Cross Modeling of Embedded Systems Using SysML and Petri Nets

Wojciech Szmuc and Tomasz Szmuc

***Abstract*—Cross modeling in embedded systems development is proposed in the paper. The main idea consists in a translation of SysML artifacts into the related Coloured Petri Net (CPN) models, which may be verified directly or using other tools, e.g. Temporal Logic Provers. The paper is an extension of [20] mainly by insertion of sequence diagrams (SysML) and presentation of their mapping into CPN models. The additional part describes communication features and completes the cross modeling approach.**

***Index Terms*—embedded systems; cross modeling; formal verification**

I. Introduction

FAST grow of embedded world has been observed in the last decade. The grow is accelerated by development of microcontroller technology, construction of new sensors and IoT technology. This development extends the area of applications – from rather simple controllers to complicated systems, e.g. variety of robot microcontrollers, avionics controllers, autonomous (intelligent) car subsystems, biomedical applications, etc. Development of the complicated systems should be supported by systematic modeling and attentive analysis of developed artifacts. The requirements become strong obligation in development of Safety Critical Systems [1, 2, 6, 10, 11, 12].

Development of embedded systems is involved both in hardware and software domains. Modeling languages are different in these fields, therefore other languages and methodologies merging the two fields were developed. The languages and the development processes are embedded in system engineering domain. The most popular modeling languages in this area are SysML [5, 8, 16] and AADL [7]. SysML (Systems Modeling Language) was chosen as modeling language in the proposed approach. The language is an extension of UML profile and was developed under OMG (Object Management Group) umbrella. Range of application of SysML is significantly wider and moreover several commercial and free (noncommercial) modeling tools are available. AADL (Architecture Analysis and Design Language) is also an interesting proposal, but less popular and used mainly in avionic and military industry.

Increasing complexity of embedded systems and growing number of safety critical applications imply needs for systematic development methodologies supported by formal methods for analysis and verification during the development. The aim of the paper is to present translations from SysML into Coloured Petri Nets (CPN) [13] models. The translated model is described using one (CPN) language, which provides several possibilities for checking (verification) of system properties.

II. The Proposed Concept

SysML offers tools for integration of several modeling views described by different languages. The modeling tools are grouped in the free categories: requirements diagrams, behavior diagrams and structure diagrams. In the paper we shall focus on structure and behavior diagrams – the requirements diagrams are used for structuring and displaying text-based requirements and will be not considered in the paper, since we mostly interest in modeling and verification of logical/operational perspectives. Sequence diagrams introduced here completes the picture presented in [20] by description of communication aspects.

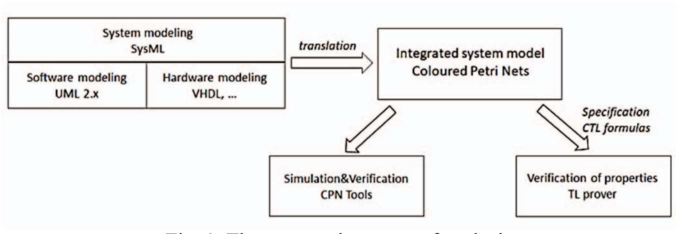

Fig. 1. The proposed concept of analysis

The integration of different description languages (diagrams) is convenient for visualization of system from different points of view (perspectives). The variety of description languages may cause problems when an analysis of integrated models (or part of them) is considered. In such cases it would be better to have models described using one language. In the proposed approach (Fig. 1) SysML artifacts are translated into Coloured Petri Net models (arrow labeled by *translation*). The models may be analyzed using CPN Tools [4] or used as an input to Temporal Logic prover. The later case needs additional effort, i.e. definition of system properties which should be satisfied in the CPN model. Sometimes input files of provers need special format, but it is easy to rewrite because CPN models are specified in clear XML format, and some of them specify the data format (see [4, 15] as examples).

W. Szmuc and T. Szmuc are with the Department of Applied Computer Science, AGH University of Science and Technology, Kraków, Poland (e-mails: wszmuc@agh.edu.pl, tsz@agh.edu.pl).

