Behavioral Agent Testing of Distributed Information Systems

Oleksandr Martynyuk[a], Oleksandr Drozd[a], Hanna Stepova[a], Viktor Antonyuk[a], Dmitry Martynyuk[b]

[a] *Odessa National Polytechnic University, Ave. Shevchenko 1, 65044 Odessa, Ukraine*
[b] *Join Venture "Nippon Auto", Academician Williams str. 71a, Odessa, Ukraine*

Abstract

The use of multi-agent technologies in modern distributed information systems, that due to the increasingly demanded properties of autonomy, mobility, intelligence, cooperation, determines their appearance in the tools of online and offline testing. This work presents an agent-based check model of behavioral testing of a component of a distributed information system in its resource environment. This agent check model has the features of the composition of the reactive online and deliberative offline components of testing, suggesting deterministic and evolutionary methods of decomposition behavioral testing. To ensure autonomy and mobility, the agent-based check model defines the resource model of the component deployment environment, models of its goals and test strategies, signatures of multi-agent operations, as part of observation, strategy execution and adaptation, as well as initial models, goals and test strategies of the component. For the intelligence of the agent-based check model, its main components are also defined as individuals in the population of evolutionary test generation, which is proposed as an implementation of the deliberative component testing strategy. For the cooperation of agent-based check models in the network and multi-level interaction of their reactive and deliberative components, there are cooperative definitions for resource models of placement, goals and strategies, operations and initializations, realizability and transportability into the network DIS model, inheritance into the hierarchical DIS model. The check model defines the formal conditions/requirements for the performance of the behavioral online and offline testing, that performed by the multi-agent system of testing, and can be taken as the basic one, when constructing methods and special distributed behavioral test systems.

Keywords:

Distributed information system, online and offline testing, behavioral testing, agent, multi-agent system, evolutionary check model

1. Introduction

Modern effective distributed information systems (DIS) [1, 2] are based on the joint use of many promising computer, communication, computing, information subject technologies, and some of which are rapidly developing service-oriented architectures, as Web services [3, 4], cluster, cloud, multi-agent [5, 6]and GRID systems [7, 8], super-large data and knowledge systems [9, 10] intelligent systems [11-13], the avalanche-growing world of the Internet of things [14, 15]. One of the most important reasons for this growth is the significant successes of fundamental mathematical computer sciences, in particular, in the field of complex, decomposition, functional, fuzzy, intellectual, concurrent methods of analysis and synthesis of DIS.

ICT&ES-2020: Information-Communication Technologies & Embedded Systems, November 12, 2020, Mykolaiv, Ukraine
EMAIL: anmartynyuk@ukr.net (O. Martynyuk); drozd@ukr.net (O. Drozd); hanna.suhak@gmail.com (H. Stepova); viktor.v.antoniuk@gmail.com (V. Antonyuk); domarty@ukr.net (D. Martynuk)
ORCID: 0000-0003-1461-2000 (O. Martynyuk); 0000-0003-2191-6758 (O. Drozd); 0000-0002-7223-8822 (H. Stepova); 0000-0003-0427-9005 (V. Antonyuk); 0000-0001-9267-1474 (D. Martynuk)

At the same time, the quality of their functioning, which is determined not only by the general completeness, accuracy and efficiency of the results, but also their reliability, and the performance of the DIS as a whole [2]. One of the most important ways to ensure the operability of computer systems is the technology of technical verification of projects, testing and diagnosing of DIS implementations, which allows developing and applying various tools, as part of automated technical diagnostic systems (ASTD) [16]. Among these tools, as a rule, there is an online and offline testing. Currently, there are developed tools of structural, functional-behavioral, deterministic, pseudo-random, evolutionary-genetic, logical, symbolic, fuzzy, intelligent, abstract, decomposition analysis and synthesis to provide the necessary level of DIS reliability of projects and the operability of functioning of its implementations, in particular, the tools of online and offline testing of DIS.

Most often, to achieve greater efficiency, these tools are used in a comprehensive manner, tuning their aggregates to the features of specific tested systems and the tasks they solve. The properties of the DIS, that adopted from the applied technologies, are reflected also in the properties of ASTD tools. Moreover, the complexity of constructing technical tools ASTD often exceeds the complexity of the DIS [15-19] itself. It is often and naturally, for example, for systems of critical applications [20-22].

At the same time, there is transfer of the basic methods of DIS analysis and synthesis to a systemic, abstract-symbolic, structural-behavioral level, which has mappings to many possible technologies of physical implementation.

Additionally, the degree of dimensionality, uncertainty and intelligence of DIS behavior, of scalability, autonomy, mobility and situational dynamic the goal formation and the cooperation of DIS components for solving objective problems in a multi-level, multi-platform global network is increasing. This increasing degree requires appropriate improvement of verification and test tools, in particular, in the abstract, behavioral, decomposition, intelligent directions.

Thus, we can conclude, that the development of effective tools for ASTD new DIS is non-trivial and relevant.

2. Review of the known approaches

Formal behavioral analysis and synthesis of computer systems (CS), usually characterized by NP-complexity [15-19], is becoming increasingly important in systemic, structural and functional verification of projects of CS and check of its implementation.

The ongoing development of the mathematical theory of experiments with automata class models [23, 24], which define a formal methodology of behavioral checking and recognition, gradually raises the upper limits of applicability of check automata models.

Generally, these limits are determined by exponential (NP-complex) dependences of the complexity of such an analysis from the dimension of input models [15-19].

To reduce the complexity, in particular, the time of preparation and execution of the behavioral testing in the class of errors of functional automata mappings with preserved values of the completeness of the testing itself, a common (mathematical) and special (taking into account architectural solutions) network and hierarchical decomposition are used [25-29] for checked systems most often.

They allow to reduce (usually polynomially) the exponential dependence of the dimension of complexity (computational resources) and time of constructing and performing of testing at a formal model level. Special probabilistic-statistical [30-32], fuzzy [33-36], intelligent [37, 38], in particular, evolutionary-genetic [25, 28, 29, 38] models and methods give experimental values for the complexity of behavioral testing for most real systems, that polynomially smaller, than the upper theoretical boundaries of deterministic methods, with a slight decrease in the completeness of this testing.

As a result, depending on the involved computing tools, behavioral check of acceptable completeness of checking for DIS of medium complexity becomes achievable, nevertheless, as noted, such a decrease does not remove this problem from the class of NP-complex ones.

The above circumstances determine the expediency of further development of verification and testing of CS, in particular, complex, formal, decomposition, intelligent, parallel check models and methods with the total effect of reducing the complexity of the computational costs of its preparation and implementation.

The growing tendency to classify a significant part of modern DIS to real-time and critical application systems, significantly strengthens the requirements for completeness, accuracy and relevance of operational checking of their performance, for resources, that necessary for its implementation.

This trend also encourages the use of complex ASTD, which include tools of online and offline testing, acting from the upper systemic, structural and functional level. In this case, the work of ASTD is inevitably based on formal models and methods of behavioral testing [24, 28, 29, 39-41], whether their analytical or intuitive expert implementations.

