VOL. 16, NO. 16, AUGUST 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

A TECHNIQUE FOR CHECKING THE ADEQUACY OF FORMAL MODEL

Vadym Shkarupylo[1,2], Jamil Abedalrahim Jamil Alsayaydeh[3,4], Igor Tomičić[5], Alexander Chemeris[6] and Valentyna Dusheba[1]

[1]Department of Mathematical and Computer Modelling, G. E. Pukhov Institute for Modelling in Energy Engineering, NAS of Ukraine, General Naumov Str., Kyiv, Ukraine

[2]Department of Computer Systems and Networks, National University of Life and Environmental Sciences of Ukraine, Heroyiv Oborony Str., Kyiv, Ukraine

[3]Fakulti Teknologi Kejuruteraan Elektrik dan Elektronik, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, Durian Tunggal, Melaka, Malaysia

[4]Center for Advanced Computing Technology, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, Durian Tunggal, Melaka, Malaysia

[5]Faculty of Organization and Informatics, University of Zagreb, Pavlinska Str., Varaždin, Hrvatska

[6]Department of Modeling and Econometrics, G. E. Pukhov Institute for Modelling in Energy Engineering, NAS of Ukraine, General Naumov Str., Kyiv, Ukraine

E-Mail: shkarupylo.vadym@ipme.kiev.ua

ABSTRACT

In this paper, the question on the expediency of checking the model, the model checking method is applied to, is discussed. To this end, corresponding technique has been proposed. Named technique is based on differentiation between the concepts of analytical plane of model perception and the concepts of corresponding implementation plane. The technique is grounded on the following constituents: Kripke structure - for analytical interpretation of formal specification; Temporal Logic of Actions and corresponding formalism - as the instruments for shifting from the analytical plane to the implementation one; TLC model checker - to examine the correctness of formal specification – with respect to the concepts of implementation plane. To prove the proposed technique, the case study has been conducted. To this end, the algorithms from the spacecraft domain have been considered. To verify the resulting specifications, two alternative implementations of TLC model checker have been applied.

Keywords: adequacy, correctness, formal specification, model checking, TLA, TLC.

1. INTRODUCTION

1.1 Distinctive Features of Model Checking Technique

Nowadays, model checking techniques and corresponding implementations - model checkers (MCs) - are broadly adopted during the developing of diverse safety-critical systems, e.g., real-time operating systems (RTOS) [1], modern processors design solutions (DSs) [2], control systems in Finnish nuclear industry [3], virtual cloud resources [4], etc. The distinctive features of these techniques can be formulated as follows:

a) Decision on system property specification correctness is carried out on the basis of corresponding model, generated through model checking of formal specification (FS).

b) In contrast to the alternatives, e.g., deductive verification, equivalence checking, the process of MCs utilization can be thoroughly automated. This aspect is of topical importance, because of the complexity of modern software system's DSs, e.g., the Amazon Web Services (AWS) [5].

Both aforementioned features of MCs demonstrate the drawback and the advantage, respectively.

1.2 Peculiarities of Work Conducted

This paper addresses the first aforementioned feature, i.e., the necessity to deal with the model, rather than a system itself. Here comes the following peculiarity: when considering the model checking technique and its application in certain domains, the problem domain needs to be specified in an unambiguous manner. To this end, the following assumptions have been made:

a) An engineering process is approached as a sequence of the steps - requirements analysis, designing, implementation, validation.

b) MC technique is supposed to be applied during the designing stage of engineering process.

c) Developer's perception of system under design is addressed with iteratively created and refined artifacts. To this end, Manfred Broy's artifact-based approach to engineering process formalization has been adopted. With respect to this approach, an "artifact" is considered as a document with structure and content, i.e., an outcome of certain step [6].

The proposed technique is intended to be applied during the designing stage of engineering process. To this end, with respect to the aforementioned features of MCs, the DSs of a system under development, e.g., the

diagrams, are treated as the input data, the FSs to be synthesized from. After that, the MC technique is implicitly applied to the FS: through corresponding FSM (Finite-state Machine), i.e. transition system. Here comes the conceptual breach prompting answering the following question: does the developer's analytical perception of the DS correspond to the FSM generated through the automated model checking of the FS, and, as an outcome, can we trust the results of such verification? Here the differentiation between the concepts of analytical and implementation planes arises. Moreover, when addressing the implementation plane, the following positions need to be worked out: choosing the appropriate temporal logic; applying the "right" formalism, making it possible to conduct the verification in an automated manner; adopting proved MC technique.

To answer the question, the operative adequacy checking technique is required to be implemented. To this end, the following assumptions have been made:

a) When applying the MC technique, the FS is approached as a prototype, the FSM is synthesized with respect to.

b) "Adequacy" concept is stressed with respect to the FS and corresponding FSM.

Grounding on the importance of such technique elaboration is provided below.

The rest of the paper is organized as follows. In section 2, the analysis of the related work is conducted. In section 3, the description of the proposed technique is presented. In section 4, the results of experimental studies are discussed. In section 5, the conclusions and the thoughts on further research are given.

2. LITERATURE REVIEW

2.1 Grounding on the Instruments Applied

Addressing the assumptions provided in the previous paragraph, the following steps have been made:

a) Temporal Logic of Actions (TLA) [7], by Leslie Lamport, has been chosen as the basis because of the following factors: massive industrial use [1-4], applicability to concurrency checking [8], proved instrument to be applied in safety-critical domain, e.g., the railway control applications [9]. Moreover, TLA has been successfully used in novel Software-defined Networking (SDN) domain – to verify the rules of SDN-compatible switch [10].

b) To represent the FS in a form applicable to automated MC, the mathematically strict TLA+ formalism has been adopted [11].

c) To conduct the verification by way of MC, the TLC checker has been applied [7].

d) To group the TLA, TLA+ and TLC, the TLA Toolbox has been utilized [12].

Among the preconditions to these steps is the possibility to represent the FS with a single temporal formula, which is critical in terms of scalability, modularity and easiness of shifting the grain of atomicity of the FS.

2.2 Adequacy Checking Techniques Analysis

Taking into consideration the peculiarities of problem domain approached, the obvious way to do the adequacy checking is a topological one, when the transition system, i.e., the model, is treated as a graph, and typical approach includes resolving the problem of graph isomorphism. Though, due to the NP-completeness of this task [13] and significant size of state space, it seems to be infeasible in practice.

Depending on the artifact type, the FS is synthesized from (e.g., DS, implementation), the approaches to FS synthesis and corresponding adequacy checking vary significantly. For instance, FS adequacy has been considered with respect to the quality of tests obtained through the automated FS-based structural test case generation [14]. The DS has been considered here as an input data for FS synthesis. It has been found out that sticking to existing FSM-based coverage criteria, e.g., state, transition, decision coverage, provide even worse results, comparing to randomly generated tests. The semantics of FS language (RSML, Requirements State Machine Language) applied and the peculiarities of problem domain (flight-guidance system) have been proclaimed to be the reason to that. NuSMV symbolic model checker has been applied as an instrument.

