UDC 004.738:004.94

Oleksandr N. Martynyuk[1], Candidate of Technical Sciences, Associate Professor of the Computer Intellectual Systems and Networks Department, E-mail: anmartynyuk@ukr.net, Scopus ID: 57103391900, ORCID: http://orcid.org/0000-0003-1461-2000
Oleksandr V. Drozd[1], Doctor of Technical Sciences, Professor of the Computer Intellectual Systems and Networks Department, E-mail: drozd@ukr.net, Scopus ID: 55388226700, ORCID: http://orcid.org/0000-0003-2191-6758
Sergey A. Nesterenko[1], Doctor of Technical Sciences, Professor of the Computer Intellectual Systems and Networks Department, vice-rector, E-mail: sa_nesterenko@ukr.net, Scopus ID: 55386373800, ORCID: http://orcid.org/0000-0002-3757-6594
Tamem Ahmesh[1], Aspirant of the Department of Computer Intellectual Systems and Networks, E-mail: tamim.nor@yahoo.com, ORCID: http://orcid.org/0000-0001-7342-4339
[1] Odessa National Polytechnic University, Avenue Shevchenko, 1, Odessa, Ukraine, 65044

BEHAVIORAL VERIFICATION OF INTERNET OF THINGS SYSTEMS BY PETRI NETS

Abstract*. The rapid development, implementation in all spheres of human activity and the growing responsibility of the functions of the Internet of things systems tighten and complicate the requirements for the reliability of their design decisions at the development stages and operability during their implementations. Well-known methods for verifying projects and implementations rely on the means of system, structural, functional, design and technological analysis and synthesis of Internet of things systems. However, their capabilities do not underestimate the feasibility of developing formalized models and verification methods, in particular, their integration at the early system-functional stages, where "manual" design is inevitable and essential. This paper presents elements of a comprehensive behavioral verification of projects at the system-functional level for Internet of Things systems represented using input UML diagrams and designed Petri nets. Verification includes at the first stage a "manual" system analysis of the correctness of entities and relationships for input UML diagrams and simple Petri nets, performed as their cover when activated in visual modeling. For this stage, general estimates of the costs of redesigning in case of errors are given. At the second stage in the verification of the system-functional level, an automated analysis of the correctness of more complex Petri nets corresponding to real objects is performed in the CPN Tools environment with modeling of their behavior and construction of a graph of reachable states (markups). In this case, the result allows the dynamics of the Petri net to evaluate the presence of dead ends, hanging peaks, endless cycles, safety and liveliness properties, and, if necessary, redesign. The combined integrated use of system "manual" and functional automated verification for input UML models and projected models of Petri nets allows reducing the time of designing Internet of things systems by eliminating or timely eliminating design errors.*

Keywords*: Internet of things; behavioral verification; Petri net; coverage of verified properties*

Introduction

The rapid development of the Internet has led to the emergence of a new concept for the Internet of Things (IoT). A feature of IoT is that the union of computers and people is transformed into a union of intellectual things [1-3]. Currently, a developed concept and related technologies have been formed in IoT, and here they are being rapidly deepened and expanded with the prospect of covering the IoT of the entire global network. It can be noted that the Internet is transforming into a significantly sophisticated, intelligent and truly global version of IoT. This is due, in particular, to the fact that the number of devices that provide and use an expanding list of Internet services is growing exponentially, now reaching almost everyone with the help of new communication technologies. A powerful distributed and shared source of a wide variety of information, computing, communication, management, intelligent services provided to end users has appeared and is rapidly developing. The conceptual borrowed multi-agent IoT, involving the use and development of the properties of autonomy, mobility, intelligence and cooperativeness, provides a high level of flexibility, operational reconfigurability of IoT to current state of hosting environment [4-7].

IoT is an approach to connecting a wide variety of services received from various sources on any virtual platform or Internet infrastructure. In 1999, Bill Joy defined the connections between devices in the Internet taxonomy [8; 9], and Kevin Ashton coined the term "Internet of Things" for related devices. The basic idea of IoT is to provide the possibility of an autonomous exchange of services between uniquely identifiable devices in the real world, which are now increasingly supporting radio frequency identification (RFID) and wireless sensor networks (WSNs), which allow them to make independent decisions depending on what action performed [7].

Many of the indicated integrable properties and mechanisms of IoT, both borrowed and proprietary, the responsibility, and often the criticality of the application, place high demands on the reliability, design and functioning of IoT. In particular, verification of projects and realizations of IoT implementations at system-functional level seems relevant and not trivial [9-12].

Applied mathematical base of verification of systems of Internet of Things (IoT) and their projects on high functional level may be based on its formal specifications with the use of corresponding high-level structural and functional linguistic foundation. This base may include automata [10] and Petri nets [13-18].

