computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

A HYBRID AGENT MODEL OF BEHAVIORAL TESTING

Anna Sugak [1], Oleksandr Martynyuk [2], Oleksandr Drozd [3]

[1] Department of Computerized Control Systems, Odessa National Polytechnic University, Shevchenko Avenue, 1, Odessa, 65044, Ukraine, e-mail: sygak.anna@mail.ru
[2] Department of Computer Intellectual Systems and Networks, Odessa National Polytechnic University, Shevchenko Avenue, 1, Odessa, 65044, Ukraine, e-mail: anmartynyuk@ukr.net, drozd@ukr.net

Abstract: Operation testing and diagnostic tests, applied for distributed information systems, inherit and employ the properties of distribution, autonomy, goal formation and cooperation, natural for the multi-agent systems. This paper presents the behavioral diagnostics agent model, based on the evolutionary organization of component tests in the automata network environment. The model can be used to construct a multi-agent diagnostics system. A hybrid agent model provides a combination of reactive operation testing and deliberative diagnostic tests, based on the deterministic and evolutionary methods of synthesis of behavioral tests. An agent model consists of the component models of allocation environment, functioning goals and strategies, operations of observation, enforcement strategy and adaptation, initial component models, goals and strategies for ensuring the autonomy. Agent intelligence is based on a locally-exhaustive deterministic and pseudorandom targeted evolutionary synthesis of behavioral tests, providing and accumulating the results. Cooperation of the agents involves their deterministic and evolutionary interactions under the conditions of test feasibility and portability.

Keywords: distributed information system, behavioral test, the evolutionary system, agent, multi-agent diagnostics system.

1. INTRODUCTION

Rapidly developing modern distributed information systems (DIS) are characterized by [1, 2]:

- the emergence of service-oriented architectures, cluster and GRID-systems, cloud and multi-agent technology, used under the conditions of partial definiteness and non-determination;
- the development of mathematical software for the purpose of creation of decomposed, functional, fuzzy, intelligent and competitive models;
- the integration of technologies, architectures and models with the use of common tools and technological means of design and application.

The effectiveness of the DIS depends on the amount of information, quality and reliable performance, efficiency and reliability of the results. The most important way to increase the reliability of the DIS is the technical diagnosing, which is the cornerstone of construction and introductions of the automated technical diagnosis systems (ATDS) [3, 4]. ATDSs generally include complementary means of operation and test control of DIS efficiency.

At the same time, the DIS improvement promotes the development of the ATDS and the tools for their constructing. As a result, the complexity of ATDS development becomes comparable with the complexity of the DIS. Furthermore, in some cases, it surpasses the last one within the systems of critical application.

These DIS features are inherited by the ATDS models. Their importance is increased according to:

- scalability, distribution, multi-level and multi-platform nature, involving consideration of a great majority of special local features and characteristics;
- complexity, dynamism and incomplete definition, required for the elaboration of behavioral high-level specifications and formal models of the entire DIS, its subsystems and the component compositions, and non-trivial interfaces of these components, their compositions and subsystems;
- autonomy, cooperation, goal formation, additional mobility, allowing to use the efficient technology of neuron networks and multi-agent systems (MAS) [5-7].

In addition, strong coherence of the DIS and ATDS life cycles, from the design stage is feasible. In particular, system, structural and functional specifications, distribution and multilevel nature of ATDS and DIS allow to use the developed behavioral test software of ATDS to verify the DIS

projects and ultimately, to use it for the DIS implementation testing [8, 9].

Thus, a conclusion can be drawn about the relevance and nontriviality of the solution of the problems of ASTD creation for the complex DIS.

2. ANALYSIS OF EXISTING SOLUTIONS

Test synthesis, usually characterized by *NP*-complexity, requirements for the check of high completeness at allowable computing costs [10, 11, 12], is of great importance for the verification and testing. Development of the models and methods for the automata theory and experiments, determining the general methodology of the behavioral test analysis, removes the upper bounds of applicability of the test automata models [13, 14]. Complexity reduction, first of all, the reduction of the time of behavioral tests synthesis in a class of errors of mapping with the substantially saved values of the check completeness, is reached by:

- the distributed network [15, 16, 17] and through-hierarchical [18] methods, reducing polynomially the dimension of the test synthesis and its duration due to the decomposition;
- the evolutionary-genetic methods [19-21], which have the upper bounds of complexity of the determined methods and give polynomially smaller experimental values of complexity of the tests synthesis for the majority of cases [22].

Nevertheless, the synthesis of behavioral tests of the acceptable check completeness is possible for DIS with an average degree of complexity; it refers to the test of *NP*-complex class despite this reduction. This promotes, in particular, the feasibility of further development of the systems of evolutionary-genetic network methods with the combined effect of complexity reduction.

Being a part of real time and critical application systems, a considerable part of modern DIS increases the requirements for completeness, accuracy and relevance of the operation check of efficiency and necessary test resources. This promotes to apply ASTD, including means of operation and test control [8, 11], even at the structural and functional system level. At this level the ATDS operation is formally and intuitively based on a system of behavioral testing methods [23].

The possibility of passive mode operation control and active mode test control is a characteristic feature of behavioral tests [13]. Passive mode deals with the current accumulation of presentable passive recognizing experiments, as a background process of the DIS components normal operation functioning. The interruption of the operation functioning occurs during the active mode of test control, thus, the control test verification experiments are conducted. Both cases are accompanied, if necessary, by the previous internal or external test synthesis.

However, insufficiently investigated formalization of walkthrough comprehensive operation and test behavioral control and ambiguous definition of the verifiable errors class doesn't allow ATDS to control effectively over the received values of the system check completeness, accuracy and relevance and functional verification of DIS. It promotes further development of the systems of the network determined and evolutionary-genetic methods, the solution of the problems of their allocation in the DIS environment, initialization and cooperative performance.

Currently, there are increasingly effective ATDSs of distributed systems, using models and methods of multi-agent technology [24-26]. Nevertheless, there are issues of multi-agent analysis that have to be developed, such as network behavior analysis, cooperation and intelligence of their agents, especially in the case of:

- incomplete determination of testable properties (errors), reducing the check completeness
- network control, limited by separate rules, and monitoring of the diagnosis, that narrows the space of behavioral tests search;
- fixed combination of deterministic and evolutionary methods, complicating the dynamic adjustment to situationally created cooperation;
- a lack of operation and test control, eliminating the background behavioral testing in operation mode and requiring for check completeness in the operation test pauses.

Thus, we can draw a conclusion about the usability of research of the system and functional control increasing in the error class of the DIS mapping and the development of the agent model of the DIS comprehensive operation and test behavioral control, based on the deterministic and evolutionary-genetic network methods, implemented in the ATDS.

3. GOAL AND TASKS

The DIS behavioral model as the network of automata (NA) is researched as the MAS environment model and is characterized by structure, alphabets and compliances of components, data and knowledge structures, which can be altered in the DIS life cycle.

The MAS operation and test control in DIS is represented by agents Ag cooperation, placed in the DIS environment. The MAS atomic element $ag_i \in Ag$ presents the first-level MAS model.

The purpose of constructing of a certain agent ag_i model is formalization of presentation of the behavioral agent-based operation and test control.

This control is performed on the base of the automata model $a_i \in A$ for the components of DIS within the contexts:

- autonomous;
- the goal forming or intellectual;
- cooperative.

The context of mobility is not being considered in this paper.