The research was supported by the Department Research Grant No 11.11.120.859: "Methods, tools and environments for knowledge and software engineering". Topic: "Modeling, analysis and optimization of selected systems".

The paper is a continuation of the research towards efficient use of formal methods to support development of embedded systems. Translation from UML 2.0 into CPN models is presented in [18, 19]. SysML is UML subset with some extensions towards system modeling including hardware components. Experience gathered in UML→CPN translation, including its implementation has been used and some results have been directly or after some modifications moved to the SysML translation. Block notion is a new and main concept in SysML. Blocks are core units for building system structure. The paper is focused on translation of SysML blocks, modified activity diagrams, and sequence diagrams. Translations of the first two diagrams (presented in [20]) describe structure and behavioral perspectives. Mapping of sequence diagrams completes the picture by adding communication aspects.

III. SysML – an overview

SysML set of diagrams consists of three main categories: behavior diagrams, requirement diagrams, and structure diagrams (Fig. 2).

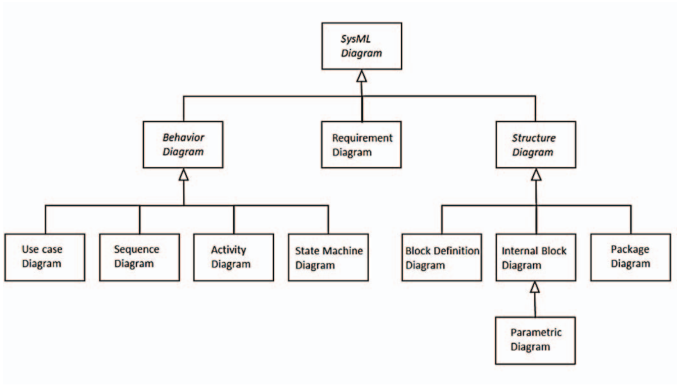

Fig. 2. SysML diagrams [5]

Use case diagrams, sequence diagrams, state machine diagrams and package diagrams are directly taken from UML 2.5. Activity diagrams are inherited with some modifications. Similarly block definition diagrams and internal block diagrams may be treated as being derived from class/object diagrams. It is not enough space in the paper to present all the diagrams, therefore we shall restrict to brief description of the "new" diagrams.

The **requirements diagrams** (RD) are used to define relationships between requirements and/or related use cases, blocks, etc. They are used for structuring textual requirements using several dependency relations: containment, trace, derive requirement, refine, satisfy, and verify.

The **block definition diagrams** (BDD) are used to specify in blocks, actors, value type, constraint blocks, flow specifications, and interfaces form types for other elements appearing in other SysML diagrams.

The **internal block diagrams** (IBD) define internal structure of the related blocks. Any IBD describes in which way parts of a block must be connected to create an instance of the block.

The **parametric diagrams** (PR) are used to specify relationships between *blocks* and *constraint blocks*. Constraint blocks are used to close inside frame constraints, i.e. bindings between parameters expressed by equations and mathematical relationships.

IV. Translation from SysML into CPN

In this paragraph the chosen SysML constructs and the corresponding CPN models are presented. Algorithm of automatic translation is complicated and its description is out of the paper scope. Translation of activity diagrams is the only one example, where relations between the related elements are simple (Fig. 3).

