12–121. URL: https://doi.org/10.1007/3-540-65306-6_14. doi:10.1007/3-540-65306-6_14.

[3] J. Desel, W. Reisig, Place/transition Petri Nets, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 122–173. URL: https://doi.org/10.1007/3-540-65306-6_15. doi:10.1007/3-540-65306-6_15.

[4] K. Jensen, L. Kristensen, Coloured Petri Nets: Modelling and Validation of Concurrent Systems, 2009. doi:10.1007/b95112.

[5] M. Silva, Las Redes de Petri : en la Automática y la Informática;1ª ed., Madrid : Editorial AC, D.L., 1985.

[6] R. David, H. Alla, Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems, Prentice-Hall, Inc., USA, 1992.

[7] R. David, H. Alla, Discrete, Continuous, and Hybrid Petri Nets, 2nd ed., Springer Publishing Company, Incorporated, 2010.

[8] L. Gomes, J. P. Barros, A. Costa, R. Nunes, The Input-Output Place-Transition Petri net class and associated tools, in: 2007 5th IEEE International Conference on Industrial Informatics, volume 1, 2007, pp. 509–514. doi:10.1109/INDIN.2007.4384809.

[9] L. Gomes, F. Moutinho, F. Pereira, J. Ribeiro, A. Costa, J.-P. Barros, Extending Input-Output Place-Transition Petri nets for distributed controller systems development, in: ICMC'2014 - International Conference on Mechatronics and Control, 2014, pp. 1099–1104. doi:10.1109/ICMC.2014.7231723.

[10] L. Gomes, J. P. Barros, Refining IOPT Petri nets class for embedded system controller modeling, in: IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, 2018, pp. 4720–4725. doi:10.1109/IECON.2018.8592921.

[11] F. Pereira, F. Moutinho, A. Costa, J.-P. Barros, R. Campos-Rebelo, L. Gomes, IOPT-Tools – from executable models to automatic code generation for embedded controllers development, in: L. Bernardinello, L. Petrucci (Eds.), Application and Theory of Petri Nets and Concurrency, Springer International Publishing, Cham, 2022, pp. 127–138. doi:https://doi.org/10.1007/978-3-031-06653-5_7.

[12] J. Barros, L. Gomes, Net model composition and modification by net operations: a pragmatic approach, in: 2nd IEEE International Conference on Industrial Informatics, 2004. INDIN '04. 2004, 2004, pp. 309–314. doi:10.1109/INDIN.2004.1417350.

[13] L. Gomes, J. P. Barros, Structuring and composability issues in Petri nets modeling, IEEE Transactions on Industrial Informatics 1 (2005) 112–123. doi:10.1109/TII.2005.844433.

[14] G. Bazydło, A. Costa, L. Gomes, Integrating different modelling formalisms supporting co-design development of controllers for cyber-physical systems – a case study, in: ICELIE'2022 – 9th International Conference on E-Learning in Industrial Electronics; October 17-20, 2022, Brussels, Belgium, 2022. doi:10.1109/ICELIE55228.2022.9969438.

[15] L. Gomes, A. Costa, Model-driven development in hardware-software co-design of controllers for cyber-physical systems, in: ICELIE'2023 – 10th International Conference on E-Learning in Industrial Electronics; October 16-19, 2023, Singapore, 2023. doi:10.1109/ICELIE58531.2023.10313106.