The purpose of this article

The main goal of this work is to reduce the time required for modeling and verification. Tasks to be solved include behavioral verification and testing of architectural properties and processes for IoT systems on the basis of Petri nets.

System behavioral verification of IoT

The objects of behavior of IoT are taken as input for Petri Nets and include: a) specifications of the technical description of the architecture of components, subsystems IoT and IoE-based systems, as well as such systems, defining the structure of topological relationships, functions, information objects, interfaces of topological interactions, the temporal behavior of functions and scenarios; b) previously prepared Petri nets, set accordingly behavioral models of process, components, IoT systems, for which system behavior verification are needed [19-22].

The following objects are considered as output objects for Petri Nets: the obtained correct, verified Petri nets, representing functional models of process, components, subsystems of IoT systems, system verification, obtained special conditions, parameters and scenarios of such verification, and also special the results of the fulfillment of conditions, the application of parameters and scenarios [23-27].

Models and methods for analyzing the functioning of Petri Netscan are used in processes of verification of the various aspects of behavior of the IoT systems and their components [28; 29].

Features in verification of processes in functioning of the IoT systems, their components on the basis of their representation by asynchronous, multiprocessing enhanced and hierarchical Petri nets.

Simulation is the imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed, this model represents the key characteristics or behaviors/functions of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time. As before, verification are independent procedures that are used together for checking that service or system meets requirements and that it fulfills its intended purpose. The behavioral online and offline testing the conformity of the behavior of the system under check to the behavior of the reference system, in the mode, respectively, for the first, basic operating functioning and, for the second, specific testing functioning.

Behavior modeling of features for functioning of IoT systems using Petri Nets. Behavior modeling is performed taking into scenarios and functions of the architecture of IoT for interactions of ports and interfaces into inter-component structure. The functional features of IoT and also their processes affect the classification of Petri Nets with special functions input/output, storage, processing for IoT.

Behavioral analysis of IoT systems is focused on the modeling and verification of the basic, component, interface and subsystem functions presented at the system level of the IoT architecture.

Thus, these tasks are defined as follows actions correctness, verification, testing [28].

Correctness proves existence of reference structural properties and includes the two stages.

– First stage and its steps for correctness confirms, that the model has the preference properties of: absence of static locks; completeness; unambiguous correspondence of states; lack of redundancy; limitedness; lack of dynamic locks; self-synchronization; partial correctness; complete correctness; security; liveliness.

– Second stage and its steps lowers the dimension of the model of achievable states due steps: analysis of the achievable states and markings; structural and functional decomposition; previously created “equivalent” states; limiting the number of parameters and detectable errors.

Verification proves compatibility of the specifications for the service objects of verifiable and adjacent levels and includes the two stages.

– First stage proves that specification of the lower level, that are used by these service objects, is consistent with the description provided by the verified level.

– Second stage proves that specification of the higher level, that uses these service objects, is consistent with the description provided by the verified level.

Behavioral online and offline testing proves existence of special subject functional properties and includes four stages.

– First stage consists in verifying, that the behavior of the system on conceptual boundary with the environment corresponds to the intended one.

– Second stage allows to get test scenarios, recognizing and checking experiments in terms of service primitives and data elements of the system.

– Third stage consists in passive recognizing experiment by behavioral on-line testing. This experiment is based on a method of recognizing behavioral automata check in the external flow of

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

the system's operational functioning based on the identification of reference states in the first step. Experiment establishes the correspondence of the reference and verified models by searching for recognizing fragments in a fixed operational behavior of the real system in the second step.

– Four stage executes an active checking experiment by testing. This experiment is based on a method of constructing behavioral checks in the internal specially formed flow of test functioning of the system based on the identification of reference states in the first step. Experiment establishes the correspondence of the reference and verified models by the embedding of checking fragments in the test behavior of the real system in the second step.

Realizing of behavioral verification of IoT

In paper, we apply manual/automatic method for verification behavior of models of IoT systems represented by input UML and main Petri net.

General objectives include familiarization with: the fundamentals of model verification, in particular, based on the use of intuitive – "manual", full-selection, and automata testing in the "promotion" method; the principles of operation and the basics of using the environment CPN Tools, which provides a performs the basic functional, event-time modeling and verification of IoT projects.

Tasks of realizing of verification includes: constructing of models of IoT system implemented at the application level; performing verification of IoT models based on CPN Tools environment.

Realizing of verification includes: analysis and subject optimization of the complexity of the design, verification and redesign of the IoT systems; assessment and subject optimization of the length, multiplicity and completeness of verification checks for designing of IoT systems.

Procedure for performing formal input UML diagrams and simple Petri nets models of the architecture of some IoT system and its manual verification scenarios include eleven realizing stage:

1. Construction behavioral models of the architecture selected according to the IoT system task – activity diagram and state diagram.