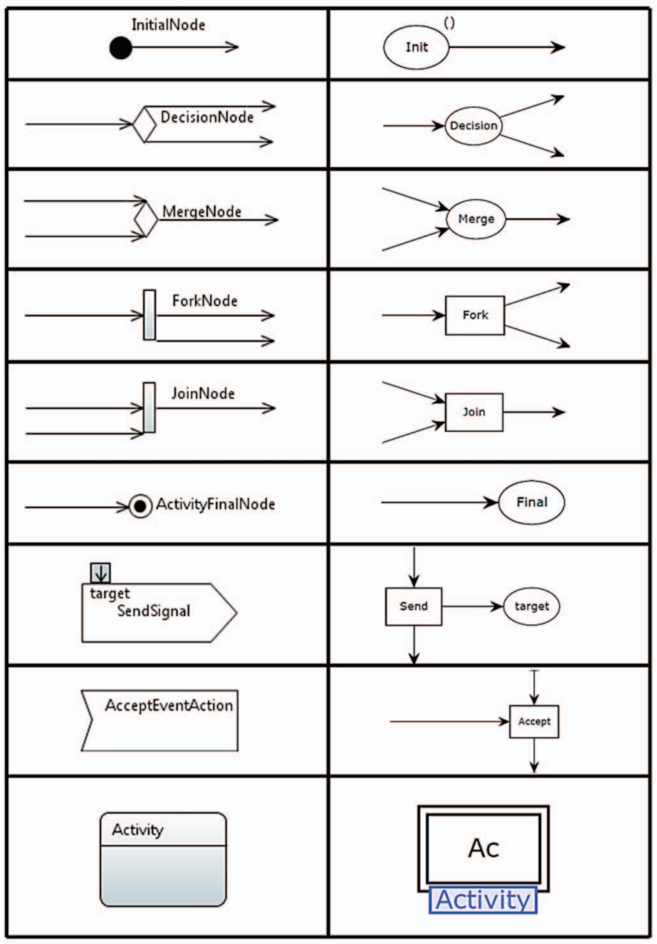

Fig. 3. Activity diagram symbols and the related CPN constructs

General rules for translation from activity diagrams into CPN constructs are illustrated in Fig. 3, where activity diagram elements are presented on the left hand side, and the relevant CPN constructs are located in the right part.

The main and most important rules are explained and illustrated on ATM example. We hope, that they will form a basis allowing understanding of the proposed concept. Any model of system consists of several artifacts grouped in the two main perspectives describing the system structure and its behavior. Sequence diagrams specify communication between related elements (blocks, actors, etc.) and belong to behavioral class, but are important for specification and analyses of communication properties. Internal block diagrams, activity diagrams and sequence diagrams have been chosen in the paper as representatives of the related perspectives. The diagrams describe main views of system and also IBD is a new SysML construct. Translation of IBD diagrams leads to homomorphic structure, which will be visible at the ATM example.

A. Internal block diagram and the related CPN model

To depict practical use let us consider an example of internal block diagram describing (using SysML) structure of simplified ATM model (Fig. 4). Behavior of the Controller is described by activity diagram in Fig. 6. The definitions of ports (Fig. 4) permit to determine the right connection of send and receive constructs, which in turn is applied in conversion of activity diagram fig. 7.

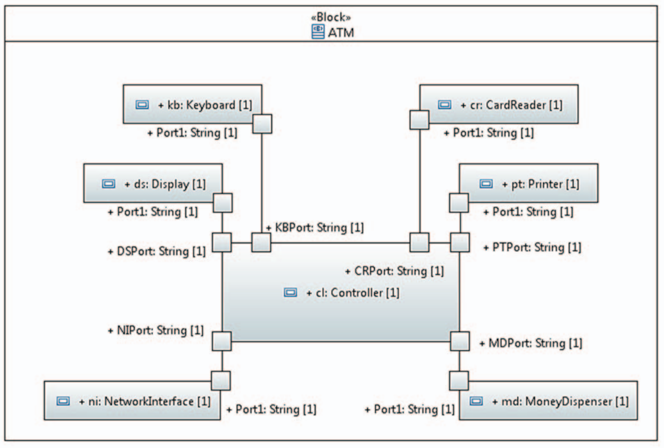

Fig. 4. IBD describing structure of ATM

During translation the IBD is used to identify data types in message exchange and distribution of functionality among the system. Result of the translation is presented in Fig. 5.