The majority of the tests of the DIS behavioral control performed by the agent ag_i of the model of automata a_i, includes:

- construction of identifiers of Ti_i resistance states;
- construction of test Tp_i and binders Lp_i primitives;
- passive component recognition of text fragments Tf_i up to the behavioral tests of DIS component for its operation control;
- the active formation of component test fragments Tf_i up to the behavioral tests of the DIS component for its test control;
- forming of the component experiments – test experiments of deterministic $texp_{Di} \in TExp_D$, evolution $texp_{Ei} \in TExp_E$, combined $texp_{DEi} \in TExp_{DE}$ behavioral testing for the agent ag_i;
- construction of component minimized, input implemented aR_i and output transported aTr_i semi-automata, nodal implemented $aR^{X}{}_{T}{}^{-1}{}_{(ai)}$ and nodal transported $aTr^{Y}{}_{T(ai)}$ semi-automata, based on the relevant input $T^{-1}(ai)$ and output $T(ai)$ nodal subnets of the DIS components topology;
- multi-agent decomposition task of behavioral testing of the DIS components, in particular, the synthesis of the component behavioral tests on a set of the agent testing $Task_T$, implementing $Task_R$, transporting $Task_{Tr}$ and dispatching $Task_S$;
- definition of the system of sets Rel_{Set}, -component -object Rel_{ObjCom}, structural Rel_{Struct}, functional Rel_{Func}, temporal Rel_{Time} relations, in particular, precedence relationships Rel_{Pref} for the agent-based tasks $Task_T$, $Task_R$, $Task_{Tr}$ and $Task_S$;
- construction of structure $G_{Task}(\{Task_T, Task_R, Task_{Tr}, Task_S\}, \Delta_{Task})$ of the agent tasks $Task_T$, $Task_R$, $Task_{Tr}$ and $Task_S$, based on a system of sets Rel_{Set}, component -object Rel_{ObjCom}, structural Rel_{Struct}, functional Rel_{Func}, temporal Rel_{Time} relations;
- resources definition $\{R_{TaskT}, R_{TaskR}, R_{TaskTr}, R_{TaskS}\}$, required for the structure $G_{Task}(\{Task_T, Task_R, Task_{Tr}, Task_S\}, \Delta_{Task})$ of the agent tasks $Task_T$, $Task_R$, $Task_{Tr}$ and $Task_S$ and available for the agents;
- forming of the component test experiments of deterministic $texp_{Di} \in TExp_D$, evolutionary $texp_{Ei} \in TExp_E$, combined $texp_{DEi} \in TExp_{DE}$ behavioral testing for the agents $ag_i \in Ag$ MAC;
- task scheduling $Task_T$, $Task_R$, $Task_{Tr}$ and $Task_S$, scheduling dispatching of the agent resources $\{R_{TaskT}, R_{TaskR}, R_{TaskTr}, R_{TaskS}\}$ on the basis of the technical solutions – input buffers of the tasks, dynamic priorities of the tasks, mechanisms of critical sections of resources, quantization of access, transaction for behavioral testing tasks, output buffers of solutions in accordance with the component test experiments $TExp = TExp_D \cup TExp_E \cup TExp_{DE}$.

4. AGENT MODEL CONSTRUCTION

Hybrid type agent ag_i, comprising two complementary components – reactive deterministic ag_{Ri} and deliberative evolutionary ag_{Di}, in general case is a system-five:

$$ag_i = (ag_{Ri}, ag_{Di}, X_i, Y_i, \Delta_i), \quad (1)$$

where ag_{Ri}, ag_{Di} correspondingly reactive and deliberative components – interacting subsystems, parallel (in environment) and/or sequential (in time), functioning within the corresponding component of DIS, sharing the inputs X_i (conditions-events) and outputs Y_i (actions) of the agent ag_i; Δ_i – general function of the components interaction.

The priority of choosing of the reactive ag_{Ri} or deliberative ag_{Di} component and its subsequent activization are defined by the solution of a random test $Task_T$ task, according to the type, complexity and condition of the corresponding component automata ai, representing a random component of DIS, the MAS condition and goal formation of the corresponding agent a_i.

The generality of the test data and knowledge models for the reactive ag_{Ri} or deliberative ag_{Di} components of the hybrid agent ag_i reduces the difference between them to the peculiarities of their methods of behavioral testing:

- additional – in a deterministic or evolutionary construction and recognition of identifiers Ti_i, tests Tp_i and links Lp_i primitives, in a deterministic or evolutionary construction of the inverse input implemented $T^{-1}(a_i)$ and direct output transported $T(a_i)$ nodal subnets, implemented $aR^{X}{}_{T}{}^{-1}{}_{(ai)}$ and transported $aTr^{Y}{}_{T(ai)}$ nodal sub-automata;
- basic – in a deterministic or evolutionary construction and recognition of the test fragments Tf_i and in a deterministic or evolutionary construction of the test experiments $TExp_i$.

Extended agent model ag_i (with reactive ag_{Ri} and/or deliberative ag_{Di} components) having information about its own input $T^{-1}(a_i)$ and output $T(a_i)$ nodal subnets NA, and about the implemented $aR^{X}{}_{T}{}^{-1}{}_{(ai)}$ and transportable $aTr^{Y}{}_{T(ai)}$ nodal subnets of sub-automata, is represented by:

$$ag_i=(M_i, Q_i, St_i, \{o_i, \sigma_i, \alpha_i\}, \{m_{0i}, q_{0i}, st_{0i}\}), \quad (2)$$

where M_i – a set of the agent-based models of the placement component of DIS; Q_i – a set of the agents goals, for certain $q_i \in Q_i$ defined as $q_i: M_i \times M_i \to D_i$,:

$$q_i(m_i, m_i') = (k_{\varphi\upsilon i}(\varphi\upsilon_i(m_i') - \varphi\upsilon_i(m_i)), k_{\lambda\varepsilon i}(\lambda\varepsilon_i(m_i') - \lambda\varepsilon_i(m_i)), k_{\theta\upsilon i}(\theta\upsilon_i(m_i') - (\theta\upsilon_i(m_i))),$$

where, $\varphi\upsilon_i$, $\lambda\varepsilon_i$, $\theta\upsilon_i$, $\rho\varepsilon_i$, $\tau\rho_i$ – detection functions of the check completeness, length, multiplicity, feasibility and portability of a test, m_i and m_i' – correspondingly the initial and the following estimated agent-based models of the placement component; St_i – a set of strategies of agent functioning, for a certain st_i defined as st_i: $M_i \to M_i$, $m_i' = st_i(m_i)$; $\{o_i, \sigma_i, \alpha_i\}$ – the signature of the agent operations, correspondingly:

a) observation $o_i: M_i \times V_i \to M_i$, $m_i = o_i(M_i, V_i)$, determining of the operation agent model and forming, if it is necessary, of the identifiers Ti_i and the test primitives Tp_i, where $V_i = (en_i, ag_i, Conn_i)$ – agent environment ag_i, for which:
   1) $en_i = (a_i, Pl_i)$ – agent-world environment for the agent ag_i;
   2) $Conn_i$ – relationship between the agent ag_i and the environment en_i;
   3) $Pl_i = \{pl_{ini}, pl_{outi}, pl_{inouti}, pl_{loopbacki}\}$ – a set of the basic agent placement for ag_i in environment en_i;