2. The upper estimate of the complexity of the behavioral analysis of input UML diagram is performed using a simplified formula for basic entities and relations.

$$C_{UML} = a*(n_e+n_e^2+3*n_r), \quad (1)$$

where: n_e is the number of classifiers of the entities;

n_e^2 is the number of cells in a square matrix of possible relationships between entities;

n_r is the number of real assigned ratios of the diagram, the multiplier "3" means the need to consider both the relation r itself and the two entities e1 and e2 incident to it; a - conditional coefficient of abstraction level (for technical specifications – 3, structural-functional – 2, object-component – 2, event-time – 1, automata-algorithmic – 1).

3. Manual construction a script for behavioral verification of properties based on a full-choice testing method for constructing a path covering all entities for each of the selected diagrams and determining the path length.

4. The upper estimate of the computational complexity of the covering path (including its construction) using the simplified formula:

$$C_{Path} = a*(n_e^2/2+n_r). \quad (2)$$

5. The upper estimate of the reduced complexity of the construction due to the use of behavioral verification script in comparison with the case of re-design in case of an error, when the complexity of the construction of input UML diagram is doubled, using the simplified formula:

$$C\Delta_C = 2*C_{UML} - (C_{UML}+C_{Path}) = \\ =a*(n_e+ n_e^2/2 +2*n_r). \quad (3)$$

6. Visual construction of Petri nets in the CPN Tools according to the input UML diagrams of the IoT system based of activities and states.

7. Determination of the upper estimate of the complexity of the analysis and the construction of Petri nets using a simplified formula for their entities - positions, transitions, chips and relations for them:

$$C_{Petri} = a*(n_e+n_q+4*n_r), \quad (4)$$

where: n_{ep} is the number of position classifiers, n_{et} is the number of transition classifiers,

n_{em} is the number of chip token classifiers, $n_e=n_{ep}+n_{et}+n_{em}$; $n_q=n_{ep}*n_{et}*n_{em}$ is the number of cells in the three-dimensional matrix of possible relationships between entities; n_r is the number of real assigned ratios of the diagram, the multiplier "4" means that it is necessary to consider both the relation r itself and the three incident entities e_{1p}, e_{2t}, e_{3m}; a – conditional coefficient of abstraction level (for technical specifications – 3, structural-functional – 2, object-component – 2, event-time – 1, automaton-algorithmic - 1).

8. Constructing of scenarios of static and dynamic verification of selected properties of Petri net based on CPN Tools for:

– step-by-step simulation;

– end-to-end modeling;

– constructing a graph of attainable markings;

– matrix definition of position and transition invariants.

9. Construction of the test, as a covering all the positions and transitions of the Petri net based on recurring behavioral testing, and determining the length of this path.

10. The upper estimate of the complexity of the covering path using the simplified formula:

$$C_{PetriPath} = a*(n_q/3+n_r). \quad (5)$$

The upper estimate of the reduced complexity of the construction due to the use of the behavioral verification scenario in comparison with the case of re-design in case of an error when the complexity of analyzing the Petri net doubles, using the formula:

$$C\Delta_C=2*C_{Petri}-(C_{Petri}+C_{PetriPath}) = \\ =a*(n_e+2n_q/3+3*n_r). \quad (6)$$

Examples of verification of Petri nets

The design and verification of architecture of the IoT selected according to the setting option using Petri nets. Two type of Petri nets demonstrate the design and verification of IoT system.

State diagrams. State diagrams (statechart diagrams) show the possible states of components or the system as a whole, transitions between them in response to any events and actions performed during this. States can be arranged hierarchically, they can be decomposed into parallel substrates. A subsystem can have global (within its framework) variables storing some data. The values of these variables are common parts of all depicted states. The peculiarity of the state diagram is the use of a modified association of state transitions, not transitions — message exchanges, as well as the possibility of representation of all scenarios as relations. And also in the temporal, rather than spatial, character of the representation of transition relations.

The general logic of work is in the form of input state diagrams (see Fig. 1).

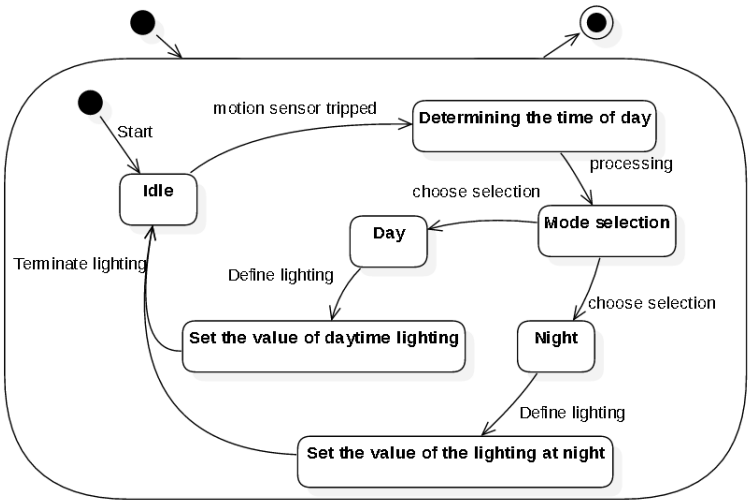

Fig. 1. Lighting system statechart diagram

Entities verification of the input state diagram:

– States: Idle, Determining the time of day, Mode selection, Day, Night, Set the value of daytime lighting, Set the value of the lighting at night, StartOut, End, StartIn.