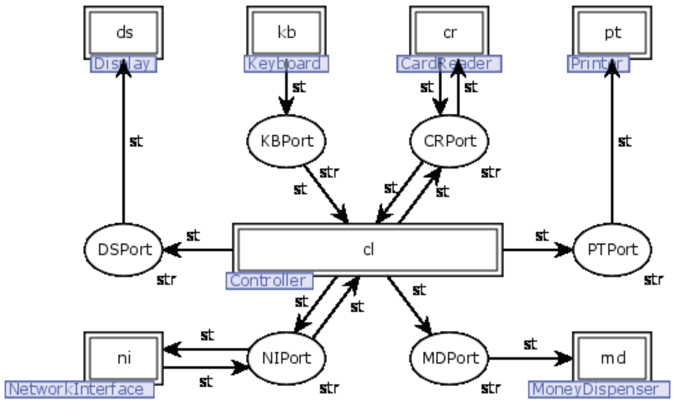

Fig. 5. CPN model after translation of IBD (Fig.4)

The structure is expressed using hierarchical CPN, where the upper level (Fig. 5) consists of substituted transitions and ports, being gates to lower level.

Detailed description of the related subsystems is located on the corresponding subpages (lower level CPNs). Behavior of the "Controller" (the corresponding subpage) is specified in Fig. 7.

B. From activity diagram into CPN model

Activity diagram describing behavior of ATM is presented in Fig. 6. The diagram is self-explaining, therefore more detailed comments seem to be not necessary.

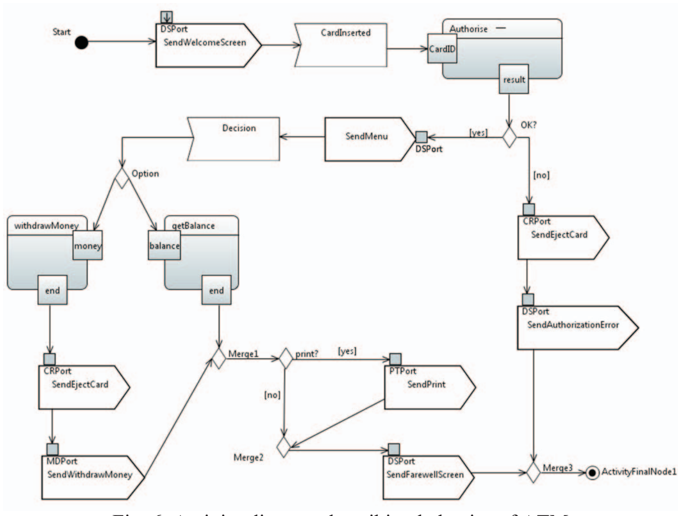

Fig. 6. Activity diagram describing behavior of ATM

Fig. 7 presents the translation of the activity diagram (Fig. 6) into CPN. The layout of the elements has been organized (to the possible extent) to reflect the input arrangement. The general rules of conversion have been presented in Fig. 3, so the following description will focus on more specific aspects.

The first is the use of fusion of places employed to make the net easier to understand – no cross connections. However, since the fusion is assigned to port places there is a need to add another construct (it is not possible in CPN to define a port place being also of fusion type). The solution is quite simple and it has not been presented in the figure. It consist in adding another place (the real port) which is connected via transition to place in proper fusion (e.g. DSPort).

The second important issue is the declaration of types (colsets). Two types were used in resulted CPN – one is "string" and another is "record" of two strings. The code (only crucial constructs) is as follows:

```
colset str=string;

colset tran=record amount:str*pr:str;
```

Last line describes a type implemented to store the value received in an earlier stage. The construct results from the selection if the ATM user wants to print the summary of the operations. This takes place in either withdrawMoney or checkBalance operations (Fig. 6).

The last element that would need an additional comment is the transition named "PrintDecision". Its existence is the consequence of easiness of backward translation – the result of CPN analysis is to detect possible problems in design therefore, the constructs had to be defined separately.