FS adequacy delivering is tightly bound with an effect of FSM state space explosion. A technique to mitigate it, i.e., BDDs (Binary Decision Diagrams), is implemented in NuSMV method. As a case study, named method has been applied to foster the single-failure tolerance of nuclear reactor protection system [15]. FS adequacy has been approached here in terms of its completeness: FS encompasses not only the application logics, but also a hardware plane (hardware component failures, communication delays).

In contrast to the aforesaid, when the DS has been approached as an input data for the FS synthesis, the adequacy metrics can be applied with respect to the structure of software system implementation, e.g., Modified Condition and Decision Coverage (MC/DC) criterion [16]. It has been stated that manipulation with implementation structure to foster the MC/DC criterion has no positive effect on the adequacy of obtained tests. On the contrary, the significant effect has been obtained through increasing the coverage completeness, on pair with costs increase though. Such implementation coverage can be delivered through the in-code assertions, e.g., VCC

(Verifying Concurrent C) applied to verify the Microsoft Hyper-V hypervisor [17]. To check the FSM, an SMT (first-order Satisfiability Modulo Theories) solver has been utilized. Semantic aspect has been addressed here with a "typestate" notion. Though a completeness criterion can be satisfied by exhaustive coverage with assertions, a substantial drawback here is that the number of such assertions can significantly outpass the size of code verified. This aspect both complicates code analysis and increases related time costs.

By elaborating the direction, where the DS is considered as an input data, in given paper, the adequacy aspect is approached in terms of providing the transparent mechanism for "shifting" between the DS and corresponding FS, thus, diminishing the semantic discrepancy between the FS language and problem domain.

3. MATERIALS AND METHODS

3.1 Analytical Plane Concepts Formalization

3.1.1 Model checking task formulation

Let the model checking task is formulated as follows:

$$M, b \models \psi, \tag{1}$$

where M - Kripke structure, defined over an AP set of atomic prepositions; b - system behavior as a sequence of states - an analytical representation of system property; ψ - temporal formula to be satisfied in each element of b sequence, i.e., FS of b. The FS is implemented as a temporal formula to make it possible to check b in an automated manner with MC applied. To this end, the TLA temporal logic, corresponding TLA+ formalism and TLC model checker have been brought to the use.

The core idea inside the proposed technique is conceptually similar to the one that takes place during the test-driven development process (TDD): the tests are created first, and then the software module satisfying these tests is made [18]. Similarly, with respect to (1), M structure is approached conceptually as the tests, and ψ temporal formula, implemented on the basis of chosen formalism - as the software module satisfying these tests.

3.1.2 Kripke structure as the mathematical model

Let M structure (model) over the AP set is defined as follows [19]:

$$M = \langle S, S_0, R, L \rangle, \tag{2}$$

where S – total set of states: $S_0 \subset S$ – set of initial states; $R \subseteq S^2$ – set of transitions between states; $L: S \to 2^{AP}$ – states labeling function.

It should be noted that (2) structure differs from typical automaton by lacking the transitions labeling. This peculiarity simplifies model checking process, making it possible to represent the behavior of reactive system with potentially infinite set of states on the basis of a finite one.

Let b is an infinite sequence of states, formed through $(s, s') \in R$ transitions, where $s \in S$ - current state, $s' = R(s) \in S$ – subsequent state:

$$b = s_0, s_1, ..., s_l, s_l ..., \tag{3}$$

where $s_1 = R(s_0)$, $s_2 = R(s_1) = R(R(s_0)), \ldots,$ $s_l = R(s_{l-1}) = R(R(s_{l-2})) = R^l(s_0)$, where l upper index is the number of times the R has been applied, e.g. the power of composition on the basis of R. To sum up, $\forall s_f \in S(f = 0,1,...,l-1)$ the following property takes place: $R(s_f) = s_{f+1}$. Starting from $s_l \in S$, there is different relation – $R(s_l) = s_l$, $R(R(s_l)) = s_l, \ldots,$ required to satisfy the totality property of S.

To form the basis for FS synthesis, the concept of "trajectory" is applied: $L(b') = L(s_0), L(s_1), ..., L(s_l)$, where $b' = s_0, s_1, ..., s_l$, comparing to b, is a finite sequence. Corresponding elements are the elements of AP set:

$$AP = \bigcup_f L(s_f), \ f = 0,1,..., l, \tag{4}$$

where $AP = V \times D$, $V = \{v_j\}_{j=1}^{n \in N}$ – state variables set; $D = \{d_k\}_{k=1}^{z \in N}$ – set of state variable values.

In case of $m \in N$ trajectories specified, (4) can be generalized as follows:

$$AP = \bigcup_{i=1}^{m} AP_i, \tag{5}$$

where $m = |B'|$: $B' = \{b_i'\}$; AP_i – i-th set of atomic prepositions to specify $b_i' \in B'$ trajectory.

3.1.3 Events formalization

By applying the principle of dichotomy, $\forall s, s' \in S$: $s' = R(s)$, $ap \in L(s) \subset AP$ and $ap' \in L(s') \subset AP$ elements are approached as pre- and post-conditions to the events prompting the $(s, s') \in R$ transition between the adjacent states $s, s' \in S$: $L(s) \Delta L(s') = \{(v_j, d), (v_j, d')\}$, respectively, where $d, d' \in D, d \neq d'$, $(v_j, d) = ap \in L(s) \subset AP$ – precondition

to j-th event, $(v_j, d') = ap' \in L(s') \subset AP$ – post-condition.

An event is analytically represented as implication, modified with X (neXt) temporal operator [20]:

$$e_j \equiv ((v_j, d) \rightarrow X(v_j, d')) \equiv \equiv (\neg(v_j, d_k) \vee X(v_j, d_{k+1})), \quad (6)$$

where $\equiv$ operator represents the tautology.

Expression (6) means that at certain current simulation step, associated with $s \in S$ state, $(v_j, d) \in L(s) \setminus L(s') \subset AP$ atomic preposition takes place – equals "true": $(v_j, d) \equiv 1$. As an outcome of e_j event, $(v_j, d') \in L(s') \setminus L(s) \subset AP$ atomic preposition becomes "true" – $(v_j, d') \equiv 1$ – in the subsequent state $s' = R(s) \in S$, addressed with X operator.

If consider e_j as a function, then $L(s) \setminus (s') = \{(v_j, d)\} = dom(e_j)$, $L(s') \setminus (s) = \{(v_j, d')\} = ran(e_j)$. Thus, it can be approached as a relation:

$$e_j : L(s) \setminus (s') \rightarrow L(s') \setminus (s), \quad (7)$$

i.e. $e_j : (v_j, d) \mapsto (v_j, d')$.

Semantic property of X temporal operator is depicted below:

$$(M, s \models X(v_j, d')) \equiv (M, s' \models (v_j, d')), \quad (8)$$

where $(M, s \models X(v_j, d'))$ expression means that, with respect to current $s \in S$ state of transition system, formalized with M (2), $X(v_j, d')$ temporal formula is true; $\equiv$ operator depicts the tautology; $(M, s' \models (v_j, d'))$ expression means that, with respect to a subsequent state $s' = R(s) \in S$, (v_j, d') atomic preposition is true.