Behavioral testing of the CS makes it possible both for the online testing [29], that is backgrounding to the main functioning and performing the accumulation of check presentability of passive recognition experiments, as well as the special offline testing, that is interrupting the main functioning of the CS and performing active check experiments [28]. Typically, both modes involve a previous (preprocessor) check analysis of the reference behavior of the CS with external or internal (own) procurement of elements of behavioral check.

The foregoing does not remove the advisability of continuing research on decomposition, compatible, dynamically integrated online and offline behavioral testing, especially for distributed, critical, intellectual CS, in particular, DIS, with dynamically reconfigurable of model representation, structure, composition and migration of components, general and component behavior.

These features, in particular, become characteristic of DIS and cause the expanding use of multi-agent technologies. For such systems, a distributed, fuzzy, dynamic definition of the class of checked properties, completeness, accuracy and relevance of the checking are also characteristic. The consequence of this condition, in particular, can be considered the emergence of the class of ASTD DIS, essentially relying on models and methods of multi-agent technologies [42].

Taking into account the aforementioned, the tasks of system-functional, network and multi-level, integrated online and offline behavioral testing of DIS can be considered relevant and carried out. This testing executes on the basis of the dynamic cooperation of intelligent agents in the conditions of variable resources, that provided by the environment of the DIS itself. Also, this behavioral testing has variable check controllability, observability and heritability of own results, bases on agent, decomposition, deterministic and evolutionary models and methods.

3. Main researched tasks

A multilevel network model – a extended hierarchical Petri net, adopted as the basic input model of checking agent for DIS, – is characterized by a two-dimensional structure, alphabets and mappings of components, data and knowledge structures, that can change in the life cycle of a checked DIS, that hosts checking agents from MAS.

Online and offline testing of DIS is carried out by cooperations of agents from a set of Ag, that located into DIS environment. Atomic element, MAS agent $ag_i \in Ag$, is the MAS model of the first level.

The goal of this work is the construction an agent check model ag_i, as a formalization of the representation of agent online and offline behavioral testing, performed on the basis of the Petri net $S(f)_i$ – an automata class model for some DIS component.

These representations are selected in accordance with the contexts for MAC testing, namely: autonomous; mobile; purposeful or intellectual; cooperative.

To achieve this goal, the tasks of behavioral testing for PN $S(f)_i$ are formalized. These tasks are solved by the agent $ag_i \in Ag$ of a certain components of DIS and include: behavioral testing for an component Petri net $S(f)_i$; controllability and observability of testing for the component Petri net $S(f)_i$ into the network model $nS(f)_i$ of DIS; inheritance of the elements of behavioral testing of a single-level Petri net $S(f)_i$ into the hierarchical model $iS(f)_i$ of DIS.

4. Formal model of behavioral testing

The model of behavioral agent-based testing is built on the basis of an extended Petri net — an input automata class model [26, 28, 29] for some *i*-th component of DIS of the form:

$$S(f)_i=(P, T, Ev, Ac, X, Y, Ep, Et, Em, F, S, M, M_0), \quad (1)$$

where P, T – positions and transitions, respectively; Ev, Ac – events for positions and actions for transitions; $X \cup @ \subseteq Co$, $Y \cup @ \subseteq Fu$ – external controlled events and observed actions, supplemented by @ – an unknown event or action; Ep – the set of variables for the energy costs, represented as triples $ep_{nep}=(epi_{nep},ep^{\circ}_{nep},ept_{nep})$ with components – the upper values of the energy expenditures, calculated on the electricity epi_{nep}, temperature difference ep°_{nep}, time interval ept_{nep}; Et – variables for the energy costs, represented as the triples $et_{net}=(eti_{net},et^{\circ}_{net},ett_{net})$ with components - upper of values of the energy costs calculated on the electricity epi_{nep}, temperature difference ep°_{nep}, time interval ept_{nep}; Em – weighted mandates for forming events and performing actions, having the form of a triple $em_{nem}=(emi_{nem},em^{\circ}_{nem},emt_{nem})$ with components – cumulative values of energy costs, that are calculated on the electricity epi_{nep}, temperature difference ep°_{nep}, time interval ept_{nep}; $F:((B(P),S(B(P)))\rightarrow T)\cup((T\times Ac\times Et)\rightarrow B(P)))$ – the incidence relation for subsets of positions from the Boolean $\{p_1,\ldots,p_{ip}\}\in B(P)$ and transitions from T, that is depended from current events, actions and energy costs; $S:(P\rightarrow Ev\times Ep)\cup(T\rightarrow Ac\times Et)$ – the correspondence of own, internal variables of events, as well as actions, expanded by the metric of energy costs in positions and transitions, included in F; $M_0:P\rightarrow Em$ – the initial marking of positions, taking into account the initial energy costs of initialization, $M:P\times Em\rightarrow Em$ – function of the current marking of positions and transitions, taking into account the current energy costs, accumulated during the movement of chips.

The model ag_i of the behavioral agent-based testing, which contains two components for simulated Petri nets (1), the deterministic reactive ag_{Ri} and the evolutionary deliberative ag_{Di} [42] has the form:

$$ag_i=(ag_{Ri}, ag_{Di}, Ev_i, Ac_i, \Delta_i), \quad (2)$$

where ag_{Ri}, ag_{Di} – its reactive and deliberative components, that interact with each other in space and in time, they are functioning in the Petri net $S(f)_i$ – the behavioral model of some *i*-th component of DIS in its space-network $nS(f)_i$ [28] and temporal hierarchical $iS(f)_i$ [29] models; Ev_i, Ac_i – the events and actions of the components; $\Delta_i: \times_{i\in I}((Ac_{Ri}\rightarrow Ev_{Di})\cup(Ac_{Di}\rightarrow Ev_{Ri}))$ – the relation of the interaction for reactive ag_{Ri} and deliberative ag_{Di} components, both parallel (in space) and serial (in time).

The type, complexity and state of the Petri net $S(f)_i$ of the DIS and the multi-agent system (MAS) of testing, these states, as well as own targeting and tasks of agent ag_i affect to the form, complexity, priority, activation of ag_{Ri} and ag_{Di} components.

The difference between reactive ag_{Ri} and deliberative ag_{Di} of the check agent ag_i consists in the features of check tasks, models and methods, which are used into definition, constructing, and recognizing of the check objects.

In general, the agent model ag_i has information about the objects of check for reference checked properties Pr_i, identifiers Ci_i, check Cp_i and link Lp_i primitives, realized $S(f)_i(R^{Ac})$ and transported $S(f)_i(Tr^{Ev})$ Petri nets, realizing $T^{-1}(S(f)_i)$ and transporting $T(S(f)_i)$ nodal subnets, realized $S(f)_i(R^{X}{}_{T^{-1}(S(f)i)})$ and transported $S(f)_i(Tr^{Y}{}_{T(S(f)i)})$ nodal Petri nets necessary, inherited properties iPr_i , identifiers iCi_i, check iCp_i and iLp_i primitives, check fragments Cf_i, realized Cf_i^R and transported Cf_i^{Tr} check fragments, inherited check fragments iCf_i.