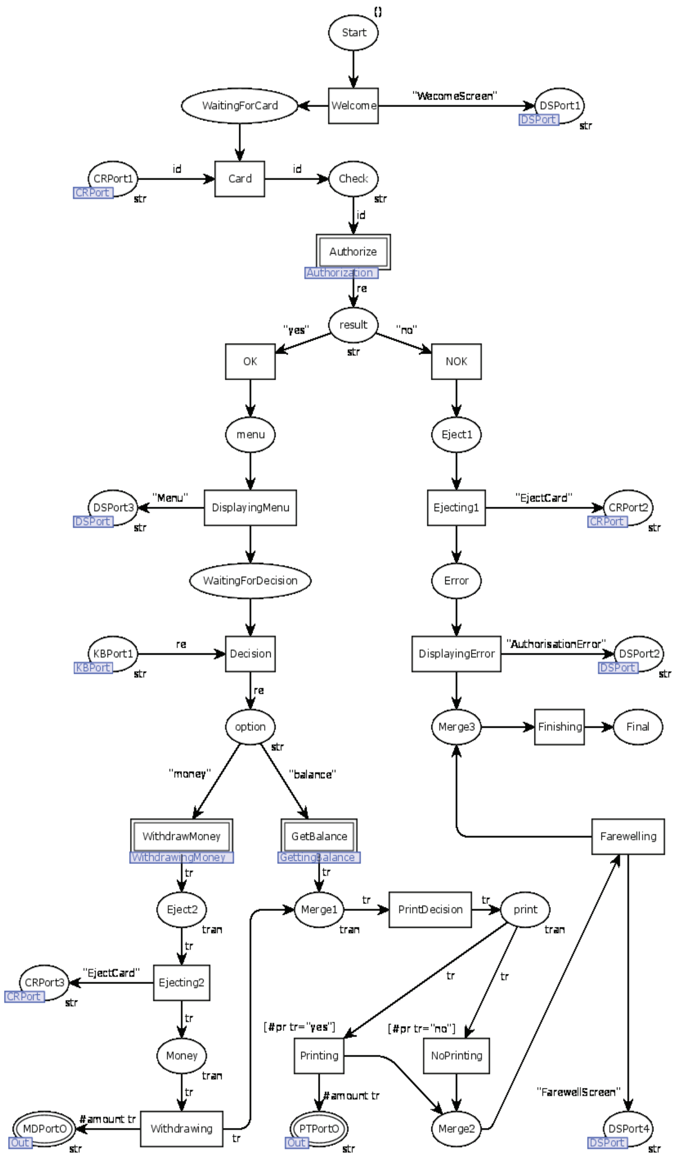

Fig. 7. CPN model after translation from activity diagram (Fig. 6)

C. Translation from sequence diagrams into CPN models

The authorization (Fig. 8) is modeled using sequence diagram. Four lifelines are involved in the scenario: keyboard, display, controller and network interface. Since the card id was already received, the next step is to ask the user to introduce PIN which is depicted as message "EnterPIN" from controller to display. If PIN is entered, then it is sent from keyboard to the controller. In the next step, the collected data are sent to network interface in order to verify their correctness. The answer is sent back to the controller. The next step may break the loop, if the data are correct. Otherwise, the message "WrongPIN" is sent. If the number of attempts to validation is less than 3 the next iteration is proceeded.

Fig. 9 presents translation of SysML sequence diagram (Fig. 8) into the corresponding CPN model. Starting from the top there are 4 places, each for one head of lifeline: *kb1* for keyboard, *ds1* for display, *cl1* for controller and *ni1* for network interface. The next element is transition which then traces token to the place that begins the loop construct.