   b) implementation of the strategy $\sigma_i: St_i \times V_i \to V_i$, $V' = \sigma_i(st_i, V_i)$ – creation of modified environment V_i' and agent ag_i' in its structure, definition of link primitives Lp_i, construction of the test fragments Tf_i and definition of the completeness $\varphi\upsilon_i$, length $\lambda\varepsilon_i$, multiplicity $\theta\upsilon_i$, feasibility $\rho\varepsilon_i$ and portability $\tau\rho_i$ of a set of the received test fragments for the operation agent model m_i;

   c) adaptation $\alpha_i = (\alpha_{mi}, \alpha_{si})$, where $\alpha_{mi}: M_i \times M_i \to M_i$, $M_i' = \alpha_{mi}(M_i, m_i, m_i')$ and $\alpha_{si}: St_i \times M_i \to St_i$, $St_i' = \alpha_{mi}(St_i, m_i, m_i')$ – fixation of updating of the sets of strategies St_i and agent-based models of the component placement M_i, composed of:
   1) initial identifiers Ti_i and test primitives Tp_i;
   2) test fragments Tf_i;
   3) input implemented $aR^X{}_i$ and output transported $aTr^Y{}_i$ component semi-automata;
   4) input implementing $T^{-1}(a_i)$ and output transporting $T(a_i)$ nodal subnets;
   5) input implemented $aR^X{}_{T^{-1}(ai)}$ and output transported $aTr^Y{}_{T(ai)}$ nodal semi-automata;

- $\{m_{0i}, q_{0i}, st_{0i}\}$ – the initial model, goal and strategy of the agent.

A certain placement agent model $m_i \in M_i$ of the above mentioned set of models M_i is defined as:

$$m_i = (a_i, Ti_i, Tp_i, Tf_i, aR^X{}_i, aTr^Y{}_i, T^{-1}(a_i), T(a_i), aR^X{}_{T^{-1}(ai)}, aTr^Y{}_{T(ai)}), \quad (3)$$

where for a certain DIS component of the automata a_i:

- a_i – tested automata of the environment component, designated for the agent, which can be a sub-automata of the entire component automata model;
- Ti_i – initial identifiers of the support states and Tp_i – test primitives;
- Tf_i – test fragments;
- $aR^X{}_i$ – the component input implemented sub-automata and $aTr^Y{}_i$ – the component output transported sub-automata;
- $T^{-1}(a_i)$ – the node input implementing subnet of a network NA and $T(a_i)$ – the node output transporting subnet of a network NA;
- $aR^X{}_{T^{-1}(ai)}$ – the node input implemented semi-automata and $aTr^Y{}_{T(ai)}$ – the node output transported semi-automata.

The specificity of the functioning of the reactive component ag_{Ri} of an agent ag_i – the performance of its operations $\{o_i, \sigma_i, \alpha_i\}$, is based on the deterministic test methods with the search to depth or/and width. These methods are applied to achieve a local-exhaustive optimization. They are based on the automata experiments [5, 9, 10] and characterized by *NP*-complexity of testing, in particular, by the synthesis of behavioral tests.

This fact implies the restriction of the analysis (a subset of the components of DIS) by the medium complexity (up to 1000 states) to obtain the solution of the test tasks, as a rule, of the required high completeness $\varphi\upsilon_i$, acceptable length $\lambda\varepsilon_i$ and multiplicity $\theta\upsilon_i$, with time $\tau\iota_i$ and memory $\mu\varepsilon_i$, limited by upper bounds of dedicated computing resources $R_{agRi} = (\tau\iota_{MaxagRi}, \mu\varepsilon_{MaxagRi})$ of reactive component ag_{Ri}.

The specificity of the functioning of the deliberative component ag_{Di} of the agent ag_i – the performance of its operations $\{o_i, \sigma_i, \alpha_i\}$ is based on the pseudo-random, goal-oriented test search of the evolutionary-genetic approach [27]. Therefore, for automata test experiments, these methods retain the upper exponential analytical evaluations of deterministic methods, but their experimental complexity is significantly less than computed *NP*-complexity.

This leads to a solution of tasks of the acceptable composition with resources of time and memory, less than of the middle range, which is limited by lower and upper boundaries of resources $R_{agDi} = (\tau\iota_{MaxagDi}, \mu\varepsilon_{MaxagDi})$ of deliberative component, in the space of analysis (a subset of the components of DIS) of objects which is above the average degree of complexity (more than 1000 automata states).

Intelligence of the agent ag_i (for its deliberative component ag_{Di}) is based on the evolutionary systems of behavioral tests synthesis [27]:

$$Te_i=(Tf_i,\ Tp_i,\ Lp_{ti},\ Sg_{ti},\ Tf_{fi}), \quad (4)$$

where Tf_i, Tp_i, Lp_{ti} – a set of test fragments, the initial set of test primitives $Tp_i=Tf_{0i}\subseteq Tf_i$, linking primitives Lp_{ti}, Tf_{fi} – the final set of test fragments; $Sg_{tagDi}=\{\mu_{tagDi},\ \kappa_{tagDi},\ \phi_{tagDi},\ \varphi_{tagDi},\ \sigma_{tagDi}\}$ – signature of the operations and functions of the test evolution, consisting of the test mutation, crossover, immunity, fitness and selection functions.

Besides the test objects of population T_{+i}, the model proposes syntactically and functionally similar to them additional infectious and vaccine objects [27]. Infectious objects $\Omega_{+i}=\Omega t_i\cup\Omega p_i\cup\Omega f_i$, as test objects, may include identifiers Ωt_i, primitives Ωp_i, fragments Ωf_i and the population, as a whole Ω_{+i}, and form a system of fragments with signature of operations and functions in the evolution Ωe_i. Infectious objects Ω_{+i} are generated within the infectivity evolution, external for the test evolution.

Formation and formal representation of infectious objects Ω_{+i} is similar to the presentation and formation of the relevant test objects T_{+i}. However, it is characterized by its own, though similar, definitions, operations and functions for the infectious objects.

Infectious objects follow $\Omega_{+i}\subseteq W_i''^{\wedge}$, $W_i''^{\wedge}$ – the behavior of the infectious automata $a_i^{\wedge}$, for which discrepancy (or non inclusion) is possible in relation to the tested automata a_i, that is, it is possible $W_i''^{\wedge}\neq W_i''$ (or $W_i''^{\wedge}\backslash W_i''\neq\varnothing$).

Infectious objects Ω_{+i} have a form, similar to their analogs – test objects from T_{+i}:

$$\omega t_i^{S^{\wedge}}=(T_i,\ W_i''^{\wedge},\ S_i^{\wedge},\ \Delta_i,\ \Lambda_i^{S^{\wedge}}), \quad (5)$$

$$\omega p_i^{W^{\wedge}}=(T_i,\ W_i''^{\wedge},\ S_i^{\wedge},\ \Delta_i,\ \Lambda_i^{S^{\wedge}},\ \Omega t_{tpi}^{S^{\wedge}},\ W_i''^{\wedge SE}), \quad (6)$$

$$\omega f_i^{W^{\wedge}}=(T_i,\ W_i''^{\wedge},\ S_i^{\wedge},\ \Delta_i,\ \Lambda_i^{S^{\wedge}},\ \Omega p_{tfi}^{S^{\wedge}},\ Lp_{tfi}^{S^{\wedge}},\ W_i''^{\wedge SE}), \quad (7)$$

where models components are presented with the definition of the test objects, the model of the infectious evolution synthesis for $a_i^{\wedge}$ is defined as infectious evolution of the common type with the test evolution:

$$\Omega e_i=(\Omega f_i,\ \Omega p_i,\ Lp_{\omega i},\ \Omega i_i,\ Sg_{\omega i}), \quad (8)$$

where Ωf_i, Ωp_i, $Lp_{\omega i}$, Ωi_i – are presented above the sets of correspondingly infectious fragments, initial set (in evolution) of infectious primitives $\Omega p_i=\Omega f_{0i}\subseteq\Omega f_i$ and linking primitives, a set of infectious identifiers $\Omega i_i\neq W_i''$ (or $\Omega i_i\backslash W_i''\neq\varnothing$); $Sg_{\omega i}=\{\mu_{\omega i},\ \kappa_{\omega i},\ \varphi_{\omega i},\ \sigma_{\omega i}\}$ – signature of infective evolution, such as a test signature.