The intelligence of the check agent ag_i for its deliberative component ag_{Di} can be based on evolutionary approaches to check a type of check evolution [28]:

$$Ce_i=(Cf_i, Cp_i, Lp_i, Sg_{ti}, Cf_{fi}), \quad (3)$$

where: Cf_i, Cp_i, Lp_i – the sets of check fragments, check and linking primitives, Cf_{0i} – initial set of check fragments, $Cf_{0i}=Cp_i\subseteq Cf_i$, Cf_{fi} – final set of check fragments; $Sg_{tagDi}=\{\mu_{tagDi}, \kappa_{tagDi}, \phi_{tagDi}, \varphi_{tagDi},$

$\sigma_{tagDi}\}$ – the signature of operations and functions of check evolution, that contain special operations of mutation, crossover, immunity, fitness function, selection function, respectively.

In the case of the presentation of DIS not only as checked, but also as a resource environment for hosting the checking MAS, the second extended model of the check agent ag_i (also ag_{Ri} and ag_{Di}) [42] is also presented as a system:

$$ag_i=(cS_i, cM_i, Q_i, St_i, \{o_i, \sigma_i, \alpha_i\}, \{cs_{0i}, cm_{0i}, q_{0i}, st_{0i}\}), \quad (4)$$

where: – cS_i – agent check models for the i-th component of the placement in the DIS, which determine the check conditions; – cM_i – agent check methods for the i-th component of the placement in the DIS, which determine the implementation of check procedures; – Q_i – agent goals, for some $q_i \in Q_i$, defined as:

$$\begin{gathered} q_i:((cS_i \times cM_i)\times(cS_i \times cM_i)\to Q_i,\ q_i((cs_i,cm_i),(cs_i',cm_i'))= \\ =(k_{\varphi\upsilon i}((\varphi\upsilon_i(cs_i')-\varphi\upsilon_i(cs_i)),(\varphi\upsilon_i(cm_i')-\varphi\upsilon_i(cm_i))), \\ k_{\lambda\varepsilon i}((\lambda\varepsilon_i(cs_i')-\lambda\varepsilon_i(cs_i)),(\varphi\upsilon_i(cm_i')-\varphi\upsilon_i(cm_i))), \\ k_{\theta\upsilon i}((\theta\upsilon_i(cs_i')-\theta\upsilon_i(cs_i)),(\theta\upsilon_i(cm_i')-\theta\upsilon_i(cm_i))), \\ k_{\rho\varepsilon i}((\rho\varepsilon_i(cs_i')-\rho\varepsilon_i(cs_i)),(\rho\varepsilon_i(cm_i')-\rho\varepsilon_i(cm_i))), \\ k_{\tau\rho i}((\tau\rho_i(cs_i')-\tau\rho_i(cs_i)),(\tau\rho_i(cm_i')-\tau\rho_i(cm_i))), \\ k_{\iota\nu i}((\iota\nu_i(cs_i')-\iota\nu_i(cs_i)),(\iota\nu_i(cm_i')-\iota\nu_i(cm_i)))), \end{gathered} \quad (5)$$

where $k_{\varphi\upsilon i}$, $k_{\lambda\varepsilon i}$, $k_{\theta\upsilon i}$, $k_{\rho\varepsilon i}$, $k_{\tau\rho i}$, $k_{\iota\nu i}$, $\varphi\upsilon_i$, $\lambda\varepsilon_i$, $\theta\upsilon_i$, $\rho\varepsilon_i=(\varphi\upsilon_i^{\rho\varepsilon}, \lambda\varepsilon_i^{\rho\varepsilon}, \theta\upsilon_i^{\rho\varepsilon})$, $\tau\rho_i=(\varphi\upsilon_i^{\tau\rho}, \lambda\varepsilon_i^{\tau\rho}, \theta\upsilon_i^{\tau\rho})$, $\iota\upsilon_i=(\varphi\upsilon_i^{\iota\nu}, \lambda\varepsilon_i^{\iota\nu}, \theta\upsilon_i^{\iota\nu})$ – the weighted priority coefficients and determination functions, respectively, of completeness ϑ of verification, length Λ (complexity), multiplicity Θ, network realizability $P\varepsilon$ and transportability $T\rho$, inter-level inheritance $I\upsilon$ of check (moreover, the functions of realizability, transportability, inheritance combine each of three own functions of preservation of completeness, lengths, multiples of check), cs_i and cs_i', cm_i and cm_i' - the initial and assumed next agent models and check methods for the placement component, respectively;

- St_i – strategies for the agent's functioning – the agent's application of deterministic, pseudo-random, expert, evolutionary check methods in its life cycle, for some st_i defined as:

$$\begin{gathered} st_i:(\cup_{j\in J}(cs_i,cm_i)_j)\to(\cup_{j\in J}(cs_i,cm_i)_j), \\ (cs_i',cm_i')=st_i(cs_i,cm_i); \end{gathered} \quad (6)$$

- $\{o_i, \sigma_i, \alpha_i\}$ – accent-agent operations directly displayed in the composition of check models from cM_i, their signatures of operations and relations (5), in composition of check methods from cM_i, their signatures of operations and relations (6), accordance with the models: a) abstract $Sg_{S(f)i}=\{\alpha,\beta,\gamma\}$ - the signature of the check operations of identification, congruence, determination [24]; b) network realizing and transporting compositions $Sg_{nS(f)i}=\{@,\times,@',\#\}$ - the signature of the check compositions – sequential, parallel, with feedback, DeMorgan half-convolution [28, 29]; c) hierarchical inheriting compositions $Sg_{iS(f)i}=\{\chi^p,\chi^t\}$ - signature of check substitutions of macro-positions and macro-transitions [29]. Own agent operations include:
  1. observation

$$\begin{gathered} o_i:(\cup_{j\in J}(cs_i,cm_i)_j)\times V_i\to(\cup_{j\in J}(cs_i,cm_i)_j), \\ (cs_i,cm_i)=o_i((\cup_{j\in J}(cs_i,cm_i)_j),V_i) \end{gathered} \quad (7)$$

– with the definition of the current agent check model and method, if necessary, the construction of identifiers Ci_i and check primitives Cp_i in their composition, here $V_i=(en_i, ag_{Ri}, Conn_i)$ is the complete environment of agent ag_i, for which: 1) $en_i=(S(f)_i, nS(f)_i, iS(f)_i, Pl_i)$ – abstract, two-dimensional, network-hierarchical space of the agent world for the agent ag_i, provided by the i-th component of DIS; 2) $Conn_i$ – the placement relation (on functional level – usually the inclusion relation) between the check agent ag_i and the space en_i for the checked i-th component; 3) $Pl_i=\{pl_{subi}, pl_{ini}, pl_{outi}, pl_{ioi}, pl_{nodei}, pl_{subni}, pl_{upi}, pl_{midi}, pl_{lowi}\}$ – a set of basic,

including initial, placements of the agent ag_i of the basic types (input, output, input-output, nodal in the structure, sub-structure, senior, middle, junior in the hierarchy) in+ the space en_i;