– Relations of the modified association - state transitions: StartOut →start()→system; system →start()→StartIn; StartIn →start()→Idle; Idle → motion sensor tripped → Determining the time of day; Determining the time of day → processing → Mode selection; Mode selection → choose selection → Day; Day → Define lighting → Set the value of daytime lighting; Set the value of daytime lighting → Terminate lighting → Idle; Mode selection → choose selection → Night; Night → Define lighting → Set the value of the lighting at night; Set the value of the lighting at night → Terminate lighting → Idle; StartPage →back()→system; system →back()→ End.

A verification of entities of a given, relatively simple state diagram is performed on the basis of a complete transition coverage test — one of the possible Eulerian paths through association relations (transitions – calls to the corresponding methods) from the initial Start entity to the final End, covering all verification entities, with registration of results and their interactive verification, in particular, of the property values against the standards (objects and association relations).

The upper complexity estimates are the form:

$$C_{UML}=a*(n_e+n_e^2+3*n_r)=3*(10+10^2+3*11)=429$$
$$C_{Path}=a*(n_e^2/2+n_r)=3*(10^2/2+11)=183$$
$$L_{Path} = 1+2+9 = 12$$
$$\Delta_C=a*(n_e+ n_e^2/2 +2*n_r)=3*(10+10^2/2 +2*11)=246.$$

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

Reducing the complexity of analysis through the use of behavioral verification script amounted to 246 conventional units of analysis.

Petri Net. To verification of the construction and identification of errors, we translate the diagrams into Petri net (Fig. 2). For the constructed Petri net, the transitions correspond to the actions on the input state diagram (Fig. 1).