3.1.4 Model checking task generalization

To encompass both positive and negative outcomes of MC - in terms of errors presence/absence in the FS, expression (1) has been extended as a disjunction:

$$((M, b \models \psi) \vee (M, b \models \neq \psi)) \equiv 1, \quad (9)$$

which is a tautology. In case of $(M, b \models \psi) \equiv 1$, no errors, e.g., unreachable states, deadlocks, have been faced during the MC. Otherwise, $(M, b \models \neq \psi) \equiv 1$ equality takes place.

In terms of the expediency of MC technique application, both disjoints in (9) represent successful scenarios. To make this statement relevant, both of the following questions need to be positively answered first:

a) Is implementation of ψ temporal formula - FS - correct?

b) Is the model generated through the MC adequate?

An approach to answering these questions is provided below.

3.1.5 Setting the relations between the concepts of both planes

To foster answering the first question, the transparent and unambiguous "bridge" between the analytical representation of ψ and its implementation on the basis of specified formalism need to be constructed (Figure-1). Named formalism is devoted to be an instrument making it possible to check the correctness of the FS in an automated manner.

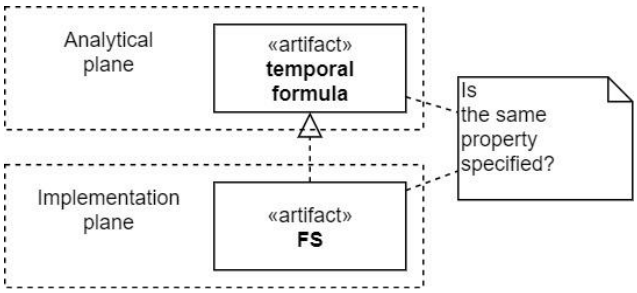

Figure-1. Differentiation between the analytical and implementation planes.

In Figure-1, the external dashed squares represent the conceptual planes discussed.

The correctness of the FS is approached in a bipartite manner - syntactic correctness and the structural one:

a) Syntactic correctness is addressed with syntactic analyzer.

b) Structural correctness should be delivered through the approach described.

To check the adequacy, the concept of "MSA" (Model Structural Adequacy) has been applied [21]. Peculiarities of the approach proposed are depicted in Figure-2.

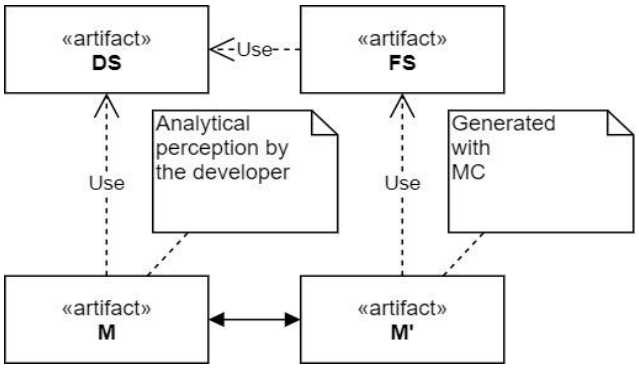

Figure-2. Relations between the DS, FS, M and M' artifacts.

With respect to Figure-2, developer's perception of the FS is addressed with M structure (2). At the same time, the FS, synthesized from the DS, is supposed to be the grounding, the M' transition system is generated from in an automated manner. In accordance with this, proposed approach is aimed at checking the adequacy of M' artifact in a topological manner: M and M' artifacts are treated as the graphs to be compared to each other with respect to the metrics chosen. Moreover, positive decision on the adequacy of resulting M' structure is also supposed to be the implicit demonstration of FS correctness.

3.1.6 Algorithm inside the proposed technique

To solve the MC task (9), corresponding algorithm has been proposed (Figure-3).

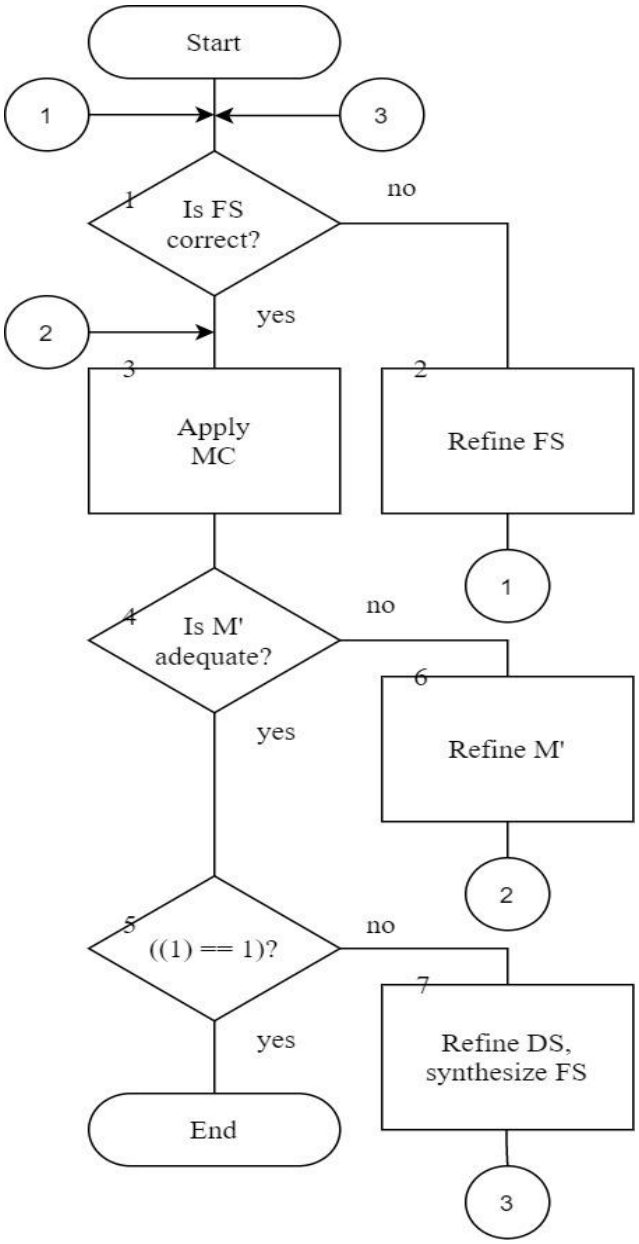

Figure-3. Conceptual view of the approach to model checking task resolving.

With respect to Figure-3, correctness of the FS is supposed to be checked first (block 1) in accordance with bipartite manner described above (Figure-1). As a result of positive outcome, the MC technique is supposed to be applied (block 3), otherwise - the FS needs to be refined (block 2). Peculiarity of the approach applied: M' adequacy checking is performed after the MC (block 4). Otherwise, M' still would not be generated. In case of M' inadequacy (block 6), it has to be refined. Otherwise, the truth of $(M, b \models \psi)$ statement is checked (block 5). In case of $(M, b \not\models \psi) \equiv 1$, DS refinement and subsequent synthesis of corresponding FS need to be performed (block 7).

3.2 Implementation Plane Formalization

3.2.1 Events implementation

Actions are the elementary constructs of TLA+ specification [7]. To specify the actions, the events – (6) and (7) - need to be implemented first. Then these implementations are approached as action constituents. To do so, the following steps have been made.