2. the implementation of the strategy

$$\sigma_i: St_i \times V_i \to V_i,$$
$$V' = \sigma_i(st_i, V_i) \tag{8}$$

– with obtaining the modified environment $V_i' = (en_i', ag_i', Conn_i')$ (in its composition - modified spaces of the agent world en_i', agent ag_i', relations of placement $Conn_i'$) for the existing agent model cs_i and method cm_i of check, with definition (if necessary) of linking primitives Lp_i and inherited properties iPr_i, with constructing of check fragments Cf_i and determining for the set of achieved check fragments the values of completeness ϑ_i, length Λ_i, multiplicity Θ_i, realizability $P\varepsilon_i$, transportability $T\rho_i$, inheritance $I\upsilon_i$;

3. adaptation

$$\alpha_i = (\alpha_{smi}, \alpha_{sti}),$$
$$\alpha_{smi}: ((\cup_{j \in J}(cs_i, cm_i)_j) \times ((\cup_{j \in J}(cs_i, cm_i)_j) \times (\cup_{j \in J}(cs_i, cm_i)_j))) \to (\cup_{j \in J}(cs_i, cm_i)_j),$$
$$(cs_i', cm_i') = \alpha_{smi}((\cup_{j \in J}(cs_i, cm_i)_j), (cs_i, cm_i), (cs_i', cm_i')),$$
$$\alpha_{sti}: (St_i \times (\cup_{j \in J}(cs_i, cm_i)_j)(\cup_{j \in J}(cs_i, cm_i)_j)) \to St_i,$$
$$St_i' = \alpha_{sti}(St_i, (cs_i, cm_i), (cs_i', cm_i')) \tag{9}$$

– with fixing updated sets (bases) of agent check models cS_i, methods cM_i and strategies St_i, consisting of: 1) identifiers Ci_i and check primitives Cp_i; 2) check fragments Cf_i; 3) output realizable $S(f)(R^{Ac})$ and input transportable $S(f)(Tr^{Ev})$ component (autonomous) partial Petri nets; 4) input realizing $T^{-1}(S(f)_i)$ and output transporting $T(S(f)_i)$ nodal topological subnets; 5) input realizable $S(f)(R^{Ev}{}_{T^{-1}(S(f)i)})$ and output transportable $S(f)(Tr^{Ac}{}_{T(S(f)i)})$ nodal (for input and output topological subnets) partial Petri nets [26, 28]; 6) inherited, adjacent up or down for hierarchy, $iS(f)_i$ hierarchies of properties and transactions iPr_i, identifiers iCi_i, check iCp_i and linking iLp_i primitives, check fragments iCf_i, respectively [29];

- $\{cs_{0i}, cm_{0i}, q_{0i}, st_{0i}\}$ – initial agent check model, method, goal, strategy.

Then the combined agent check model $cs_i \in cS_i$ from the above set models cS_i for components of abstract $S(f)_i$, network $nS(f)_i$ and hierarchical $iS(f)_i$ models is defined as:

$$cs_i = ((W_i^{\wedge}, Ci_i, Cp_i, Lp_i, Cf_i, Sg_{S(f)i}),$$
$$(S(f)_i(R^{Ac}{}_i), S(f)_i(Tr^{Ev}{}_i), T^{-1}(S(f)_i), T(S(f)_i), S(f)_i(R^{Ev}{}_{T^{-1}(S(f)i)}), S(f)_i(Tr^{Ac}{}_{T(S(f)i)}), Sg_{nS(f)i}),$$
$$(iPr_i, iCi_i, iCp_i, iLp_i, iCf_i, Sg_{iS(f)i})). \tag{10}$$

The representation of the abstract component agent meta-method of behavioral check $cm_i \in cM_i$ as a special, asynchronous-event parallel behavioral procedure with many realizations, that is satisfying the conditions of the corresponding component agent check model $cs_i \in cS_i$, can be formally determined by using the special Petri net - a special kind of the Rabin-Scott automata, the components of which individually and the relationships for them were mostly determined earlier:

$$cm_i = (\{W^{\wedge}, Ev_i, Ac_i\}, \cup_{j \in J}(W_{ij}{}^{\wedge}, Cf_{ij}, Pr_{ij}, \{\vartheta_{ij}, \Lambda_{ij}, \Theta_{ij}\}), \{Pr_i, Ci_i, Cp_i, Cf_i\}, Mm_i,$$
$$\{\rho\varepsilon_i, \alpha\sigma_i, \kappa o_i\}, \{\alpha_i, \beta_i, \gamma_i\}, \{\varphi\upsilon_i, \lambda\varepsilon_i, \theta\upsilon_i\}, \{\mu_i, \mu_{i0}\}, (W_{i0}{}^{\wedge} = Cf_{i0}, Pr_{i0} = \varnothing,$$
$$\{\vartheta_{i0} = 0, \Lambda_{i0} = 0, \Theta_{i0} = 0\}), (W_{iF}{}^{\wedge}, Cf_{iF}, Pr_{iF}, \{\vartheta_{iF}, \Lambda_{iF}, \Theta_{iF}\})), \tag{11}$$

where: a) $\{Ev_i, Ac_i, W^{\wedge}\}$ – events Ev_i, actions Ac_i, input-output words $W^{\wedge} \subseteq Ev_i \times Ac_i$ with unmarked (unrecognized) non-terminals from $Ev_i \backslash X_i$ and $Ac_i \backslash Y_i$; b) $\cup_{j \in J}(W_{ij}{}^{\wedge}, Cf_{ij}, Pr_{ij}, \{\vartheta_{ij}, \Lambda_{ij}, \Theta_{ij}\})$ – states of Rabin-Scott automata, as part of the current sets of input-output words $W_{ij}^{\wedge}$, verified properties Pr_{ij}, reached values of completeness ϑ_{ij}, length Λ_{ij}, multiplicity Θ_{ij} of the check; c) $\{Pr_i, Ci_i, Cp_i, Cf_i\}$ – properties and transactions Pr_i, identifiers Ci_i, check primitives Cp_i and fragments Cp_i, presented earlier; d) $Mm_i = \{mm_1, mm_2, \ldots, mm_{nmm}\}$ – chips – weighted mandates for forming events in states (non-terminals) and performing actions in check fragments, the chip has the form of a triple