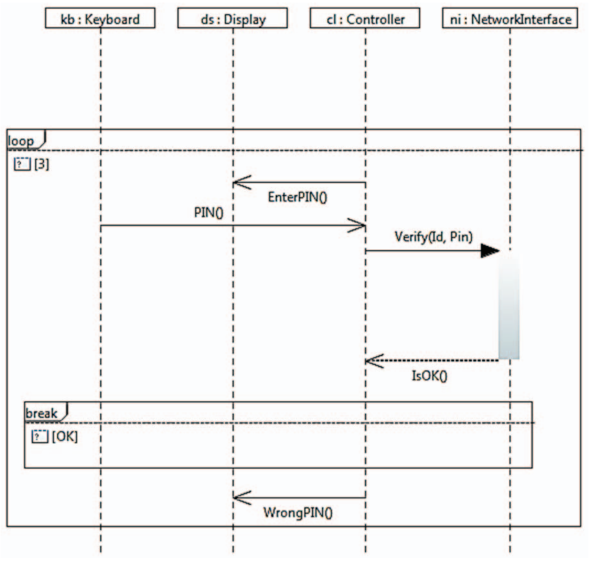

Fig. 8. Sequence diagram of authorization operation

The first action in the loop is message arrival from controller to display. It is implemented with transition *sEnterPIN* which sends token ("EnterPIN") to port (*DSPort5*). The transition *rEnterPIN* is now enabled and after firing puts token in place *ds12*. The consequent transition allows waiting for execution and passes token to *kb12*. Transition *sPIN* sends message to *rPIN* (through *KBPort2*). PIN saved in the variable *st* is than concatenated with value stored in *stt*. In the next step, the token through *cl13* is taken by *sVerify* and sent with "v" added at the beginning to *NIPort1* (bidirectional port). Tokens may be gathered from that place only by network interface (via *rVerify*) because of the guard (the first character has to be "v"). After the verification (*verify*(*st*)) the answer is sent back to controller (*sIsOK*, *NIPort1* and *rIsOK*). The answer is put in result place in the next step. In order to execute either *OK* or *NOK* transition, the place *break* must contain 3 tokens. They should be provided by *kbBreak*, *dsBreak* and *niBreak*. There is no need to wait for *clBreak* since it puts token in *result* place. If PIN is correct ("ys"), 4 tokens are sent to enable exit transition for each lifeline (*kbExit*, *dsExit*, *clExit*, *niExit*). The transitions put tokens in the last places of each lifeline. In the case of wrong PIN 5 tokens are inserted in *continue* place. Four of them are used to continue lifelines execution (transitions with *continue*) and the fifth one is used to increase counter of loop iterations. As it is depicted in Fig. 9 the message "WrongPIN" need to be sent which is implemented with part containing: *sWrongPIN*, *DSPort6*, *rWrongPIN*. After that step the loop condition is checked. Transition *check* takes token from iteration counter and puts in place *checking*. Two possibilities are available now. One is the end of the loop, when the number of iterations is greater than 2. In that case 4 tokens are placed in *ending* and thus appropriate transitions are enabled. The other possibility is the next iteration of the loop. The transition *again* returns token to the instance counter and also puts 4 tokens in *onceAgain*. That enables transitions putting back tokens in places representing the beginning of the loop for each lifeline (*kb11*, *ds11*, *cl11* and *ni11*). The token for controller is also trimmed to contain only *CardID* data.

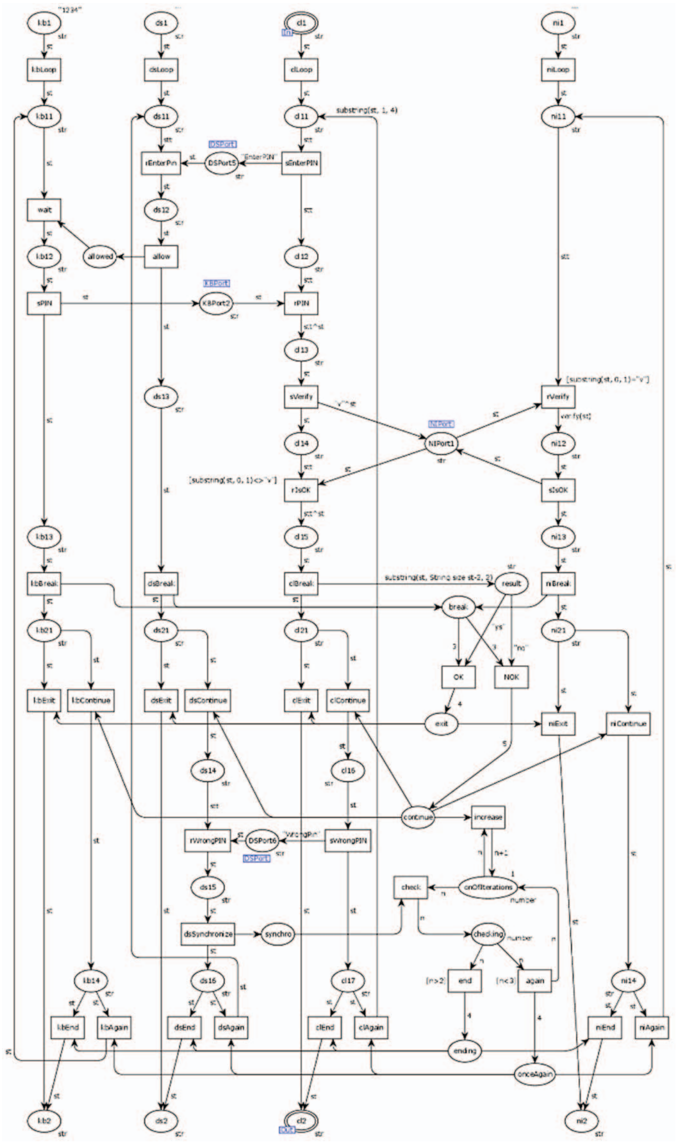
Fig. 9. CPN model of authorization specified in Fig. 8