Step 1. Initial state specification. Correspondence between the notions of "state label" - $L(s),\ s \in S$ (2) - and state specification - φ - taking place in the implementation plane has been established:

$$\varphi : L(s) \to \{0,1\}, \tag{10}$$

where $\{0,1\} = ran(\varphi)$ – Boolean domain: 0 - "false", 1 - "true".

To start model checking, the initial state specification is approached as a conjunction over the elements of $L(s_0),\ s_0 \in S_0 \subset S$ (2):

$$\varphi(L(s_0)) = (v_1,0) \wedge (v_2,0) \wedge ... \wedge (v_n,0), \tag{11}$$

where $(v_j,0) \in L(s_0) \subset AP$ [22].

Step 2. Event specification. To implement the analytical concept of event (6) on the basis of TLA+ formalism, an "if-then-else" construct has been utilized [23]:

$$\begin{matrix} e'_j \equiv if\left(\varphi(L(s_0))\right) then\left(v'_j := d'\right) \\ else\left(v'_j := d\right) \end{matrix}, \tag{12}$$

where e'_j - TLA+ implementation of e_j (6); $\varphi(L(s_0))$ - precondition for the event to take place; $v'_j \in V'$ - copy of $v_j \in V$ applied with respect to the subsequent state (8) - as an outcome of X temporal operator utilization. As $s : v_j \mapsto d$, then, in case of $\varphi \equiv 1$, $s' : v'_j \mapsto d'$, where $s' = R(s)$. Depending on $|D|$ value, $(v'_j := d')$ operation can be performed up to $z-1$ times, i.e., e'_j event can take place up to $z-1$ times (4). On contrary, in case of $\varphi \equiv 0$, $(v'_j := d)$ expression means that $v_j \in V$ does not change its value during $(s,s') \in R$ transition. Here comes the necessity to formalize an "empty event" concept, similar to (6):

$$\begin{matrix} e^0_j \equiv \left((v_j,d) \to X(v_j,d)\right) \equiv \\ \equiv \left(\neg(v_j,d) \vee X(v_j,d)\right) \end{matrix}, \tag{13}$$

where; $e^0{}_j$ - analytical interpretation of j-th empty event. Comparing to (6), $v_j \in V$ does not change its value with X operator application: $s' : v'_j \mapsto d$.

Concept of an empty event (13) is based on Hoare's empty statement rule [24]:

$$\{(v_j,d)\}\ skip\ \{(v_j,d)\}, \tag{14}$$

where *skip* statement does not change the state of (2) automaton.

Thus, depending on φ value, either event (6) or empty event (13) takes place. Here comes generalization property of event implementation (12) (Figure-4).

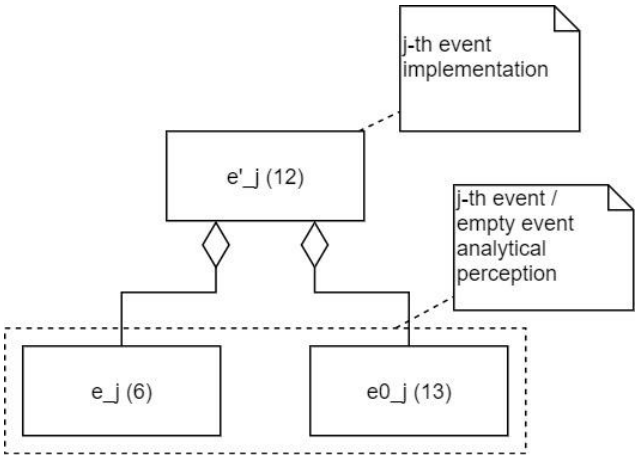

Figure-4. Relations between the concepts of (6), (12) and (13).

In Figure-4, generalization property of (12) is depicted with "aggregation" relation.

Similar to (12), preconditions to the subsequent events are obtained in a like manner.

3.2.2 Actions specification

Actions are the distinctive concepts of TLA. To specify an action, $\forall v'_j \in V'$, $s'(v'_j)$ value needs to be assigned. Thus, total number of events / empty events to be specified to construct an action equals $|V'| = n$.

To implement empty event (13) on its own, the "unchanged" operator (u) of TLA+ formalism has been applied:

$$u:(v_j, s(v_j)) \mapsto (v'_j, s(v_j)), \quad (15)$$

To shift from the concept of "event" to TLA-concept of "action" - function specifying the transition between the adjacent states [7], events / empty events are formalized per each $v_j \in V$. To group these specifications together, conjunction operator has been applied:

$$\begin{aligned} a_j \equiv\ & u(v_1, s(v_1)) \wedge u(v_2, s(v_2)) \wedge \ldots \\ & \ldots \wedge u(v_{j-1}, s(v_{j-1})) \wedge \\ & \wedge e'_j \wedge u(v_{j+1}, s(v_{j+1})) \wedge \ldots \\ & \ldots \wedge u(v_n, s(v_n)), \end{aligned} \quad (16)$$

where a_j - j-th action built upon single e_j event and $n-1$ empty events.

With respect to (16), depending on the number of non-empty events, proportions between (12) and (15) constituents can vary.

Architecture, demonstrating relations between the concepts of analytical and implementation planes, including actions, is depicted in Figure-5.