$mm_{nmm}=\{\vartheta_{nmm},\Lambda_{nmm},\Theta_{nmm}\}$ with elements - current cumulative values of variables of completeness ϑ_{nmm}, length (complexity, time) Λ_{nmm}, multiplicity Θ_{nmm} of check; e) $\{\rho\varepsilon_i,\alpha\sigma_i,\kappa o_i\}$ – operations: – $\rho\varepsilon_i:(W_{ij}^{\wedge},Ev_i,Ac_i)\rightarrow(W_{ij}^{\wedge},Ev_i,Ac_i)$ – registration of input-output words, events and actions, as a part of input-output words; – $\alpha\sigma_i$: – associations of events and actions with reference properties, identifiers, check primitives; – κo_i – compatibility (intersection) of the contexts of the neighborhoods from the corresponding input-output words; g) $\{\alpha_i,\beta_i,\gamma_i\}$ – operations, presented earlier, respectively, of identifying α_i, congruence β_i and determining γ_i; h) $\{\varphi\upsilon_i,\lambda\varepsilon_i,\theta\upsilon_i\}$ – previously defined functions of determining, respectively, completeness ϑ_i, length (complexity) Λ_i, multiplicity Θ_i; i) $\mu_{i0}:(W_{i0}^{\wedge}=Cf_{i0},Pr_{i0}=\varnothing,\{\vartheta_{i0}=0,\Lambda_{i0}=0,\Theta_{i0}=0\})\rightarrow\{\vartheta_{i0}=0,\Lambda_{i0}=0,\Theta_{i0}=0\}$ – initialization markup for the initial state, taking into account zero results, here $mm_{i0}=\{\vartheta_{i0}=0,\Lambda_{i0}=0,\Theta_{i0}=0\},\mu_i:(\cup_{j\in J}(W_{ij}^{\wedge},Cf_{ij},Pr_{ij},\{\vartheta_{ij},\Lambda_{ij},\Theta_{ij}\}))\rightarrow(\cup_{j\in J}\{\vartheta_{ij},\Lambda_{ij},\Theta_{ij}\})$ – function of the current markup of states, taking into account the current completeness, length, multiplicity of check, which are accumulated during movement in state and modified chips, here $mm_i=\{\vartheta_i,\Lambda_i,\Theta_i\}$, for some current $(W_{ij}^{\wedge},Cf_{ij},Pr_{ij},\{\vartheta_{ij},\Lambda_{ij},\Theta_{ij}\})\in\cup_{j\in J}(W_{ij}^{\wedge},Cf_{ij},Pr_{ij},\{\vartheta_{ij},\Lambda_{ij},\Theta_{ij}\})$, that is $mm_i=\{\vartheta_{ij},\Lambda_{ij},\Theta_{ij}\}=\mu_i((W_{ij}^{\wedge},Cf_{ij},Pr_{ij},\{\vartheta_{ij},\Lambda_{ij},\Theta_{ij}\}))$; j) $(W_{i0}^{\wedge}=Cf_{i0},Pr_{i0}=\varnothing,\{\vartheta_{i0}=0,\Lambda_{i0}=0,\Theta_{i0}=0\})$ – initialization markup for the initial state, taking into account the initial state of the Rabin-Scott automata, consisting of initial sets of input-output words $W_{i0}^{\wedge}$, verified properties Pr_{i0}, initial (zero) values of completeness $\vartheta_{i0}=0$, length $\Lambda_{i0}=0$, multiplicity $\Theta_{i0}=0$ of the check; k) $(W_F^{\wedge},Cf_{iF},Pr_{iF},\{\vartheta_{iF},\Lambda_{iF},\Theta_{iF}\})$ – the final state of the Rabin-Scott automata, as a part of finite sets of input-output words $W_{iF}^{\wedge}$, verified properties of Pr_{iF}, achieved values of completeness $\vartheta_{iF}=\varphi\upsilon_i(Pr_{iF})$, length $\Lambda_{iF}=\lambda\varepsilon_i(Cf_{iF})$, multiplicity $\Theta_{iF}=\theta\upsilon_i(Cf_{iF})$ of check.

5. Notes for method and implementation

The meta-method can be implemented in deterministic, pseudo-random, evolutionary, and also combined methods based on the search in depth and/or in width.

So, model evolutionary concretization and modification of the component agent method of behavioral testing $cem_i\in ceM_i$ has specialized view for evolutionary operations β-μ_i,β-κ_i and functions $\varphi\upsilon$-ϕ_i,$\lambda\varepsilon$-ϕ_i,$\theta\upsilon$-ϕ_i,$\varphi\upsilon$-σ_i,$\lambda\varepsilon$-σ_i,$\theta\upsilon$-σ_i:

$$cem_i=(\{W^{\wedge},Ev_i,Ac_i\},\cup_{j\in J}(W_{ij}^{\wedge},Cf_{ij},Pr_{ij},\{\vartheta_{ij},\Lambda_{ij},\Theta_{ij}\}),\{Pr_i,Ci_i,Cp_i,Cf_i\},Mm_i,\{\rho\varepsilon_i,\alpha\sigma_i,\kappa o_i\},\{\alpha\sigma_i\text{-}\alpha_i,\beta\text{-}\mu_i,\beta\text{-}\kappa_i,\gamma_i\},\{\varphi\upsilon\text{-}\phi_i,\lambda\varepsilon\text{-}\phi_i,\theta\upsilon\text{-}\phi_i,\varphi\upsilon\text{-}\sigma_i,\lambda\varepsilon\text{-}\sigma_i,\theta\upsilon\text{-}\sigma_i\},\{\mu_i,\mu_{i0}\},(W_{i0}^{\wedge}=Cf_{i0},Pr_{i0}=\varnothing,\{\vartheta_{i0}=0,\Lambda_{i0}=0,\Theta_{i0}=0\}),(W_{iF}^{\wedge},Cf_{iF},Pr_{iF},\{\vartheta_{iF},\Lambda_{iF},\Theta_{iF}\})), \quad (12)$$

here modified objects – operations and functions [24, 28, 29]: a) $\alpha\sigma_i$-α_i – associations of evolutionary events and actions with identifiers, based on check identification α_i; b) β-μ_i – binary operation of a modifying or expanding mutation based on the check congruence β_i; c) β-κ_i – binary operation of modifying or expanding crossing-over based on the control congruence β_i; d) $\varphi\upsilon$-ϕ_i – the posteriori unary fitness function for determining the completeness ϑ_i of a new check fragment Cf_i (a new individual of evolution), based on the function for determining of the check completeness φ_i; e) $\lambda\varepsilon$-ϕ_i – the posteriori unary fitness function for determining the length (complexity) Λ_i of the new check fragment Cf_i, based on the function for determining the check length λ_i; g) $\theta\upsilon$-ϕ_i – the posteriori unary fitness function for determining the multiplicity Θ_i of the new check fragment Cf_i, based on the function for determining the check multiplicity θ_i; h) $\varphi\upsilon$-σ_i – the priori binary selection function for determining the total completeness ϑ_i of the check fragments Cf_{i1} and Cf_{i2} – possible parents of the new check fragment Cf_i, based on function for determination of the check completeness φ_i; i) $\lambda\varepsilon$-σ_i, – the priori binary selection function for determining the total length (complexity) Λ_i of the control fragments Cf_{i1} and Cf_{i2} – possible parents of the new check fragment Cf_i, based on function for determination of the check length λ_i; j) $\theta\upsilon$-σ_i – the priori binary selection function for determining the total multiplicity Θ_i of the check fragments Cf_{i1} and Cf_{i2} – possible parents of the new check fragment Cf_i, based on the function for determination of the check multiplicity Θ_i.