The presented authorization model contains loop in the sequence diagram (Fig. 8). The loop is distributed among 4 life lines what makes the related Petri Net more complicated. The above detailed description explains in which way such construct may be modeled using CPN.

V. Summary

The paper describes concept of translation from SysML selected diagrams into the related CPN models. The three SysML diagrams: internal block definition, activity and sequence diagrams representing structural and behavioral (including communication) perspectives have been chosen, which in turn provide formal verification of designed system. The concept has been illustrated by translation of the diagrams modeling ATM model. Translation of the other SysML diagrams may be carried out using the rules and algorithms defined in [18]. In particular UML state (machine) diagrams being the second main tool for modeling of software behavior have been directly moved to SysML [5, 8, 16]. The same diagrams are used in SysML for specification of hardware part.

The resulted CPN model is an integrated model of the system described using one formal language. The language provides several possibilities for proving properties of the system model. The two main methods are presented in Fig. 1, i.e. direct application of CPN tool, or using the CPN as a model for verification if Temporal Logic (TL) formulas describing required properties are satisfied. The two approaches are briefly presented below.

CPN model is a net (bipartite graph with marking) which by its execution (simulation) generate graph of states (markings). The graph specifies all states which are reachable from the initial state. The CPN tool [4] enables generation of the tree and checking of several its properties specified by predicates over the states. The following main properties may be proved [13].

- Reachability of states, i.e. if given state may be reached from the initial one;
- Home property, i.e. if given state may be reached from any reachable state, e.g. if state when controller is ready for servicing is always reached in some future;
- Classical properties in concurrency: safety, liveness and fairness defined by a composition of reachability properties.

The tool enables also modeling and analysis of time properties – see [17] for details.

The second approach is based on proving of TL properties in the model which is defined by the tree covering reachability graph generated by CPN. The required properties are specified using mainly Computational Tree Logic (CTL). Many CTL logic provers exist [3, 9, 14, 21] and efficiency of proving have been remarkably improved for last years, extending scope of verified models to industrial system dimension.

References

[1] J.-P. Blanquart, J.-M. Astruc, P. Baufreton, J.-L. Boulanger, H. Delseny, J. Gassino, et al., “Criticality categories across safety standards in different domains “, ERTS2 Congress. Embedded Real Time Software and Systems, 2012, pp. 3–4.

[2] J. Boercsoek, JM. Schwarz, E. Ugljesa, P. Holub, and A. Hayek, “High-availability controller concept for steering systems: The degradable safety controller”, Recent Researches in Circuits, Systems, Communications and Computers, WSEAS. 2011, pp. 222–228.

[3] B. Cook, H. Khaaaf, and N. Piterman “Verifying increasingly expressive Temporal Logics for infinite-state systems”, Journal of the ACM, vol. 64, No 2, May 2017, Article 15.