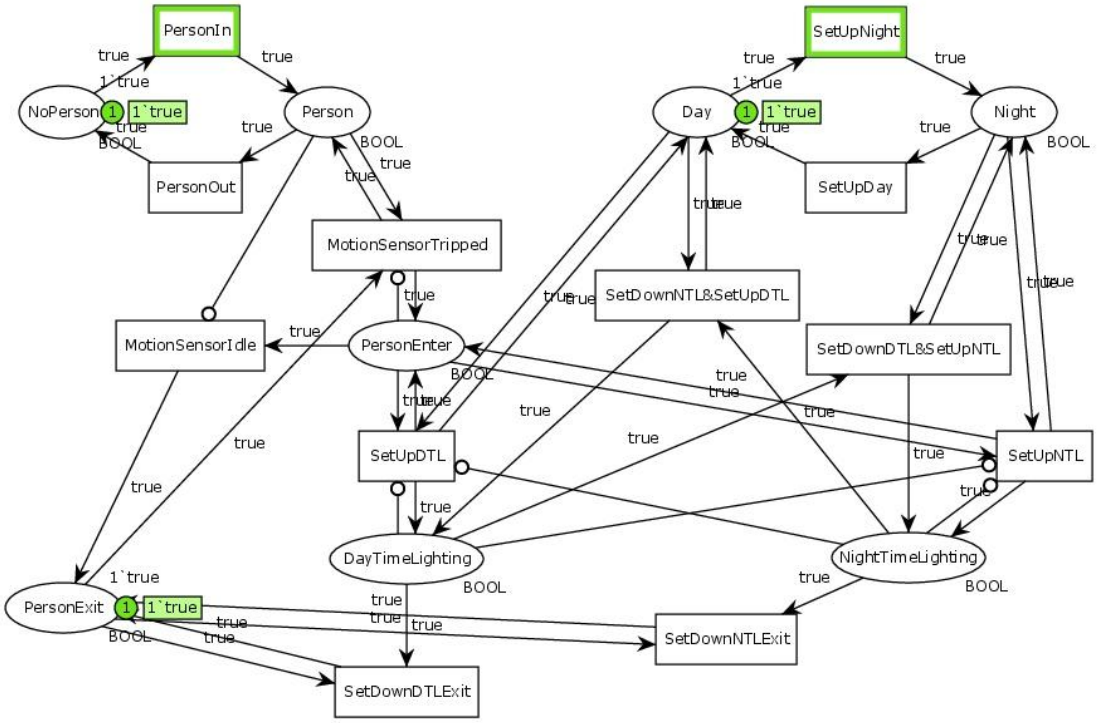

Fig. 2. Lighting system statechart diagram

Comparing the detailed graph generated using CPN Tools; one can draw a conclusion about their similarity and the possibility of "gluing" the graphs (scenarios). It is possible to analyze the state graph by examining individual scenarios obtained from the general algorithm (input state diagram) and then combining them if necessary. This network can be divided into 3 parts. The first, top left (NoPerson, Peerson, PersonIn, PersonOut), is responsible for having a person as a physical object. The second one from the top right (Day, Night, SetUpDay, SetUpNight), is responsible for changing the time of day. The third part, all other blocks, are responsible for the operation of system, or rather, its reaction to movement of chips in the first two parts. In fig. 1th shows the initial position of the Petri net, else eight diagrams may represent modelling stages.

A detailed Petri net with a larger dimension and complexity of behavior, than diagrams, requires computer construction, modeling, explicit or implicit determination of covering ways of moving chips (markups). This paper presents the use of CPN Tools to obtain a graph of reachable states (fig. 3) with an implicit definition of the set of covering paths.

The state of the network is uniquely determined by its labeling – distribution of chips by position. The vertices of the graph are valid markings of the Petri net; arcs are indicated by symbol of triggered transition.

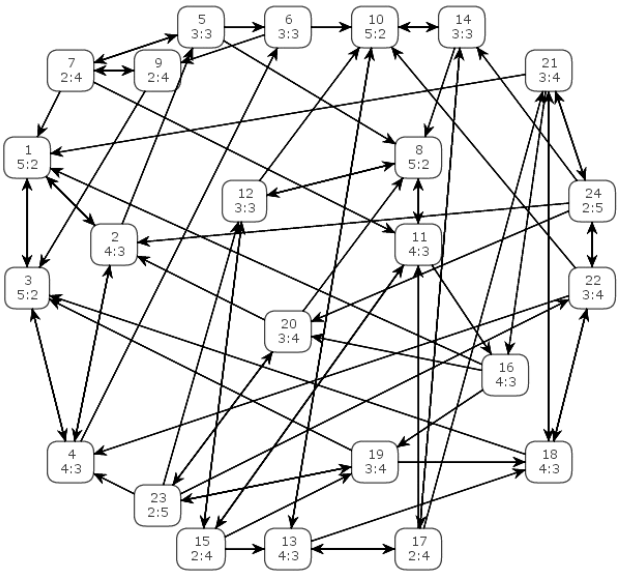

Fig. 3. Graph of reachable states

The arc is built for each excited transition. The construction stops when we receive a label in which none of the transitions is excited, or a label that is

already contained in the graph. Note that the graph of achievable markings is an automaton.

The modeling of the Petri net and the obtained graph of reachable states confirm the absence of design errors, in particular, dead ends, dangling vertices, infinite loops, that is, the correctness of both the Petri net and the input state diagram.

Conclusions

Formalized analysis and synthesis of computer systems is becoming an important part of the process of training specialists in the field of computer science. This paper presents an "early" system verification of the basic entities and relationships of the Internet of Things systems, performed in an interactive "manual" mode for the corresponding input UML diagrams and simple Petri nets. The above manual estimates make it possible to estimate the costs of redesigning at this level. Also, automated verification of the dynamics of complex Petri nets representing the behavior of real objects at the functional level in the CPN Tools environment is presented.

References

1. Pallavi, Sethi & Smruti, R. (2017). "Sarangi Internet of Things: Architectures, Protocols, and Applications", *Journal of Electrical and Computer Engineering* Vol. 5, Article ID 9324035, 25 p. https://doi.org/10.1155/2017/9324035.

2. (2018). Kharchenko, Vyacheslav, Ah Lian Kor, & Rucinski, Andrzej (Eds). "Dependable IoT for Human and Industry: Modeling, Architecting, Implementation", *River Publishers Series in Information Science and Technology*, 450 p.

3. Tara, Salman "Networking Protocols and Standards for Internet of Things". https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot/index.html, Internet source.

4. Talari, Saber, Shafie-khah, Miadreza, Siano, Pierluigi, Loia, Vincenzo, Tommasetti, Aurelio & Catalão, João P. S. (2017). "A Review of Smart Cities Based on the Internet of Things Concept", *Publ. Energies* 10, 421, 23 p. http://www.mdpi.com/1996-1073/10/4/421/pdf, DOI.org/10.3390/en10040421.

5. Strielkina, A. A., Uzun, D. D., Kharchenko, V. S., Tetskyi, A. H. (2018). "Modelling and Availability Assessment of Mobile Healthcare IoT Using Tree Analysis and Queueing Theory", in book: "Dependable IoT for Human and Industry: Modeling, Architecting, Implementation", Kharchenko, Vyacheslav, Ah Lian Kor, Rucinski, Andrzej (Eds.), *Publ. River Publishers Series in Information Science and Technology*.