The procedure of the component agent evolutionary method cem_i of behavioral testing is an event model and does not determine the serial-parallel organization of check, in particular, the combined organization of search in width and depth. Such an organization can be determined by using relations for subjects and objects in their structures and locations, as well as for their operations/functions.

Relationships include special (dependencies, generalizations, implementations, associations, aggregations, inheritance), basic (precedence, order, equivalence, compatibility, incompatibility, uncertainty, indifference), properties (reflexivity, symmetry, transitivity connectivity), both forming and limiting the possible structures of the actions of the method. At the internal level, these relations are determined by abstract resources shared in a certain way, first of all, by basic computer-processors, information objects, and communication tools.

Computing processors and communication tools directly depend on the environment of the agent's location, therefore, their influence on the relationship between objects and operations/functions of the agent is indirect to the environment of placement and can be considered instrumental.

Information objects are abstracted from the placement environment, their influence on the relationship between objects and agent operations/functions is direct and can be considered methodological. In this regard, it is necessary to consider the current, previous and subsequent influences - temporary relationships for objects (including information) and operations/functions, as well as their instances in the agent's lifetime. In such relations, the input and output weight characteristics of the action object (usually an information object) performed by the subject of the action (usually an operation/function) are significant. These weight characteristics are essential precisely for the subjects of action and their instances, when organizing the system of priorities and parallelism, and also the corresponding system of method relations.

Own relations for operations/functions, as subjects of the current moment of time and the current shared object of action in the evolutionary method of cem_i behavioral check, are general relations: firstly, of direct preceding (see table 1), which defines adjacent sequential subjects of action - operations/functions with the transfer of a shared action object between them; secondly, compatibility (see table 2), which defines parallel subjects of the action - operations/functions with conflict-free separation of the objects of action. It is obvious, that in the evolutionary method of behavioral check cem_i for operations/functions, as subjects of the current moment of time, the relation of indifference acts for different current individual objects of action not separated by subjects. That is, parallelism for such operations/functions and individual objects is methodologically not limited.

Table 1

The Relationship of the immediate preceding (quasiorder)

	$\rho\varepsilon_i$	κo_i	$\alpha\sigma_i$	$\alpha\sigma_i$-α_i	β-μ_i	β-κ_i	γ_i	$\varphi\upsilon$-ϕ_i,	$\lambda\varepsilon$-ϕ_i,	$\theta\upsilon$-ϕ_i	$\varphi\upsilon$-σ_i,	$\lambda\varepsilon$-σ_i,	$\theta\upsilon$-σ_i
$\rho\varepsilon_i$	*	*	*	*									
κo_i		*	*	*									
$\alpha\sigma_i$		*	*	*				*	*	*			
$\alpha\sigma_i$-α_i		*		*	*	*		*	*	*			
β-μ_i		*	*	*	*		*						
β-κ_i		*	*	*		*	*						
γ_i		*	*	*			*						
$\varphi\upsilon$-ϕ_i,								*			*		
$\lambda\varepsilon$-ϕ_i,									*			*	
$\theta\upsilon$-ϕ_i										*			*
$\varphi\upsilon$-σ_i,					*	*					*		
$\lambda\varepsilon$-σ_i,					*	*						*	
$\theta\upsilon$-σ_i					*	*							*

Relationships of indifferent is valid for operations/functions, as subjects of the current moment of time, for different current (individual) action objects, that are not separated by these operations/functions in the evolutionary behavioral check method cem_i, therefore, the parallelism for such operations/functions and individual objects is unlimited methodologically.

Table 2

The ratio of compatibility (tolerance):

	$\rho\varepsilon_i$	$\alpha\sigma_i$	κo_i	$\alpha\sigma_i$-α_i	β-μ_i	β-κ_i	γ_i	$\varphi\upsilon$-ϕ_i,	$\lambda\varepsilon$-ϕ_i,	$\theta\upsilon$-ϕ_i	$\varphi\upsilon$-σ_i,	$\lambda\varepsilon$-σ_i,	$\theta\upsilon$-σ_i
$\rho\varepsilon_i$	*												
$\alpha\sigma_i$		*	*	*									
κo_i		*	*	*									
$\alpha\sigma_i$-α_i		*	*	*									
β-μ_i					*	*							
β-κ_i					*	*							
γ_i							*						
$\varphi\upsilon$-ϕ_i,								*	*	*			
$\lambda\varepsilon$-ϕ_i,								*	*	*			
$\theta\upsilon$-ϕ_i								*	*	*			
$\varphi\upsilon$-σ_i,											*	*	*
$\lambda\varepsilon$-σ_i,											*	*	*
$\theta\upsilon$-σ_i											*	*	*

The specificity of the check functioning of the deterministic reactive component ag_{Ri} of the agent ag_i – the execution of its operations $\{o_i, \sigma_i, \alpha_i\}$ – can be based on the methods of depth and breadth search with local search optimization for behavioral testing, which bases on experiments with automata model, and is characterized by computational NP-complexity.

This circumstance implies the limitation of the analysis space — a subset of the DIS components — by objects of medium complexity (up to 1000 positions/ /transitions) to obtain the solution of check tasks of the required characteristics. These characteristics include high completeness $\varphi\upsilon_i$, acceptable properties of length $\lambda\varepsilon_i$ and multiplicity $\theta\upsilon_i$, realizability $\rho\varepsilon_i$, transportation $\tau\rho_i$, inheritance $\iota\nu_i$ for spent of time $\tau\iota_i$ and memory $\mu\varepsilon_i$, limited by the upper bounds of the allocated computing resources $R_{agRi}=(\tau\iota_{MaxagRi}, \mu\varepsilon_{MaxagRi})$ of the reactive component ag_{Ri}.

The specificity of the functioning of the deliberative component of the agent for performance of its operations $\{o_i, \sigma_i, \alpha_i\}$ can be based on a pseudo-random goal-oriented check evolutionary search. As result, this functioning retains the upper exponential analytical estimates of deterministic methods for experiments with automata and Petri nets, but it has experimental complexity, significantly less computational NP-complexity of the latter.

This leads to the possibility of solution of check tasks of acceptable completeness and length with time and memory costs lower, than average. This range is limited by the lower and upper boundaries of the resources $R_{agDi}=(\tau\iota_{MaxagDi}, \mu\varepsilon_{MaxagDi})$ of the deliberative component in the analysis space of the objects above an average degree of complexity (more, than 1000 positions/transitions) - a subset of the DIS components and DIS in as a whole. The results of analytical modeling made it possible to verify the models and assess the domain of their applicability, in particular, the possibility of moving the *NP*-complexity of the behavioral check into the reachable area.