For infectious Ωe_i and test Te_i evolutions the interaction of certain objects – identifiers, primitives, fragments – are based on monobasic (inside Ωe_i or Te_i) operations of crossover $\{\kappa_{ti},\ \kappa_{\omega i}\}$ or dibasic $(\Omega e_i\times Te_i)$ operations of mutation $\{\mu_{ti},\ \mu_{\omega i}\}$. Operations are preceded by definition of the fitness $\{\varphi_{ti},\ \varphi_{\omega i}\}$ and selection $\{\sigma_{ti},\ \sigma_{\omega i}\}$ functions with deterministic and pseudorandom priority settings Π_{ti}, Σ_{ti}, $\Pi_{\omega i}$, $\Sigma_{\omega i}$.

The vaccine objects – identifiers $\Psi t_i\subseteq W_i''^{\wedge}$, primitives $\Psi p_i\subseteq W_i''^{\wedge}$, fragments $\Psi f_i\subseteq W_i''^{\wedge}$ and population $\Psi_{+i}\subseteq W_i''^{\wedge}$ – are located in the space $W_i''^{\wedge}$, which is enlarged correspondingly to a_i, as well as infectious objects, generating them (knowledge and recognition of infections). The vaccine objects, expanding the conventional test objects, accumulate a successful immune experience. We can talk about the inclusion of $\Psi t_i\subseteq Tt_i^{\sim}$, $\Psi p_i\subseteq Tp_i^{\sim}$, $\Psi f_i\subseteq Tf_i^{\sim}$, $\Psi e_i\subseteq Te_i^{\sim}$ and in general $\Psi_{+i}\subseteq T_{+i}^{\sim}$, where $\Psi_{+i}=\Psi t_i\cup\Psi p_i\cup\Psi f_i$, into the context of the current status of the population $T_{+i}^{\sim}$ of the evolution development Te_i, because the Ψ_{+i} is a part of test T_{+i}, formed by infections.

Therefore, the complex model of the test Ce_i synthesis for a_i is defined as the co-evolution – internal complex evolution in the space of test Te_i and infectious Ωe_i evolutions:

$$Ce_{\Omega i}=(Te_i,\ \Omega e_i,\ \Psi_{\Omega+i},\ \{\mu_{\psi i},\ \kappa_{\psi i}\},\ \Phi_{\Omega i},\ Tf_{fi}), \quad (9)$$

with the selection of vaccine population Ψ_{+i} in Te_i, a signature of the internal mutations operations $\mu_{\psi i}\subseteq\mu_{ti}\cup\mu_{\omega i}$ and presumably a crossover $\kappa_{\psi i}\subseteq\kappa_{ti}\cup\kappa_{\omega i}$, immune search function $\Phi_{\Omega i}=\{\phi_{\Omega+i\sim},\ \phi_{T+\sim i(\Omega+\sim i)},\ \phi^{-1}_{Pre\mu\psi i}\}$, and the final set of test fragments (test population) Tf_{fi}.

Agent cooperation of evolutions for the test Te_i or viral-test Ce_i is possible as external interactions with evolutions of the other agents, placed on the other components of DIS.

In this case, the feature of "bonding" of the fragments in the operations of the new external mutation $\mu_{\psi i}$' and crossover $\kappa_{\psi i}$' can be shown through the identity or compatibility (intersection) of the adjacent projections *«pr»* of basic behaviors of adjacent components in the composition NA – MAS pairs of agents, such as:

- test output $pr_2(Tf_{1i})$ for a_{1i} automata of previous *1i*-component and the test input $pr_1(Tf_{2i})$ for a_{2i} automata of *2i*-current component – $pr_2(Tf_{1i})=pr_1(Tf_{2i})$;
- implemented output $pr_2(Rf_j)$ for the automata a_j of the previous *j*-component and test input $pr_1(Tf_i)$ for automata a_i of current *i*-component - $pr_2(Rf_j)=pr_1(Tf_i)$;
- test output $pr_2(Tf_i)$ for automata a_i *of* current *i*-component and transported input $pr_1(Trf_k)$ for

automata a_k of the next k-component – $pr_2(Tf_i)=pr_1(Trf_k)$.

In addition, new external functions of fitness $\varphi_{\psi i}'$ and selection $\sigma_{\psi i}'$ include not only the examination of the criteria of completeness $\varphi\upsilon_i$, length $\lambda\varepsilon_i$ and multiplicity $\theta\upsilon_i$ of the test patches and multiplicity $\theta\upsilon\varepsilon_i$ of population, but also the criteria of feasibility $\rho\varepsilon_i$, and portability $\tau\rho_i$. In the simplest case, these criteria are of the semaphore format, restricting or blocking the application of implemented and/or transported fragments.

In this case, the co-evolutionary interaction is identical with the co-evolutionary viral-test interaction with the allocation of vaccine population Ψ_{+i}, $\mu_{\psi i}'\subseteq\mu_{t1i}'\cup\mu_{t2i}'$ in Te_i , signatures of external operations of mutation and presumably a crossover $\kappa_{\psi i}'\subseteq\kappa_{t1i}'\cup\kappa_{t2i}'$, immune search function $\Phi_{Ti}=\{\phi_{T+\sim i}, \phi_{T+\sim 1i(T+\sim 2i)}, \phi^{-1}_{Pre\mu\psi i}\}$, fitness $\varphi_{\psi i}'\subseteq\varphi_{t1i}'\cup\varphi_{t2i}'$, and selection $\sigma_{\psi i}'\subseteq\sigma_{t1i}'\cup\sigma_{t2i}'$ functions and the fragments Tf_{fi} of final population T_{+fi} of test evolution Te_i [28]:

$$Ce_{T12i}=(Te_{1i}, Te_{2i}, \Psi_{T+i}, \{\mu_{\psi i}', \kappa_{\psi i}'\}, \Phi_{Ti}, Tf_{fi}), \quad (10)$$

At first, input implementing Re_j and transporting Tre_k evolutions are formed in co-evolutions Ce_{RjTi} and Ce_{TiTrk} for Te_i. This creates input implementing $T^{-1}(a_i)$ and output transporting $T(a_i)$ as two-level nodal subnets, and input implemented $aR^{X}{}_{T}^{-1}{}_{(ai)}$ and output transported $aTr^{Y}{}_{T(ai)}$ as nodal semi-automata.