6. Dovgal, V. A., Dovgal, D. V. (2017). "Management of Resources on the Internet of Things", *Distance educational technologies: proceedings of the II Russian scient.-pract. conf.*, Yalta: Simferopol: *Publ. ARIAL*, pp. 168-173.

7. Ashton, Kevin. (2009). "That "Internet of Things", *RFID Journal*. (2009. 22 June). URL: http://www.rfidjournal.com/articles/pdf?4986

8. Evans, D. (2009). "Internet of Things", *Cisco, white paper*. https://www.cisco.com/c/dam/en_us/about/ac79/_docs/innov/IoT_IBSG_0411FINAL.pdf. Internet source.

9. Dovgal, V. A. & Dovgal, D. V. (2017). "Security Issues and Challenges for the Intellectual Networks Founded on the Internet of Things", *The Bulletin of the Adyghe State University*. Ser. Natural-Mathematical and Technical Sciences, Iss. 4. pp. 140-147. URL: http://vestnik.adygnet.ru (in Russian).

10. Esparza, Javier. (2017). "Automata theory. an Algorithmic Approach. Lecture Notes", (August 26, 2017), 321 p. Available from: https://www7.in.tum.de/~esparza/autoskript.pdf.

11. Kondratenko, Y., Kozlov, O., Topalov, A., Korobko, O. & Gerasin, O. (2018). "Automation of Control Processes in Specialized Pyrolysis Complexes Based on Industrial Internet of Things", in book: "Dependable IoT for Human and Industry: Modeling, Architecting, Implementation", Kharchenko, Vyacheslav, Ah Lian Kor, Rucinski, Andrzej (Eds.), *River Publishers Series in Information Science and Technology*.

12. Boyarchuk, A., Kharchenko, V., Illiashenko, O., Maevsky, D., Phillips, C., Plakhteev, A. & Vystorobska, L. (2018). "Internet of Things for Human and Industry Applications: ALIOT Based Curriculum", in book: "Dependable IoT for Human and Industry: Modeling, Architecting, Implementation", Kharchenko, Vyacheslav, Ah Lian Kor, Rucinski, Andrzej (Eds.), *River Publishers Series in Information Science and Technology*.

13. Hahanov, V., Litvinova, E., Chumachenko, S. (2017). "Cyber Physical Computing for IoT-driven Services", *Publ. Springer*, 279 p.

14. Desel, Jorg, Esparza & Javier Free (1995). "Choice Petri Nets", *Cambridge University Press, Cambridge*, 256 p. Available from: https://www7.in.tum.de/~esparza/fcbook-middle.pdf.

15. (2007). Kleijn, J., Yakovlev, A. (Eds). "Petri nets and Other Models of Concurrency", *ICATPN, Lecture Notes in Computer Science*, Vol. 4546, ISBN 978-3-54073093-4, *Publ. Springer-Verlag*, 2007, 515 p.

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

16. Poliakov, I., Mokhov, A., Rafiev, A., Sokolov D. & Yakovlev, A. (2008). "Automated Verification of Asynchronous Circuits Using Circuit Petri Nets", *Proceedings of the 14th IEEE International Symposium on Asynchronous Circuits and Systems*, Newcastle upon Tyne, UK, (April 2008), pp. 161-170. DOI**:** 10.1109/ASYNC.2008.18.

17. Zaitsev, D. A. (2013). "Toward the Minimal Universal Petri Net", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1-12.

18. Westergaard, Michael. (2010). "CPN Tools", Eindhoven: 46 p. https://westergaard.eu/wp-content/uploads/2010/09/CPN-Tools.pdf. Internet source.

19. Sugak, A., Martynyuk, O. & Drozd, O. (2015). "The Hybrid Agent Model of Behavioral Testing", *International Journal of Computing*, Vol. 14, Issue 4, Ternopol: Ukraine, pp. 232-244.

20. Kharchenko, V., Gorbenko, A., Sklyar, V. & Phillips, C. (2013). "Green Computing and Communications in Critical Application Domains: Challenges and Solutions", *IX International Conference of Digital Technologies*, Zhilina: Slovak Republic, pp. 191-197.

21. Sugak, A., Martynyuk, O. & Drozd, O. (2015). "Models of the Mutation and Immunity in Test Behavioral Evolution", *Proceedings of the 2015 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, Warsaw: Poland, pp. 790-795. **DOI:** 10.1109/IDAACS.2015.7341411.

22. Martynyuk, O., Sugak, A., Martynyuk, O. & Drozd, O. (2017). "Evolutionary Network of Testing of the Distributed Information Systems", *Proceedings of the 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, Bucharest: Romania, pp. 888-893. https://ieeexplore.ieee.org/document/8095215.