Petri nets *S(f)* are represented in the memory of the controllers for monitoring systems by using of the list structures. Let the following dimensions of the sets be adopted for the Petri net: $|P|=n_p$, $|T|=n_t$, $n= n_p+n_t$, $|X|=m$, $|Y|=l$. The structures require for the upper limit of the number of conditional fields:

$$c_{iS}^{max}=(n_t(4n_p+3L+4)+n_p(3m+2))+(\Sigma_{i\in I}\, n_{t\,i}(4n_{pi}+3L_i+4)+n_{pi}(3m_i+2))+ \\ +(\Sigma_{j\in J}\, n_{tj}(4n_{pj}+3L_j+4)+n_{pj}(3m_j+2)). \quad (13)$$

The complexity of the check analysis of Petri net *S(f)* includes the preprocessor and main stages for each of evolutions and is determined by upper bound:

$$c_{ciS}^{max}=n_t(4n_p+3L+4)+n_p(3m+2)+2n_tn_p(n_t-1)+2(2Lmn_pn_t)^{nt}-3)+(n_t-1)(n_pn_t)!+ \\ +(\Sigma_{i\in I}\, n_t(4n_{pi}+3L_i+4)+n_{pi}(3m_i+2)+2n_{ti}n_{pi}(n_{ti}-1)+2(2L_im_in_{pi}n_{ti})^{niti}-3)+(n_{ti}-1)(n_{pi}n_{ti})!)+ \\ +(\Sigma_{j\in J}\, n_{tj}n_{tj}(4n_{pj}+3L_j+4)+n_{pj}(3m_j+2)++2n_{tj}n_{pj}(n_{tj}-1)+2(2L_jm_jn_{pj}n_{tj})^{njt\,j}-3)+(n_{tj}-1)(n_{pj}n_{tj})!). \quad (14)$$

These formulas contain the three parts for Petri net and Petri subnets. The comparison of the online testing programs, based on automata-deterministic and Petri-evolutionary models and methods for

onboard automated control systems and terminal video surveillance confirmed a decrease for the computational complexity of online testing and reducing the its time. Upper computational complexity and length of the online testing for the Determination (solid) and Evolutional (dashed) models and methods in cases of simple Petri net (•) and hierarchical Petri net (♠) are presents in Fig.1.

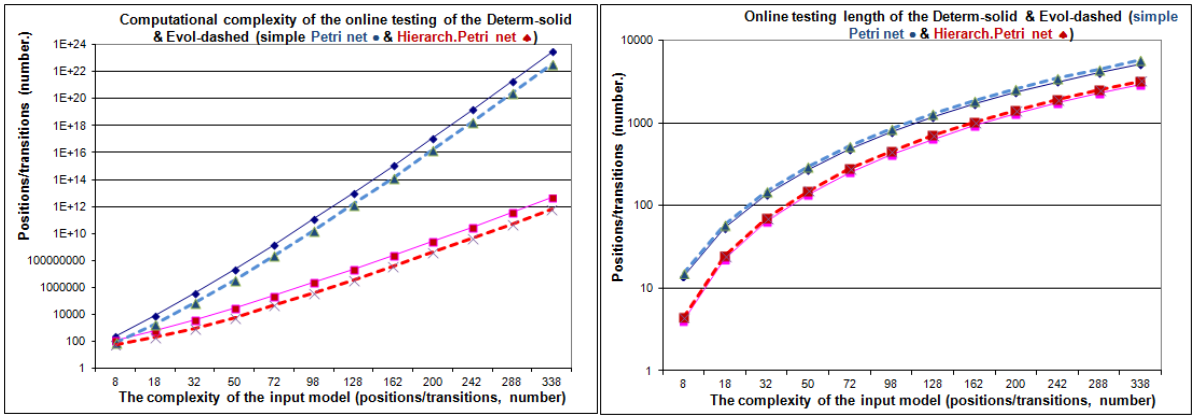

Figure 1: Computational complexity and length of online testing of the Determ-solid & Evol-dashed (simple PN • & Hierarch.PN ♠)

The basic programs of the behavioral online testing were carried out for component and decomposition of DIS, the results are presented in Tab. 3.

Table 3

Experimental values of computational complexity of check:

Object	Degree of decomposition	Input complexity	Complexity of check
Module BSAC	8	362	64980
Module TVS	2	50	4323
Module IP/IPSec	3	115	71871
Module of Naming	2	146	38150450
Module of Encaps.	3	180	1572123

The comparison of the results of basic programs for automata-deterministic and Petri-evolutionary models and methods of online testing for onboard automated control systems (BSAC) and terminal video surveillance (TVS) confirmed: a) decrease in computational complexity of online testing; b) reducing timer time of the check with preservation of: c) the length of the check and their d) completeness.

6. Conclusions

The proposed agent-based model of behavioral checking by taking into account the resource conditions of the deployment environment, when checking the DIS components allows to balance and perform distributed behavioral online and offline testing, based on its combined deterministic and evolutionary models and methods, as well as models of the agent deployment environment in the DIS itself.

The proposed combined models, due to the decomposition of the check processes of the distributed MAS, coordinated with the resources of the DIS, make it possible to reduce the time of the DIS check by up to 90% in comparison with the non-decomposition approach.

These models and methods increase the completeness, accuracy and efficiency of the checking of the structure and functioning of real DIS with acceptable computational costs of the MAS level built into DIS. Its exponential complexity, limiting the use of behavioral check, is polynomially reduced

due to component (DIS) and agent-based (MAS) decomposition. This, in turn, allows to check more complex DIS.

7. References

[1] N. M. Josuttis, SOA in Practice: The Art of Distributed System Design (Theory in Practice), O'Reilly Media. 2007.

[2] M. Tamer Ozsu, P. Valduriez, Principles of Distributed Database Systems (3rd Edition), Springer, 2011.

[3] M. Ouzzani, A. Bouguettaya, Semantic Web Services for Web Databases, Springer Science+Business Media, 2011.

[4] S. Pakari, E. Kheirkhah, M. Jalali, Web-service Discovery Methods and Techniques: a Review, International Journal of Computer Science, Engineering & Information Technology 4 (2014) 1–14.

[5] X. Jing, L. Chen-Ching, Multi-agent systems and their applications, Journal of International Council on electrical engineering 7(1) (2017) 188–197.

[6] Y. Shoham, K. Leyton-Brown, Multiagent systems. Algorithmic, Game-Theoretic, and Logical Foundations (Revision 1.1), Shoham and Leyton-Brown, 2009, 2010.

[7] F. Berman, G. Fox, T. Hey, Grid Computing: Making the Global Infrastructure a Reality, John Wiley & Sons Ltd, England, 2005.

[8] Installation Guide for UNICORE Server Components, 2018. URL: http://linksceem.cyi.ac.cy/ls2/images/stories/UNICORE_Admin_Workshop.pdf

[9] C. J. Date, Date on Database: Writings 2000–2006, Apress, 2006.

[10] M. Stonebraker, S. Madden, P. Dubey, Intel "big data" science and technology center vision and execution plan, ACM SIGMOD Record 42(1) (2013) 44-49.

[11] S. J. Russell, P. Norvig, Artificial Intelligence: a Modern Approach, Prentice-Hall, A Simon & Schuster Company Englewood Cliffs, New Jersey, 2010.

[12] P. H. Winston, Artificial Intelligence (3rd Edition), Addison-Wesley Pub Co; 3rd edition, 1992.

[13] P. Norvig, S. J. Russell, Artificial Intelligence: A Modern Approach (2nd Edition), Prentice Hall; 2002.