After this construction or simultaneously with it, co-evolutions Ce_{RjTi} and Ce_{TiTrk} are formed, correspondingly as implemented and transported restrictions of the evolution Te_i. Then at the same time implemented and transported restrictions of Ce_{Ti} of evolution Te_i appear as the intersection of co-evolutions Ce_{RjTi} and Ce_{TiTrk} with their possible optimization - common objects determination and their dual use (serial or parallel) as:

- identifiers Ti_i;
- primitives Tp_i and Lp_i;
- fragments Tf_i;
- two-level subnets $t^{-1}(a_i)$, $t(a_i)$ in subnets $T^{-1}(a_i)$, $T(a_i)$;
- two-level sub-automata $ar^{X}{}_{T}^{-1}{}_{(ai)}$, $atr^{Y}{}_{T(ai)}$ in semi-automata $aR^{X}{}_{T}^{-1}{}_{(ai)}$, $aTr^{Y}{}_{T(ai)}$, according to the sub-networks $t^{-1}(a_i)$, $t(a_i)$.

Appropriate signatures of search functions Φ_{Ti}, Φ_{RjTi}, Φ_{TiTrk} are identical with the signatures of the immune search generic function Φ_{Ti} of internal co-evolution Ce_{Ti} for external cooperative co-evolutions Ce_{T12i}, Ce_{RjTi}, Ce_{TiTrk} and test evolutions Te_{1i}, Te_{2i}. They provide recognition of input operands of component mutation μ_{ti}' and a crossover κ_{ti}'. It is possible when there is an appropriate experience which can be stored. Therefore, the prepared (stored) results (Ti_i, Tp_i, Lp_i, Tf_i, $T^{-1}(a_i)$, $T(a_i)$, $aR^{X}{}_{T}^{-1}{}_{(ai)}$, $aTr^{Y}{}_{T(ai)}$) can be applied.

Thus, the cooperativeness of the component agent ag_i, its ability to participate in forming and providing of the internal Ce_i and external Ce_{T12i}, Ce_{RjTi}, Ce_{TiTrk} cooperation is based on the construction of component minimized input implemented $aR^{X}{}_{T}^{-1}{}_{(ai)}$ and output transported $aTr^{Y}{}_{T(ai)}$ nodal semi-automata. They are the key elements of the implemented and transported behavior passing forward and backward in the NA composition through the DIS components.

Therefore, the testing model MAS (of the second level) is a cooperation of Ce_i, Ce_{T12i}, Ce_{RjTi}, Ce_{TiTrk} of the hybrid agents $Ag=\cup_{i\in I} ag_i$. In accordance with the problem, which has to be solved, this model deterministically and/or evolutionarily forms and activates the required test structure into a set of reactive Ag_R and deliberative Ag_D components of the agents Ag. These agents are placed in automata of NA – in the DIS components.

The purpose of the external test cooperation of co-evolutions Ce_{T12i}, Ce_{RjTi}, Ce_{TiTrk} is tetra-evolution $Ce_{RjTiTrk}$, as a joint cooperation of agents Ce_i, Ce_{T12i}, Ce_{RjTi}, Ce_{TiTrk}. Tetra-evolution $Ce_{RjTiTrk}$ is the formal representation of the structural and topological, implemented and transported, goal-oriented and intelligent performance of the tasks during the operation and test control of the DIS components.

5. IMPLEMENTATION AND SIMULATION

The basic component-object (programming technology) programs in MS Visual.Net environment were offered for the experimental implementation of the agent model ag_i, with the reactive ag_{Ri} and deliberative ag_{Di} components (parts of the agent), which model behavioral testing of the DIS component, performed by the MAS agent. The deterministic and evolutionary-genetic generators and the agent supervisor are selected as the first stage program.

Simulation of deterministic generator is based on a set of component implementations of deterministic models and methods of passive and active synthesis of behavioral tests Tex_i, which form a technological structure (system, conditional procedure) of deterministic construction of the test fragments Tf_i (see Fig. 1). The composition and communication of this structure are defined by the solved deterministic tasks and the relations between them.

The set of basic procedures defines the operation of the deterministic generators during the performance of the test tasks.

The general procedure of the preprocessor test synthesis of behavioral tests Tex_i for automata model a_i of DIS component includes:

1) preliminary structural and topological (graph) analysis with the detection in the graph of the component automata a_i of inputs, outputs, chains (straight-line paths), trees, hammocks, cycles, the formation of condensation with folded chains and cycles;
2) preliminary definition of identifiers Ti_i supporting states (*NP*-complexity), linking Lp_i and test Tp_i primitives based on them; preview definition of the input a^X_i and output a^Y_i component semi-automata;
3) preliminary definition of the transported (recognizable) component sub-automata aTr^Y_i (*NP*-complexity).

The general procedure of the preprocessor test synthesis of behavioral tests for the model of the DIS automata network includes:

1) preliminary structural and topological (graph) analysis, identifying the inputs, outputs, chains (straight-line paths), trees, hammocks, cycles, the formation of condensation with folded chains and cycles in a graph of the automata network NA;
2) forward and reverse recursive construction of the implementing $T^{-1}(a_i)$ and transporting $T(a_i)$ nodal subnets;
3) forward recursive construction of the implementable nodal semi-automata $a^{R}_{X}{}^{T-1(ai)}$ based on implementing nodal subnets $^{T-1}(_{ai})$;
4) reverse recursive construction of the transported nodal semi-automata $aTr^{Y}_{T(ai)}$ based on transporting nodal subnets $T(a_i)$;
5) narrowing of the identifiers Ti_i of the supporting states, linking Lp_i and test Tp_i primitives – receiving of the implemented and transported identifiers $Ti_{RjTiTrk}$, linking $Lp_{RjTiTrk}$ and test $Tp_{RjTiTrk}$ primitives based on the implemented $aR^{X}{}_{T}{}^{-1}{}_{(ai)}$ and transported $aTr^{Y}{}_{T(ai)}$ of the nodal semi-automata.

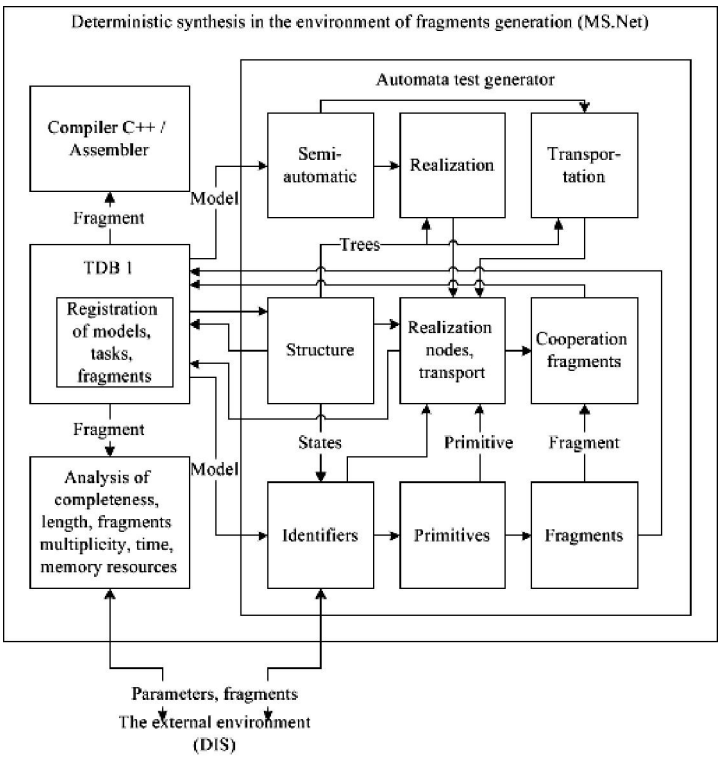


Fig. 1 – Block diagram of a deterministic generator.