23. Wang, Anduo. (2010). "Formal Analysis of Network Protocols", *University of Pennsylvania Department of Computer and Information Science Technical Report*, No. MS-CIS-10-16. 32 p. https://repository.upenn.edu/cgi/viewcontent.cgi?article=1970&context=cis_reports

24. Câmara, Daniel. (2009). "Formal Verification of Communication Protocols for Wireless Networks", *Publ. Belo Horizonte*, 136 p. http://www.eurecom.fr/~camara/files/ThesisCamara_FormalVerification.pdf. Internet source.

25. Gomes, Luís & Fernandes João M. (2010). "Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation", 494 p. ISBN13: 9781605667508 | ISBN10: 1605667501 | EISBN13: 9781605667515 | DOI: 10.4018/978-1-60566-750-8.

26. Schamai, Wladimir. (2013). "Model-Based Verification of Dynamic System Behavior against Requirements Method, Language, and Tool", Linköping University SE-581 83, Linköping, Sweden, 257 p. Internet source.

27. Kudryavtsev, V. B., Grunskii, I. S., & Kozlovskii, V. A. (2010). "Analysis and Synthesis of Abstract Automata", *Journal of Mathematical Sciences*, (September 2010), Vol. 169, Issue 4, pp. 481-532.

28. Martynyuk, O., Drozd, O., Ahmesh, Tamem, Bui Van Thuong, Sachenko, A., Mykhailova, H. & Dombrovskyi, M. (2019). "Hierachical Model of Behavior On-line Testing for Distributed Information Systems", *The 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, Vol. 2, Metz, France: pp. 724-729. DOI: 10.1109/IDAACS.2019.8924314.

29. Drozd, M., Drozd, O. & Antoniuk, V. V. (2019). "Power-oriented Checkability and Monitoring of the Current Consumption in FPGA Projects of the Critical Applications", *Scientific Journal Applied Aspects of Information Technology*, Vol. 2, No 2, Odessa: Ukraine, pp. 105-114.

Received 07.10.2019
Received after revision 19.11 2019
Accepted 2.12.2019

УДК 004.738:004.94

[1]**Мартинюк, Олександр Миколайович**, канд. техніч. наук, доцент, доцент кафедри комп'ютерних інтелектуальних систем і мереж, E-mail: anmartynyuk@ukr.net, Scopus ID: 57103391900, ORCID: http://orcid.org/0000-0003-1461-2000

[1]**Дрозд, Олександр Валентинович**, доктор техніч. наук, професор кафедри комп'ютерних інтелектуальних систем і мереж, E-mail: drozd@ukr.net, Scopus ID: 55388226700, ORCID: https://orcid.org/0000-0003-2191-6758
[1]**Нестеренко, Сергій Анатольевич,** доктор техніч. наук, професор кафедри комп'ютерних інтелектуальних систем і мереж, проректор, E-mail: sa_nesterenko@ukr.net, Scopus ID: 55386373800, ORCID: http://orcid.org/0000-0002-3757-6594
[1]**Ахмеш, Тамем**, аспірант кафедри комп'ютерних інтелектуальних систем і мереж, E-mail: tamim.nor@yahoo.com, ORCID: http://orcid.org/0000-0001-7342-4339
[1] Одеський національний політехнічний університет, пр. Шевченка, 1, м. Одеса, Україна, 65044

ПОВЕДІНКОВА ВЕРИФІКАЦІЯ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ НА ОСНОВІ МЕРЕЖ ПЕТРІ

Анотація. *Швидкий розвиток, впровадження в усі сфери діяльності людини і зростання відповідальності виконуваних функцій систем Інтернет речей посилюють і ускладнюють вимоги до достовірності їх проектних рішень на етапах розробки і працездатності в ході експлуатації реалізацій. Відомі методи верифікації проектів і реалізацій спираються на засоби системного, структурного, функціонального, конструкторсько-технологічного аналізу і синтезу систем Інтернет речей. Однак їхній ресурс не применшує доцільність розвитку формальних моделей і методів верифікації, зокрема, їх комплексування на ранніх системно-функціональних етапах, де неминуче і істотно «ручне» проектування. У даній роботі наведені елементи комплексної поведінкової верифікації проектів системно-функціонального рівня для систем Інтернет речей, які подаються за допомогою вхідних UML-діаграм і проектованих мереж Петрі. Верифікація включає на першому етапі «ручний» етап системного аналізу коректності сутностей і відносин вхідних UML-діаграм і простих мереж Петрі, що виконується як їх покриття при активізації в візуальному моделюванні. Для цього етапу наведено загальні оцінки витрат на перепроектування в разі появи помилок. На другому етапі в верифікації системно-функціонального рівня виконується автоматизований аналіз коректності більш складних мереж Петрі, що відповідають реальним об'єктам, в середовищі CPN Tools з моделюванням їх поведінки і побудовою графа досяжних станів (розміток). В цьому випадку результат дозволяє динаміку роботи мережі Петрі, оцінити наявність тупиків, висячих вершин, нескінченних циклів, властивості безпеки і жвавості і при необхідності виконати перепроектування. Спільне комплексне застосування системної «ручний» і функціональної автоматизованої верифікації для вхідних UML-моделей і проектованих моделей мереж Петрі дозволяє скоротити час проектування систем Інтернет речей за рахунок виключення або своєчасного усунення проектних помилок.*