[14] E. Friedman-Hill, Jess in Action: Rule-Based Systems in Java, Manning Publications Company, 2003.

[15] Y. Kondratenko, O. Kozlov, O. Korobko, A. Topalov, Complex Industrial Systems Automation Based on the Internet of Things Implementation, in: Bassiliades, N. et al. (Eds.), Information and Communication Technologies in Education, Research, and Industrial Applications: ICTERI'2017, CCIS 826, Springer, Cham, 2018, pp.164–187.

[16] O. Drozd, V. Kharchenko, A. Rucinski, T. Kochanski, R. Garbos, D. Maevsky, Development of Models in Resilient Computing, in: 10th IEEE International Conference on Dependable Systems, Services and Technologies, Leeds, UK, 2019, pp. 2–7. doi: 10.1109/DESSERT.2019.8770035

[17] V. Hahanov, O. Mishchenko, V. Soklakova, V. Abdullayev, S. Chumachenko, E. Litvinova, Cyber-Social Computing, in: V. Kharchenko, Y. Kondratenko, J. Kacprzyk (Eds.) Green IT Engineering: Social, Business and Industrial Applications, volume 171 Studies in Systems, Decisions and Control, Springer, Berlin 2019, pp. 489-515.

[18] K. Sinha, Structural Complexity and its Implications for Design of Cyber Physical Systems, Ph.D. thesis, Massachusetts Institute of Technology, Engineering Systems Division, January 3, 2014.

[19] S. Chernov, S. Titov, L. Chernova, V. Goginskii, Lb. Chernova, K. Koles, Algorithm for the simplification of solution to discrete optimization problems, Eastern-European Journal of Enterprise Technologies 3(4) (2018) 1–12.

[20] O. Drozd, M. Kuznietsov, O. Martynyuk, M. Drozd, A method of the hidden faults elimination in FPGA projects for the critical applications, in: 9th IEEE International Conference DESSERT, Kyiv, Ukraine, 2018, pp. 231–234. doi: 10.1109/DESSERT.2018.8409131

[21] O. Drozd, V. Romankevich, M. Kuznietsov, Using natural version redundancy of FPGA projects in area of critical applications, in: 11th IEEE International Conference DESSERT, Kyiv, Ukraine, 2020, pp. 58–63. doi: 10.1109/DESSERT50317.2020.9125050

[22] O. Drozd, I. Perebeinos, O. Martynyuk, K. Zashcholkin, O. Ivanova, M. Drozd, Hidden fault analysis of FPGA projects for critical applications, in: IEEE International Conference TCSET, Lviv–Slavsko, Ukraine, 2020, pp. 467-471. doi: 10.1109/TCSET49122.2020.235591

[23] V. Kudryavtsev, S. Grunskii, V. Kozlovskii, Analysis and synthesis of abstract automata, Journal of Mathematical Sciences 169(4) (2010) 481-532.
[24] R. Praveen, S. Km Baby, Automated Software Testing Using Metaheuristic Technique Based on An Praveen Ranjan Ant Colony Optimization Electronic System Design (ISED), International Symposium, 20-22 Dec. 2010, Bhubaneswar, 2010, pp. 235–240.
[25] D. Zaitsev, Toward the Minimal Universal Petri Net, IEEE Transactions on Systems, Man, and Cybernetics: Systems 44(1) (2013) 47-58.
[26] V. Opanasenko, S. Kryvyi, Synthesis of multilevel structures with multiple outputs, in: CEUR Workshop Proc. of the 10th International Conference of Programming, UkrPROG 2016, Kyiv, Ukraine, 24 May 2016, Volume 1631, 2016, pp. 32–37.
[27] O. Martynyuk, O. Drozd, H. Stepova, D. Martynyuk, L. Sugak, Multidimensional Hierarchical Model of Behavioral Check of Distributed Information Systems, in: Proc. of 2019 IEEE East-West Design & Test Symposium (EWDTS), Batumi, 2019, pp. 517-522.
[28] O. Martynyuk, A. Sugak, D. Martynyuk, O. Drozd, Evolutionary Network Model of Testing of the Distributed Information Systems, in: Proc. 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Bucharest, Romania, 2017, pp. 888–893.
[29] O. Martynyuk, O. Drozd, T. Ahmesh, B. V. Thuong, A. Sachenko, H. Mykhailova, M. Dombrovskyi, Hierachical Model of Behavior On-line Testing for Distributed Information Systems, in: 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Vol. 2, Metz, France, 2019, pp. 724-729.
[30] C. Pacheco, K. Shuvendu M. Lahiri, D. Ernst, T. Ball, Feedback-directed random test generation, in: Proc. of the 29th International Conference on Software Engineering, 2007, pp. 75–84.
[31] I. Ciupa, A. Pretschner, M. Oriol, A. Leitner, B. Meyer, On the number and nature of faults found by random testing, Software Testing, Verification and Reliability 21 (2009) 3–28.
[32] Y. Wang, T. Nicol, Statistical Properties of Pseudo Random Sequences and Experiments with PHP and Debian OpenSSL, in: European Symposium on Research in Computer Security, 2014, pp. 454–471.
[33] A. Takanen, J. D. Demott, C. Miller, Fuzzing for Software Security Testing and Quality Assurance (Second Edition), Artech House, 2018.
[34] J. Offutt, W. Xu, Generating Test Cases for Web Services Using Data Perturbation, ACM SIGSOFT Software Engineering Notes 29(5) (2004) 1-10.
[35] Y. Kondratenko, N. Kondratenko, Soft Computing Analytic Models for Increasing Efficiency of Fuzzy Information Processing in Decision Support Systems, in: R. Hudson (Ed.) Decision Making: Processes, Behavioral Influences and Role in Business Management, Nova Science Publishers, New York, 2015, pp.41-78.
[36] V.-T. Pham, M. Böhme, A. Roychoudhury, Model-based whitebox fuzzing for program binaries, in: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering - ASE 2016, 2016, pp. 543–553.
[37] J. Hernández-Orallo, F. Martínez-Plumed, U. Schmid, M. Siebers, D. L. Dowe, Computer models solving intelligence test problems programs and implications, Artifficial Intelligence 230 (2016) 74-107.
[38] Y. Skobtsov, V. Skobtsov, Evolutionary test generation methods for digital devices, Design of Digital Systems and Devices, in: M.Adamski et al. (Eds.), volume 79 of Lecture Notes in Electrical Engineering, Berlin: SpringerVerlag, 2011, pp. 331–361.
[39] A. Classen, C. Cordy, P. Y. Schobbens, P. Heymans, A. Legay, J.-F. Raskin, Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking, IEEE Transactions on Software Engineering 39(8) (2012) 1069–1089.
[40] D. Brucker, D. Achim, W. Burkhart, On Theorem Prover-based Testing, Formal Aspects of Computing 25 (5) (2012) 683–721.
[41] L. Gomes J. M. Fernandes, Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation, International Science Reference, 2010.
[42] A. Sugak, O. Martynyuk, O. Drozd, The hybrid agent model of behavioral testing, International Journal of Computing 14 (4) (2015) 234-246.