Recursive steps of the procedures are performed due to the precedence of the automata models of the DIS structure components, based on the search into the depth/width with the local optimization, accumulating (providing) nodal results.

Preprocessor procedures provide nodal results only once. These results can be reused in order to accelerate the synthesis of behavioral tests.

The basic procedure of deterministic one-fold active synthesis of behavioral tests Tex_i and repetitive subsequent testing, performed in test mode for the automata model of the tested DIS component, includes:

1) construction of the test fragments $Tf_{RjTiTrk}$ and synthesis of the complete behavioral test Tex_i modified by the search into the depth/width with local optimization, for example, pseudo-Euler crawl, on the base of narrowed identifiers $Ti_{RjTiTrk}$, link $Lp_{RjTiTrk}$ and test $Tp_{RjTiTrk}$ primitives;
2) recursive building of the inverse compliances $Tf_{RjTiTrk}$ and complete behavioral test $Tex_{RjTiTrk}$ into the relevant structures of input actions – input test semi-automata $T^{-1}(a_i)(Tf_{iY})$ and $T^{-1}(a_i)(Tex_i)$ for implementing nodal subnet of automata model a_i of tested component;

3) recursive building of the direct mapping of test fragments $Tf_{RjTiTrk}$ and complete behavioral test $Tex_{RjTiTrk}$ into the relevant structures of output reactions – output test semi-automata $T(a_i)(Tf_{iY})$ and $T(a_i)(Tex_i)$ for transporting nodal subnet $T(a_i)$ of automata model a_i of the tested component.

The basic procedure of deterministic repeated passive synthesis of behavioral tests Tex_i and testing, executed in the operation mode for the automata model a_i of the tested DIS component includes:

1) real time input-buffering in a set of analyzed input/output words W"^ based on the current operation functioning of a component;
2) deterministic treelike search-recognition of identifiers $Ti_{RjTiTrk}^{\sim}$ of supporting states, linking $Lp_{RjTiTrk}^{\sim}$ and test $Tp_{RjTiTrk}^{\sim}$ primitives, implemented and transported in the automata network on a set of input/output words W"^ based on identifiers $Ti_{RjTiTrk}$, primitives $Lp_{RjTiTrk}$ and $Tp_{RjTiTrk}$, which were previously built by the pre-preprocessor procedure;
3) formation of pseudo-test fragments $Tf_{RjTiTrk}^{\sim}$ (candidates) and structure – automata $a_i^{\sim}$ – of the input/output behavior, based on the identification of the supporting states and the existing determinism of automata functions a_i;
4) deterministic treelike search-recognition of linking $Lp_{RjTiTrk}$ and test $Tp_{RjTiTrk}$ primitives, test fragments $Tf_{RjTiTrk}$ based on the formed structure – automata $a_i^{\sim}$ – of input/output behavior W"^;
5) registration of recognized test fragments $Tf_{RjTiTrk}$ and evaluation of private and general completeness $\varphi\upsilon_i$, length $\lambda\varepsilon_i$, multiplicity $\theta\upsilon_i$ and $\theta\upsilon\varepsilon_i$, feasibility $\rho\varepsilon_i$ and portability $\tau\rho_i$ of testing.

The preprocessor procedure of deterministic implementation of the behavioral tests Tex_i and testing, performed in the operation and test modes for automata model a_i of the tested DIS component in case of forward recursive construction of implemented nodal semi-automata $aR^{X}{}_{T^{-1}(ai)}$, includes:

1) deterministic alternative, based on the search into the depth/width of the current following unexplored automata model $a_i^{\sim}$ with the lowest number in the input nodal reverse subnet $T^{-1}(a_i)$ for automata model a_i of the tested component;
2) input/buffering of output nodal implemented semi-automata $aR^{Y}{}_{T^{-1}(ai^{\sim})}$ of the current reverse nodal subnet $T^{-1}(a_i^{\sim})$ for the current automata model $a_i^{\sim}$;
3) alphabetic mapping of the output nodal implemented semi-automata $aR^{Y}{}_{T^{-1}(ai^{\sim})}$ of the current nodal reverse subnet $T^{-1}(a_i^{\sim})$ into the input nodal implemented semi-automata $aR^{X}{}_{i}^{\sim}$ of the current automata model $a_i^{\sim}$, representing a set of input words, available in the current input nodal reverse subnet $T^{-1}(a_i^{\sim})$;
4) composition $aR^{X}{}_{i}^{\sim} \circ a_i^{\sim}$ of the current input nodal implemented semi-automata $aR^{X}{}_{i}^{\sim}$ and current automata model $a_i^{\sim}$, which generates the modified implemented current automata model $a_i^{\sim}{}_{Rj}$;
5) output narrowing $\downarrow_Y(a_i^{\sim}{}_{Rj})$ of the modified implemented current automata model $a_i^{\sim}{}_{Rj}$, resulting into the output nodal implemented semi-automata $a_i^{\sim Y}{}_{Rj}$ at the output of current automata model $a_i^{\sim}$;
6) minimization $min(a_i^{\sim Y}{}_{Rj})$ of current output nodal implemented semi-automata $a_i^{\sim Y}{}_{Rj}$;
7) if not all of the automata models in the input nodal reverse subnet are considered, then go to step 1).;
8) alphabetic mapping of the output nodal implemented semi-automata $aR^{Y}{}_{T^{-1}(ai)} = a_i^{\sim Y}{}_{Rj}$ of the reverse nodal subnet $T^{-1}(a_i)$ of the automata model a_i of the tested component into its input nodal implemented semi-automata $a_i^{X}{}_{Rj}$, representing a set of input words, available in the input nodal reverse subnet $T^{-1}(a_i)$.

The preprocessor procedure of deterministic transportation of behavioral tests Tex_i and the testing, executed in the operation and test modes for automata model a_i of the tested DIS component in case of the reverse recursive construction of transported nodal semi-automata $aTr^{Y}{}_{T(ai)}$, includes:

1) deterministic alternative, based on the search into the depth/width of the current following unexplored automata model $a_i^{\sim}$ with the largest number in the output direct nodal subnet $T(a_i)$ for the automata model a_i of the tested component;
2) input/buffering of the input nodal transported semi-automata $aTr^{X}{}_{T(ai^{\sim})}$ of the current nodal direct subnet $T(a_i^{\sim})$ for the current automata model $a_i^{\sim}$;
3) alphabetic mapping of input nodal transported semi-automata $aTr^{X}{}_{T(ai^{\sim})}$ of the current nodal direct subnet $T(a_i^{\sim})$ into the output nodal transported semi-automata $aTr^{Y}{}_{i}^{\sim}$ of current automata model $a_i^{\sim}$, that represents a set of output words, transported in current output nodal direct subnet $T(a_i^{\sim})$;
4) composition $a_i^{\sim} \circ aTr^{Y}{}_{i}^{\sim}$ of the current automata model $a_i^{\sim}$ and output nodal transported semi-automata $aTr^{Y}{}_{i}^{\sim}$, which generates a modified transported automata model $a_i^{\sim}{}_{Trk}$;
5) input narrowing $\downarrow_X(a_i^{\sim}{}_{Trk})$ of modified transported current automata model $a_i^{\sim}{}_{Trk}$ resulting into the input nodal transported semi-automata $a_i^{\sim X}{}_{Trk}$ at the input of current automata model $a_i^{\sim}$;
6) minimization $min(a_i^{\sim X}{}_{Trk})$ of current input nodal transported semi-automata $a_i^{\sim X}{}_{Trk}$;
7) if not of the all automata models in the output nodal direct subnet are considered, then go to step 1);

8) alphabetic mapping of the input nodal semi-automata $aTr^{X}_{T(ai)}=a_i^{\sim X}{}_{Trk}$ of the direct nodal subnet $T(a_i)$ of the automata model a_i of the tested component in its output nodal semi-automata $a_i{}^{Y}{}_{Trk}$, representing a set of output words, transported in the output nodal direct subnet $T(a_i)$.