[4] CPN web: http://cpntools.org/

[5] L. Delligatti, SysML Distilled. A Brief Guide to the Systems Modelling Language, Addison-Wesley, 2014.

[6] DO-178C, Software Considerations in Airborne Systems and Equipment Certification

[7] P.H. Feiler, and D.P. Gluch, Model-Based Engineering using AADL. An Introduction to the SAE Architecture Analysis & Design Language”, Addison-Wesley, 2012.

[8] S. Friedental, A. Moore, and R. Steiner, “A Practical Guide to SysML. The System Modeling Language”, Morgan Kaufman OMG Press, 2010.

[9] R. Gore, J. Thomson, and F. Widmann, "An experimental comparison of theorem provers for CT, Proceedings of the 2011 International Symposium on Temporal Representation and Reasoning, IEEE Computer Society, 2011, pp. 49-56.

[10] K. Greb, A. Seely, Design of Microcontrollers for Safety Critical Operation (ISO 26262 Key Differences from IEC 61508), 2009.

[11] IEC 61508 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES).

[12] Road vehicles – Functional safety, ISO 26262.

[13] K. Jensen, and L. M. Kristensen, Coloured Petri Nets. Modelling and Valiation of Concurrent Systems, Springer, 2009.

[14] NuSMV, a new symbolic model checker: http://nusmv.fbk.eu/

[15] Papirus, https://www.isis-papyrus.com/software

[16] D. Rosenberg, and S. Mancarella: "Embedded systems development using SysML", Sparx Systems Pty Ltd and ICONIX, 2010

[17] S. Samolej, and T. Szmuc, "Time constrains modeling and verification using Timed Coloured Petri Nets, Proceedings of the 28th IFAC/IFIP Workshop on the Real Time Programming, Elsevier Science Ltd, 2005, pp. 127-132.

[18] W. Szmuc, Modelling of Selected UML 2.0 Diagrams with Coloured Petri Nets. PhD Report, Supervisor M. Szpyrka, AGH 2014

[19] W. Szmuc, and T. Szmuc, "Modeling UML object event handling with Petri Nets. Towards improvement of embedded systems analysis and design", 23rd International Conference on "Mixed Design and Integrated Circuits and Systems, MIXDES 2016, June 2-25, 2016, Łódź, Poland, pp. 454-457.

[20] W. Szmuc, and T. Szmuc, "Towards Embedded Systems Formal Verification. Translation from SysML into Petri Nets", Proceedings of the 25th International Conference "Mixed Design of Integrated Circuits and Systems", MIXDES 2018, June 21-23, 2018, Gdynia, Poland, pp.422-425.

[21] Verics model checker: http://verics.ipipan.waw.pl/

Wojciech Szmuc received MSc in Automatics and Robotics (Computer Science in Control and Management) from the AGH University of Science and Technology (AGH) in 2001. Employed since 2006 at the AGH University of Science and Technology, where received Ph.D. (2015) in Computer Science. The research focuses on formal methods (Petri Nets, Temporal Logics, Process Algebras) in application of Software Engineering modelling and support of software development. Although, the main concern are real-time systems and embedded systems some research was made also in Digital Watermarking and Steganography. This work resulted in achieving US patent as one of co-authors.

Tomasz Szmuc received MSc in Electrical and Control Engineering from the AGH University of Science and Technology (AGH) in 1972. Employed since the beginning at the AGH University of Science and Technology, where received Ph.D. (1979) and Sc.D. (1989) degrees (both in Computer Science), and Professor title (1999). The research focuses on Software Engineering, in particular applications of formal methods (Petri Nets, Temporal Logics, Process Algebras) for modelling and support of software development. Development of real-time systems and embedded systems constitute the main stream in application related research. He is author or co-author of 10 books and more than 180 articles mainly related the specified above Software Engineering and application categories. He was supervisor of 15 Ph.D. thesis. He is a member of IEEE Computer Society, ACM, Computer Science Committee of PAN, and 2 Scientific Committees of PAU and many other scientific committees.