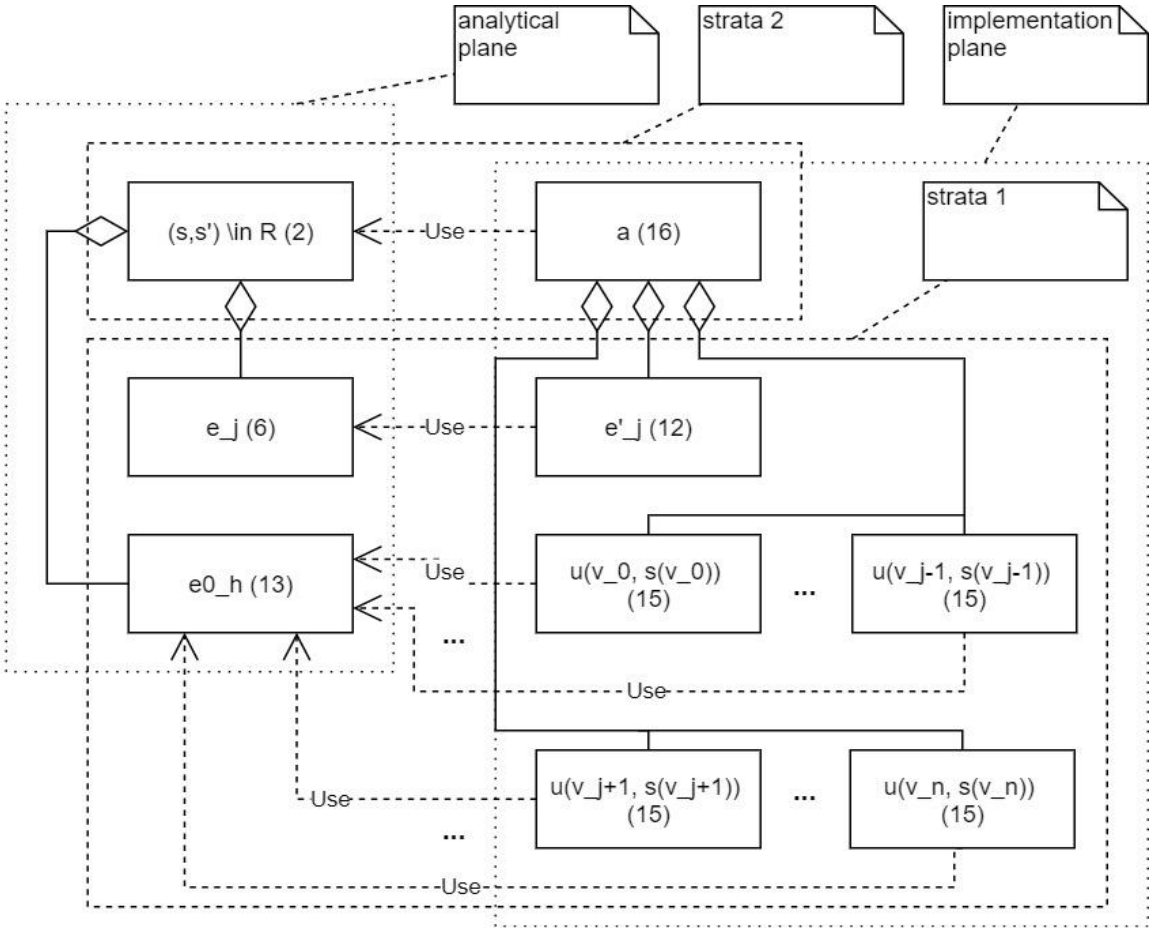

Figure-5. Concepts stratification.

In Figure-5, $h = 1,2,\ldots,j-1,j+1,\ldots,n; h \neq j$. Diagram is intended to be applied as a template to shift between the concepts of analytical and implementation planes, delimited with dotted squares. Moreover, two hierarchical layers (stratas) have been distinguished. Upper strata - 2 - encompasses solely a pair of elements: $(s, s') \in R$ - from analytical plane (2), (16) - from implementation plane. Stratification has been applied to emphasize the conceptual breach between the "events" and the "actions". Events here are "gluing" constituents - elementary building blocks to construct the resulting temporal formula ψ (1) to be checked on the basis of (2).

So far, the specified concept of "action" (16) does not provide the way to reason on system behavior b (1). To do so, further stratification needs to be conducted to encompass also the behavior [25].

3.2.3 Behavior specification

Behavior specification is constructed with respect to a template provided in [7]. To this end, G (Globally) temporal operator has also been used:

$$\psi \equiv \varphi(L(s_0)) \wedge G\begin{bmatrix} a'_0 \wedge Xa'_1 \wedge \\ \wedge X^2 a'_2 \wedge ... \\ ... \wedge X^f a'_f \wedge ... \\ ... \wedge X^{l-1} a'_{l-1} \end{bmatrix}, \quad (17)$$

where l actions prompt corresponding transitions between $l+1$ states - elements of b'; upper index of X^f expression means that X temporal operator has been applied f times, e.g., $X^2 a'_2 \equiv XXa'_2$, etc. Upper dash of a'_f means that corresponding action has been re-enumerated with respect to the sequence, set with X operators. Thus, in (17), implementation of behavior is encapsulated:

$$b'' \equiv a'_0 \wedge Xa'_1 \wedge X^2 a'_2 \wedge ... \\ ... \wedge X^f a'_f \wedge ... \wedge X^{l-1} a'_{l-1}, \quad (18)$$

where b'' - TLA+ implementation of b' (Figure-6).

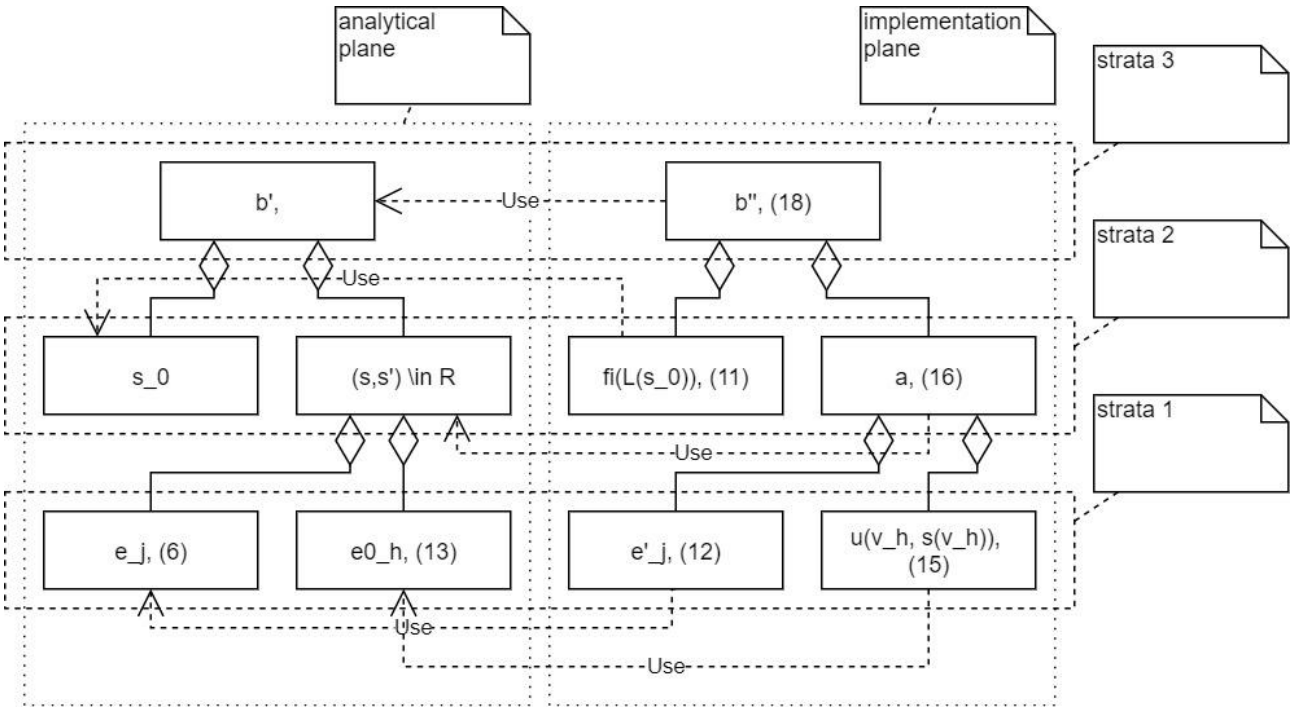

Figure-6. Resulting stratification of the FS.

In Figure-6, three hierarchical layers have been distinguished, where upper strata covers the concepts of b' and corresponding implementation b'' (18).

To generalize (17) for the case of multiple behaviors, with respect to (5), ψ can be rewritten as follows:

$$\psi \equiv \varphi(L(s_0)) \wedge G[b''_1 \vee b''_2 \vee ... \vee b''_m], \quad (19)$$

where $\vee$ operator defines the alternativeness of behaviors. The diagram, depicted in Figure 6, and expression (19) provide the mechanism to obtain the resulting temporal formula ψ (1) on the basis of Kripke structure (2).