Modeling of the evolution generator is based on the expanded set of component implementations of evolutionary-genetic models and methods of passive and active synthesis of behavioral tests Tex_i, forming the technological evolution structure of the test fragments Tf_i (see Fig. 2). The composition and communications of structure are defined by the solved evolutionary tasks and relationships between them.

A set of additional procedures for the test tasks execution by the evolutionary generator has a number of features.

The general procedure of preprocessor test synthesis of behavioral tests Tex_i for automata model a_i of DIS component is complemented by:

1) the evolutionary definition of identifiers Ti_i of supporting states, linking Lp_i and test Tp_i primitives on their basis;
2) the evolutionary definition of transportable (recognizable) sub-automata aTr^{Y}_{i}.

The general procedure of preprocessor test synthesis of behavioral tests for the DIS automata network model is supplemented by:

1) evolutionary direct and reverse construction of correspondingly implementing $T^{-1}(a_i)$ and transporting $T(a_i)$ nodal subnets;
2) direct evolutionary construction of implementable nodal semi-automata $aR^{X}{}_{T}{}^{-1}{}_{(ai)}$ on the basis of the implementing nodal subnets $T^{-1}(a_i)$;
3) reverse evolutionary construction of transported nodal semi-automata $aTr^{Y}{}_{T(ai)}$ on the basis of the transporting nodal subnets $T(a_i)$.

Evolutionary steps of procedures are executed due to the precedence of the automata models of the DIS structure components, based on the pseudo-random targeted search, with the accumulation (provision) of nodal results.

When the evolutionary generator is used, nodal results are initially provided by the execution of the preprocessor procedures in order to reuse them for the purpose of acceleration of the synthesis of behavioral tests and development.

The basic procedure of evolutionary one-fold active synthesis of behavioral tests Tex_i and repetitive subsequent testing, executed in the test mode of the automata model of the tested DIS component, is supplemented by:

1) evolutionary construction of the test fragments $Tf_{RjTiTrk}$ and synthesis of behavioral test Tex_i due to the pseudo-random targeted search, based on the restricted identifiers $Ti_{RjTiTrk}$, linking $Lp_{RjTiTrk}$ and test $Tp_{RjTiTrk}$ primitives;
2) evolutionary construction of the inverse mapping of test fragments $Tf_{RjTiTrk}$ and behavioral test $Tex_{RjTiTrk}$ in the relevant structures of input actions – input test semi-automata $T^{-1}(a_i)(Tf_{iY})$ and $T^{-1}(a_i)(Tex_i)$ in the operating nodal subnet $T^{-1}(a_i)$ of the automata model a_i of the tested component;
3) evolutionary building of the direct mapping of the test fragments $Tf_{RjTiTrk}$ and behavioral test $Tex_{RjTiTrk}$ into the relevant structures of output reactions – output test semi-automata $T(a_i)(Tf_{iY})$ and $T(a_i)(Tex_i)$ in the transporting nodal subnet $T(a_i)$ of the automata model a_i of the tested component.

The basic procedure of the evolutionary repetitive passive synthesis of behavioral tests Tex_i and testing, executed in the operation mode of the automata model a_i of the tested DIS component is supplemented by:

1) evolutionary search-recognition of identifiers $Ti_{RjTiTrk}{}^{\sim}$ of supporting states, linking $Lp_{RjTiTrk}{}^{\sim}$ and test $Tp_{RjTiTrk}{}^{\sim}$ primitives, implemented and transported in the automata network, on a set of input/output words W''^, based on identifiers $Ti_{RjTiTrk}$, primitives $Lp_{RjTiTrk}$ and $Tp_{RjTiTrk}$, previously built according to the preprocessor procedure;
2) evolutionary search-recognition of linking $Lp_{RjTiTrk}$ and test $Tp_{RjTiTrk}$ primitives, test fragments $Tf_{RjTiTrk}$, based on the formed structure – automata $a_i^{\sim}$ – of input/output behavior W''^.

The preprocessor procedure of evolutionary implementation of behavioral tests Tex_i and testing, executed in the operation and test modes for automata model a_i of the tested DIS component in case of direct evolutionary construction of implemented nodal semi-automata $aR^{X}{}_{T}{}^{-1}{}_{(ai)}$, is complemented by pseudo-random targeted search of the current following unexplored automata model $a_i^{\sim}$ with the best evolutionary criteria in the input nodal reverse subnet $T^{-1}(a_i)$ of automata model a_i of the tested component.

The preprocessor procedure of evolutionary transportation of behavioral tests Tex_i and testing, executed in the operation and test modes for automata model a_i of the tested DIS component in case of the reverse evolutionary construction of transported nodal semi-automata $aTr^{Y}{}_{T(ai)}$, is supplemented by a pseudo-random targeted search of the current following unexplored automata model $a_i^{\sim}$ with the best evolutionary criteria in the output nodal direct subnet $T(a_i)$ of automata model a_i of the tested component.