Ключові слова: *системи Інтернет речей; поведінкова верифікація; мережа Петрі; покриття перевіряються властивостей; складність верифікації*

УДК 004.738:004.94

[1]**Мартынюк, Александр Николаевич**, канд. технич. наук, доцент, доцент кафедры компьютерных интеллектуальных систем и сетей, E-mail: anmartynyuk@ukr.net, Scopus ID: 57103391900, ORCID: http://orcid.org/0000-0003-1461-2000
[1]**Дрозд, Александр Валентинович**, д-р технич. наук, профессор кафедры компьютерных интеллектуальных систем и сетей, E-mail: amberk4@gmail.com, Scopus ID: 55388226700, ORCID: https://orcid.org/0000-0003-2191-6758
[1]**Нестеренко, Сергей Анатольевич,** д-р технич. наук, профессор кафедры компьютерных интеллектуальных систем и сетей, проректор, E-mail: sa_nesterenko@ukr.net, Scopus ID: 55386373800, ORCID: http://orcid.org/0000-0002-3757-6594
[1]**Ахмеш, Тамем**, аспирант кафедры компьютерных интеллектуальных систем и сетей, E-mail: lukianov@bsu.by, ORCID: 0000-0001-7342-4339
[1]Одесский национальный политехнический университет, пр. Шевченко, 1, г. Одесса, Украина, 65044

ПОВЕДЕНЧЕСКАЯ ВЕРИФИКАЦИЯ СИСТЕМ ИНТЕРНЕТА ВЕЩЕЙ НА ОСНОВЕ СЕТЕЙ ПЕТРИ

Аннотация. *Быстрое развитие, внедрение во все сферы деятельности человека и рост ответственности выполняемых функций систем Интернет вещей ужесточают и усложняют требования к достоверности их проектных решений на этапах разработки и работоспособности в ходе эксплуатации реализаций. Известные методы верификации проектов и реализаций опираются на средства системного, структурного, функционального, конструкторско-технологического анализа и синтеза систем Интернет вещей. Однако их возможности не преуменьшают целесообразность развития формализованных моделей и*

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

методов верификации, в частности, их комплексирования на ранних системно-функциональных этапах, где неизбежно и существенно «ручное» проектирование. В настоящей работе приведены элементы комплексной поведенческой верификации проектов системно-функционального уровня для систем Интернет вещей, представляемых с помощью входных UML-диаграмм и проектируемых сетей Петри. Верификация включает на первом этапе «ручной» этап системного анализа корректности сущностей и отношений входных UML-диаграмм и простых сетей Петри, выполняемого как их покрытия при активизации в визуальном моделировании. Для этого этапа приведены общие оценки затрат на перепроектирование в случае появления ошибок. На втором этапе в верификации системно-функционального уровня выполняется автоматизированный анализ корректности более сложных сетей Петри, соответствующих реальным объектам, в среде CPN Tools с моделированием их поведения и построением графа достижимых состояний (разметок). В этом случае результат позволяет динамику работы сети Петри, оценить наличие тупиков, висячих вершин, бесконечных циклов, свойства безопасности и живости и при необходимости выполнить перепроектирование. Совместное комплексное применение системной «ручной» и функциональной автоматизированной верификации для входных UML-моделей и проектируемых моделей сетей Петри позволяет сократить время проектирования систем Интернет вещей за счет исключения или своевременного устранения проектных ошибок.

Ключевые слова: *системы Интернет вещей; поведенческая верификация; сеть Петри; покрытие проверяемых свойств; сложность верификации*

Oleksandr N. Martynyuk, Candidate of Technical Sciences, Associate Professor
Research field: behavioral testing and diagnosis of computer systems, formal verification of project, recognizing of digital systems

Oleksandr V. Drozd, Doctor of Technical Sciences, Professor
Research field: Testing and diagnosis of computer systems, arithmetical foundations of computer systems, computer systems and components

Sergey A. Nesterenko, Doctor of Technical Sciences, Professor
Research field: analyze, modelling and synthesis of computer systems and networks

Tamem Ahmesh, Aspirant
Research field: testing and diagnosis of computer systems, computer networks