3.3 Adequacy Checking

To prove the proposed technique, M and M' structures have been stepped up from the topological viewpoint: corresponding transition systems have been considered as graphs to be compared:

a) Let $G = \langle S, R \rangle$ is a structural constituent of M artifact (Figure-2), where S - set of vertices, R - set of edges between the elements of S; G represents developer's analytical perception of the DS.

b) 2. Let $G' = \langle S', R' \rangle$ is a structural constituent of M' artifact (Figure 2), where S' – set of vertices, R' - set of edges between the elements of S'; G' is synthesized in an automated manner through the model checking with respect to Figure-3. Its metrics are obtained from the listing of model checking results.

c) 3. Statement on FS adequacy is made on the basis of the following conditions: $|S| = |S'|$ - numbers of distinct states found are equal; $depth(G) = depth(G')$ - depths of space search are equal. In case of both these conditions are satisfied, the FS is considered as an adequate one.

Applicability of the proposed technique has been proven through the case study provided below.

4. RESULTS AND DISCUSSIONS

4.1 Technique Peculiarities Demonstration

To demonstrate the peculiarities of the proposed technique, a case study has been conducted: fragment of a flowchart of spacecraft orientation control unit has been considered (Figure-7):

a) Control equipment (CE) status flag predefines the forthcoming scenario (Block 1).

b) In case of 0, power delivery system (PDS) needs to be started (Block 3).

c) Otherwise, configuration control unit (CCU) should be started first (Block 2).

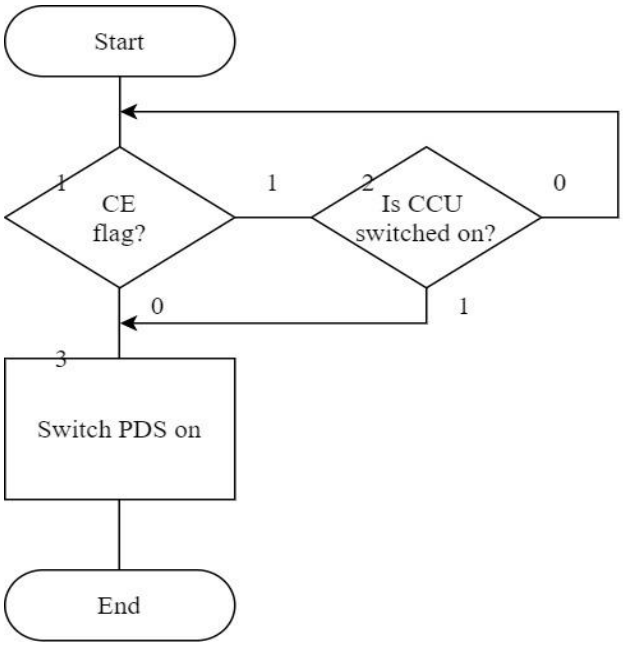

Figure-7. Fragment of the flowchart.

In accordance with Figure 7, the following scenarios can be contemplated (depicted as the sequences of block numbers): 1, 3; 1, (2, 1), 3.

To create corresponding FS, the V, D and the resulting AP sets need to be synthesized first (4): $V = \{v_1, v_2, v_3\}$, where $v_j \in V\ (j = 1,2,3)$ represents j-th block; $D = \{0,1\}$, where $d \in D$ is either "true" or "false"; $AP = \{(v_j,0)\} \cup \{(v_j,1)\}$: $(v_j,0) \in AP$ – fragment of code, represented with j-th block, yet to be executed, $(v_j,1) \in AP$ - has already been executed.

With respect to Figure-7, depending on the value of CE flag, the set of initial states is as follows: $S_0 = \{s_0, s_1\} \subset S$: $L(s_0) = \{(v_1,0),(v_2,0),(v_3,0)\}$, $L(s_1) = \{(v_1,1),(v_2,0),(v_3,0)\}$. Corresponding state diagram covering all the elements of S is provided in Figure-8.

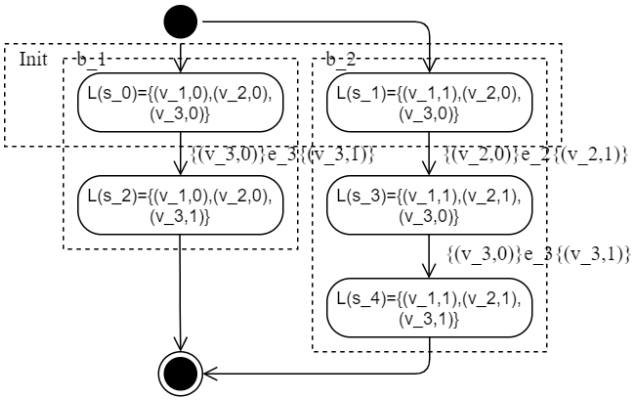

Figure-8. State diagram for the flowchart fragment.

In Figure-8, graph G is depicted: $|S| = 5$; depth of state space search - 3 - number of states the "longest" behavior is built of.

In Figure-8, there are two alternative behaviors – b_1, b_2: $b_1 = s_0, s_2$; $b_2 = s_1, s_3, s_4$, representing the scenarios of system functioning. Dashed "Init"-square encompasses the elements of $S_0 \subset S: |S_0| = 2$, with L function applied. Transitions between the states are marked with Hoare triples: $(s_0, s_2) \in R: L(s_0)\Delta(s_2) = \{(v_3,0),(v_3,1)\}$ - with $\{(v_3,0)\}e_3\{(v_3,1)\}$ triple, where e_3 – the event prompting the transition: $(v_3,0) \in AP$ - precondition for e_3 event, $(v_3,1) \in AP$ - post-condition; the same triple takes place for $(s_3, s_4) \in R: L(s_3)\Delta(s_4) = \{(v_3,0),(v_3,1)\}$ transition. With respect to $(s_1, s_3) \in R: L(s_1)\Delta(s_3) = \{(v_2,0),(v_2,1)\}$ transition, $\{(v_2,0)\}e_2\{(v_2,1)\}$ triple is applied. Thus, there are two events - e_2 and e_3, prompting the transitions between the states.

Thus, specification of b_1 is as follows: $\psi_1 \equiv \varphi(L(s_0)) \wedge G[a_3]$. With respect to (12) and (16), $a_3 \equiv (if((v_1 = 0) \wedge (v_3 = 0))then(v_3' := 1 - v_3)else(v_3' := v_3)) \wedge u(v_1, s(v_1)) \wedge u(v_2, s(v_2))$.

Specification of b_2 is as follows: $\psi_2 \equiv \varphi(L(s_1)) \wedge G[a_2 \wedge Xa_3]$. Basing on (12) and (16), $a_2 \equiv (if((v_1 = 1) \wedge (v_2 = 0))then(v_2' := 1 - v_2)else(v_2' := v_2)) \wedge u(v_1, s(v_1)) \wedge u(v_3, s(v_3))$.

The resulting TLA+ FS, encompassing both behaviors (b_1 and b_2), is a disjunction: $\psi \equiv \psi_1 \vee \psi_2 \equiv (\varphi(L(s_0)) \vee \varphi(L(s_1))) \wedge G[a_3 \vee (a_2 \wedge Xa_3)]$. Thus, with respect to (1), the model checking task can be formulated as follows: $M, b_1, b_2 \models \psi_1 \vee \psi_2$.

Complete specification is provided in an Appendix A, where "Spec" is the TLA+ representation resulting temporal formula ψ. As an outcome of TLC model checker application to the FS, the metrics of G' graph are provided in corresponding listing: no deadlocks have been reached, total number of distinct states found - 5, the depth of state space search - 3.

With respect to the results obtained, by comparing the metrics of G and G' graphs, the resulting M' model can be characterized as an adequate one. The correctness of the FS has been checked in an automated manner by applying the built-in instruments of TLA Toolbox. To estimate corresponding time costs, two alternative implementation of TLC model checker have been considered: the one, relying on the Breadth-first Search (BFS) technique, and the implementation based on the Depth-first Search (DFS) technique.

An experiment has been conducted on the following platform: TLC checker version - 2.14; Java Runtime Environment (JRE) version - 64 bit, build 1.8.0_251-b08; 8 threads, 3.8 GHz; random access memory capacity: 16 GB.

Obtained results: $\overline{t_{BFS}} = 0.895$ sec, $\overline{t_{DFS}} = 0.355$ sec, where $\overline{t_{BFS}}$ - average value of BFS-related time costs, $\overline{t_{DFS}}$ - DFS-related one. Each value is an average of ten measures.

Thus, it can be concluded that, with respect to a scenario considered, the DFS-based implementation of TLC checker is about 2.52 times more efficient, comparing to the BFS-alternative, in terms of corresponding time costs.

4.2 Sophisticating the Case Study

To generalize the outcome of proposed technique implementation, more complex scenario has been considered: software component of the on-board digital computer complex (DCC) of a spacecraft has been approached. Corresponding algorithm is devoted to control an initial state of the registers of the on-board computer of the input/output unit of DCC:

a) Algorithm block-diagram has been considered as an input data.

b) Total number of state variables - 18.

c) Total number of blocks – 77 (Figure 9).

In Figure-9, dashed square Z depicts the construct repeating 15 times, but with different variables. Thus, the resulting number of blocks is as follows: $5 \cdot 15 + 2 = 77$.

Table-1. Spatial characteristics of synthesized and verified specifications.

No.	n	$\lvert S \rvert$	$\lvert S^*_{BFS} \rvert$	$\lvert S^*_{DFS} \rvert$	$\frac{\lvert S \rvert}{\lvert S^*_{BFS} \rvert}$	$\frac{\lvert S \rvert}{\lvert S^*_{DFS} \rvert}$	$\frac{\lvert S^*_{DFS} \rvert}{\lvert S^*_{BFS} \rvert}$	$depth$, **vertices**
1	4	82	130	700	0,631	0,117	5,385	10
2	6	342	534	4519	0,640	0,076	8,463	15
3	7	702	1086	11189	0,646	0,063	10,303	18
4	7	712	1096	11927	0,650	0,060	10,882	19
5	8	1432	2200	28156	0,651	0,051	12,798	22
6	10	5742	8814	141343	0,651	0,041	16,036	27
7	11	11502	17646	317237	0,652	0,036	17,978	30
8	11	11512	17656	328775	0,652	0,035	18,621	31
9	12	23032	35320	726652	0,652	0,032	20,573	34
10	14	92142	141294	3367327	0,652	0,027	23,832	39
11	15	184302	282606	7287605	0,652	0,025	25,787	42
12	15	184312	282616	7471943	0,652	0,025	26,438	43
13	16	368632	565240	16049788	0,652	0,023	28,395	46
14	17	737272	1130488	34311398	0,652	0,021	30,351	49
15	18	1474552	2260984	73046458	0,652	0,020	32,307	52

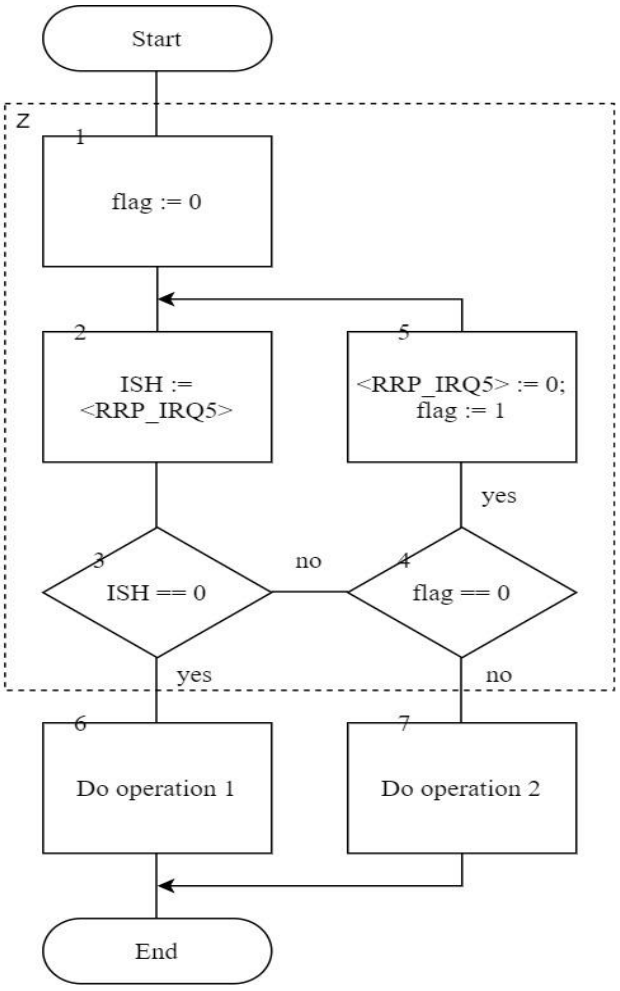

Figure-9. Fragment of algorithm block-diagram.

In Table-1, No - the number of dashed blocks (Figure 9), n - number of state variables, $|S|$ - total number of distinct states found during the model checking, $|S^*_{BFS}|$ - number of sates generated through BFS-search, $|S^*_{DFS}|$ - through DFS-search. It can be seen that spatial characteristics of BFS-implementation of the TLC method are significantly better, comparing to the DFS-alternative: up to 32 times.

No deadlocks have been reached during the verification. With respect to a proposed technique, the metrics of G and G' graphs have been found to be equal.

5. CONCLUSIONS

Within this paper, we have proposed the technique for checking the adequacy of model, the model checking method is devoted to be applied to.

With respect to the proposed technique, the concepts of analytical and implementation planes have been distinguished. The technique is based on metrics comparison between the analytical and implementation planes - total number of distinct states, depth of state space search.

The following results have been obtained:

a) The fragment of spacecraft orientation control unit functioning flowchart has been considered as a case study. Corresponding FS has been synthesized with respect to the proposed technique. To implement the FS, the TLA+ formalism of TLA temporal logic has been applied. The correctness of the FS has been checked through TLA Toolbox. To verify the resulting FS, the TLC model checker has been utilized. With respect to the proposed technique, the adequacy of formal model has been successfully proved: total number of distinct states found - 5, depth of state space search - 3. No deadlocks have been faced.

b) Two alternative implementations of TLC model checker have been applied - the BFS- and the DFS-based one. It has been shown that, with respect to the case study conducted, the DFS-based implementation is about 2,52 times more efficient (in terms of corresponding time costs), comparing to the BFS-based alternative. It has been demonstrated that spatial characteristics of BFS-implementation are significantly better: up to 32 times - with respect to a case study conducted.

Further research is aimed at developing the mechanisms automating the implementation of the proposed technique.

ACKNOWLEDGMENT

Paper has been prepared with respect to the tasks of the research works carried out by the Department of Mathematical and Computer Modeling of G.E. Pukhov Institute for Modelling in Energy Engineering (PIMEE), NAS of Ukraine: 0121U110615 “Development of methods and means for safety-critical systems designing process artifacts verification”; 0120U102683 “Development of specialized computer technologies for modeling and processing of operational information in energy problems”.

The work has also been supported by Metal Centre Čakovec under the project KK.01.1.1.02.0023.

REFERENCES

[1] Verhulst E., Boute R. T., Faria J. M. S., Sputh B. H. C. and Mezhuyev V. 2011. Formal development of a network-centric RTOS: software engineering for reliable embedded systems. Springer Publishing Company, Inc.

[2] Beers R. 2008. Pre-RTL formal verification: an Intel experience. In 45th annual Conference on Design

Automation (DAC '08), Anaheim, California. pp. 806-811. DOI: https://doi.org/10.1145/1391469.1391675

[3] Pakonen A., Tahvonen T., Hartikainen M. and Pihlanko M. 2017. Practical applications of model checking in the Finnish nuclear industry. In 10th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies, San Francisco, CA, USA. pp. 1342-1352.

[4] Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. and Smirnov O. A. 2020. Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources. Journal of Theoretical and Applied Information Technology. 98(21): 3334-3346.

[5] Newcombe C., Rath T., Zhang F., Munteanu B., Brooker M. and Deardeuff M. 2015. How Amazon web services uses formal methods. Communications of the ACM. 58(4): 66-73. DOI: https://doi.org/10.1145/2699417

[6] Broy M. A 2013. A logical approach to systems engineering artifacts and traceability: from requirements to functional and architectural views. In Engineering dependable software systems: NATO Science for Peace and Security Series - D: Information and Communication Security, M. Broy, D. Peled, G. Kalus, Ed., Amsterdam: IOS Press. 34: 1-48. DOI: https://doi.org/10.3233/978-1-61499-207-3-1.

[7] Lamport L. 2002. Specifying systems: the TLA+ language and tools for hardware and software engineers. Boston: Addison-Wesley.

[8] Lamport L. 2006. Checking a multithreaded algorithm with +CAL. In 20th international conference on distributed computing (DISC'06), Stockholm, Sweden, September 18-20. pp. 151-163.

[9] Resch S. and Paulitsch M. 2017. Using TLA+ in the development of a safety-critical fault-tolerant middleware. In 2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW'2017), Toulouse, France. DOI: https://doi.org/10.1109/ISSREW.2017.43

[10] Kim Y.-M. and Kang M. 2020. Formal verification of SDN-based firewalls by using TLA+. IEEE Access. 8: 52100-52112.

[11] Konnov I., Kukovec J. and Tran T.-H. 2019. TLA+ model checking made symbolic. Proceedings of the ACM on Programming Languages. 3(OOPSLA): 1-30.

[12] Kuppe M. A., Lamport L. and Ricketts D. 2019. The TLA+ Toolbox. In 5th Workshop on Formal Integrated Development Environment (F-IDE 2019), Porto, Portugal. pp. 50-62.

[13] Babai L. 2016. Graph isomorphism in quasipolynomial time. In Forty-eighth annual ACM symposium on Theory of Computing (STOC '16), Cambridge, MA, USA. pp. 684-697.

[14] Heimdahl M. P. E., George D. and Weber R. 2004. Specification test coverage adequacy criteria = specification test generation inadequacy criteria. In Eighth IEEE International Symposium on High Assurance Systems Engineering. pp. 178-186, DOI: https://doi.org/10.1109/HASE.2004.1281742

[15] Pakonen A. and Buzhinsky I. 2019. Verification of fault tolerant safety I&C systems using model checking. In 2019 IEEE International Conference on Industrial Technology, ICIT 2019, Melbourne, Australia. pp. 969-974. DOI: https://doi.org/10.1109/ICIT.2019.8755014

[16] Gay G., Rajan A., Staats M., Whalen M. and Heimdahl M. P. E. 2016. The effect of program and model structure on the effectiveness of MC/DC test adequacy coverage. ACM Transactions on Software Engineering and Methodology. 25(3). DOI: https://doi.org/10.1145/2934672

[17] Cohen E., Dahlweid M., Hillebrand M., Leinenbach D., Moskal M., Santen T. and Schulte W. 2009. VCC: A Practical System for Verifying Concurrent C. In Berghofer S., Nipkow T., Urban C., Wenzel M. (eds) Theorem Proving in Higher Order Logics. TPHOLs 2009. Lecture Notes in Computer Science, vol. 5674. Springer, Berlin, Heidelberg. pp. 23-42. DOI: https://doi.org/10.1007/978-3-642-03359-9_2

[18] Fucci D., Erdogmus H., Turhan B., Oivo M. and Juristo N. 2017. A Dissection of the Test-Driven Development Process: Does It Really Matter to Test-First or to Test-Last? IEEE Transactions on Software Engineering. 43(7): 597-614. DOI: https://doi.org/10.1109/TSE.2016.2616877

[19] Clarke E. M., Grumberg O., Kroening D., Peled D. and Veith H. 2018. Model checking, 2nd ed. Massachusetts: The MIT Press.

[20] Shkarupylo V. and Polska O. 2018. The approach to SDN network topology verification on a basis of Temporal Logic of Actions. In 14th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET'2018). pp. 183-186.

[21] Gupta H. V., Clark M. P., Vrugt J. A., Abramowitz G. and Ye M. 2012. Towards a comprehensive assessment of model structural adequacy. Water Resources Research. 48(8): 1-16. DOI: https://doi.org/10.1029/2011WR011044

[22] Shkarupylo V. V., Tomičić I. and Kasian K. M. 2016. The investigation of TLC model checker properties. Journal of Information and Organizational Sciences. 40(1): 145-152.

[23] Shkarupylo V., Kudermetov R., Golub T., Polska O. and Tiahunova M. 2018. Towards Model Checking of the Internet of Things Solutions Interoperability. In 2018 IEEE International Scientific and Practical Conference on Problems of Infocommunications. Science and Technology (PIC S&T-2018), Kharkiv, Ukraine. pp. 465-468.

[24] Hoare C. A. R. 1969. An axiomatic basis for computer programming. Communications of the ACM. 12(10): 576-583.

[25] Alsayaydeh J. A. J., Shkarupylo V., Hamid M. S. B., Skrupsky S. and Oliinyk A. 2018. Stratified model of the Internet of Things infrastructure. Journal of Engineering and Applied Sciences. 13(20): 8634-8638.