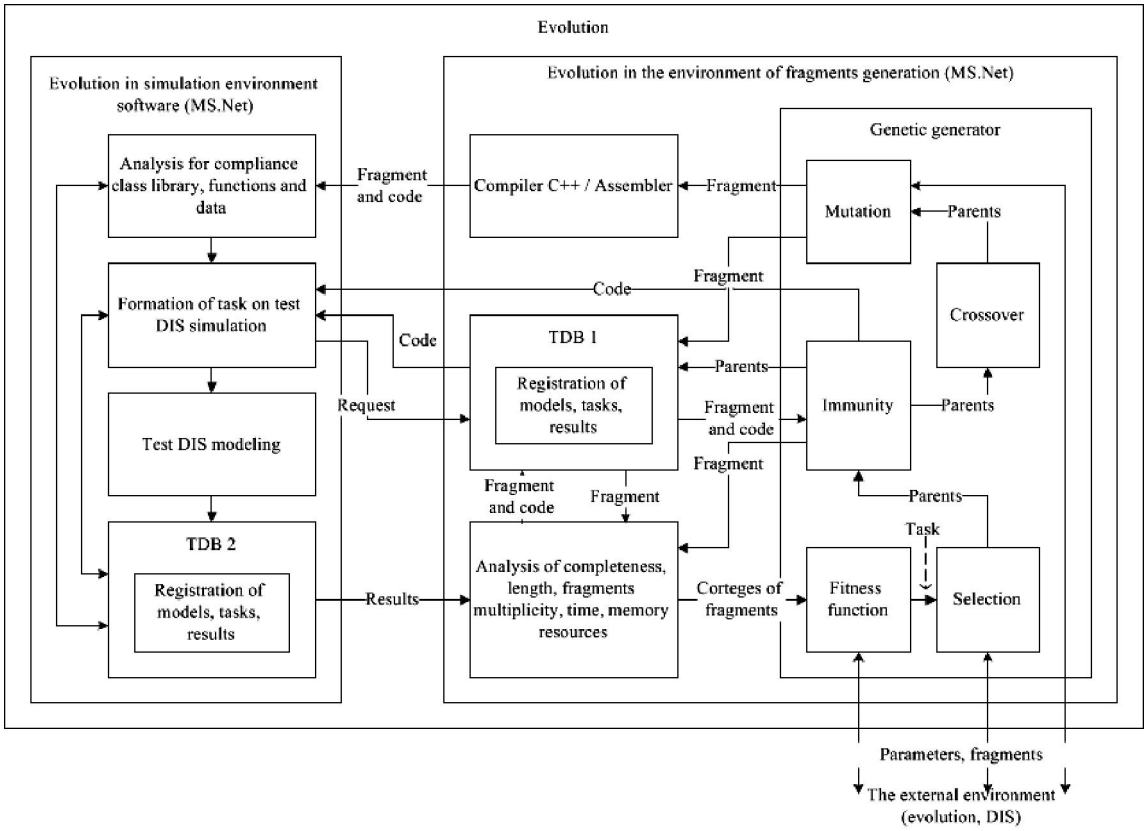


Fig. 2 – Block diagram of an evolutionary generator.

A priori and a posteriori value assessment of the criteria of completeness $\varphi\upsilon_i$, length $\lambda\varepsilon_i$, multiplicity $\theta\upsilon_i$, $\theta\upsilon\varepsilon_i$, feasibility $\rho\varepsilon_i$ and portability $\tau\rho_i$, defining the synthesis of behavioral tests Tex_i, as the conditional control and/or recognition experiment, is applied for the source data and for the received intermediate and final results of deterministic and evolutionary generators.

The agent supervisor ensures the consistent placement and diagnostic functioning of the agent in the environment of the selected DIS component, consistent with its operation functioning.

The basic supervisor procedure of the functioning of an agent ag_i is characterized by the following actions:

1) Direct (on behalf of the agent ag_i) or indirect (on behalf of MAS) identification of the target DIS component for agent ag_i.
2) Remote, direct (on behalf of the agent ag_i) or indirect (on behalf of the MAS) authentication of an agent ag_i in the DIS component.
3) Remote, direct or indirect preliminary coordination of required computing resources of the agent a_{gi}, in particular, of reactive a_{gRi} and deliberative a_{gDi} components.
4) The internal (agent ag_i) or external (MAS) control of the agent ag_i transportation to the selected DIS component through the network DIS environment, in particular, the network of automata NA.
5) If a set of test tasks $Task_i$ of the agent ag_i is not empty, then the formation of the next test task $task_i \in Task_i$ with the initialization of the iterative cycle of the agent ag can take place, otherwise there is the transition to n. 13.
6) Initialization of iteration of the agent ag_i, as the formation of the agent model of placement $m_i=(a_i, Ti_i, Tp_i, Tf_i, aR^X_i, aTr^Y_i, T^{-1}(a_i), T(a_i), aR^X{}_{T^{-1}(ai)}, aTr^Y{}_{T(ai)})$ (on the initial step – initial model of placement m_{0i}) and its test criteria of completeness $\varphi\upsilon_i$, length $\lambda\varepsilon_i$, multiplicity $\theta\upsilon_i$ and $\theta\upsilon\varepsilon_i$, feasibility $\rho\varepsilon_i$ and portability $\tau\rho_i$.
7) If the test task $task_i$ of iterative cycle is solved and the final values for the test criteria of completeness $\varphi\upsilon_i$, length $\lambda\varepsilon_i$, multiplicity $\theta\upsilon_i$ and $\theta\upsilon\varepsilon_i$, feasibility $\rho\varepsilon_i$ and portability $\tau\rho_i$ are received, then there is the transition to n. 5.
8) Formation of the environment en_i of the agent-world in the DIS component, as a part of its tested automata a_i and a set of basic places Pl_i, the agent's ag_i own model, and also the connection $Conn_i$ between the agent ag_i and the environment en_i, defining a set of the boundary computing resources Res_i of a component for the corresponding basic places Pl_i.
9) The current authorization of the agent ag_i in the DIS component.
10) Specification of the final q_i^F and formation of the current q_i goals of the agent ag_i, according to the analysis of the status of a model m_i

concerning the part Ti_i, Tp_i, Tf_i, aR^X_i, aTr^Y_i, $T^{-1}(a_i)$, $T(a_i)$, $aR^X_{T^{-1}(ai)}$, $aTr^Y_{T(ai)}$, and also due to the current and final test criteria of completeness $\varphi\upsilon_i$, length $\lambda\varepsilon_i$, multiplicity $\theta\upsilon_i$ and $\theta\upsilon\varepsilon_i$, feasibility $\rho\varepsilon_i$ and portability $\tau\rho_i$. $\rho\varepsilon_i$

11) Formation and execution of the strategy st_i of the current goal q_i achievement as a choice of a system of the interconnected deterministic and/or evolutionary steps (in the simplest case – one step) for creation of the following model $m_i'=(a_i, Ti_i', Tp_i', Tf_i', aR^X_i', aTr^Y_i', T^{-1}(a_i)', T(a_i)', aR^X_{T^{-1}(ai)}', aTr^Y_{T(ai)}')$ according to the current goal q_i.
12) The transition to n. 7.
13) Completion of the current session of the agent ag_i with the expectation of the event of non-empty set of the test tasks $Task_i$.

Individual program components, their properties, methods, and interfaces can be applied separately. Being included into the parallel diagnostic components, they can be used at the request of this agent and other related MAS agents. Cooperation of the components is specifically determined by the agents, initiating the testing, taking into account the criteria of the results assessment – completeness of the error checking, length, multiplicity, the tests feasibility and portability, computing costs, depending on the allocation environment.

At the system, program and information level of modeling, the last task is performed due to the use of distributed input buffers (cache) of the agents tasks, dynamic (the current state) priorities of the tasks, mechanisms of the critical sections of resources, quantization of access to them, transactions in the tasks of behavioral testing and output buffers (cache) of the agents solutions.

At the transportation level, adjacent to the application test level below, interaction of the cooperation agents in the DIS environment can be provided by its own component means of communication, the virtualized for MAS agents as the external transportation interface components.

Functions of the libraries of the MS Visual.Net program modeling environment and special agent libraries, in particular, are used for implementation of the basic programs of generators and supervisor.

Program modeling results allowed to verify the models and evaluate the area of their applicability, in particular, the possibility of transportation of the *NP*-complexity of the synthesis of behavioral tests into the accessible area (see. Fig. 3).

The time of synthesis was reduced, but the length and the tests completeness remained the same, with the use of multi-agent decomposition and evolutionary methods. The time of the test synthesis has been reduced to 92% (from 95 minutes (100%) to 8 minutes (8%)) for the experimental DIS mechanisms at the complexity level IPv.4, IPv.6 and IPSec, applying correspondingly the deterministic and evolutionary methods.

Multi-agent synthesis of behavioral tests and testing based on verification of the functional mapping of automata models of the DIS components increases the multiversion of the problem solution and allows to take into account the DIS features.

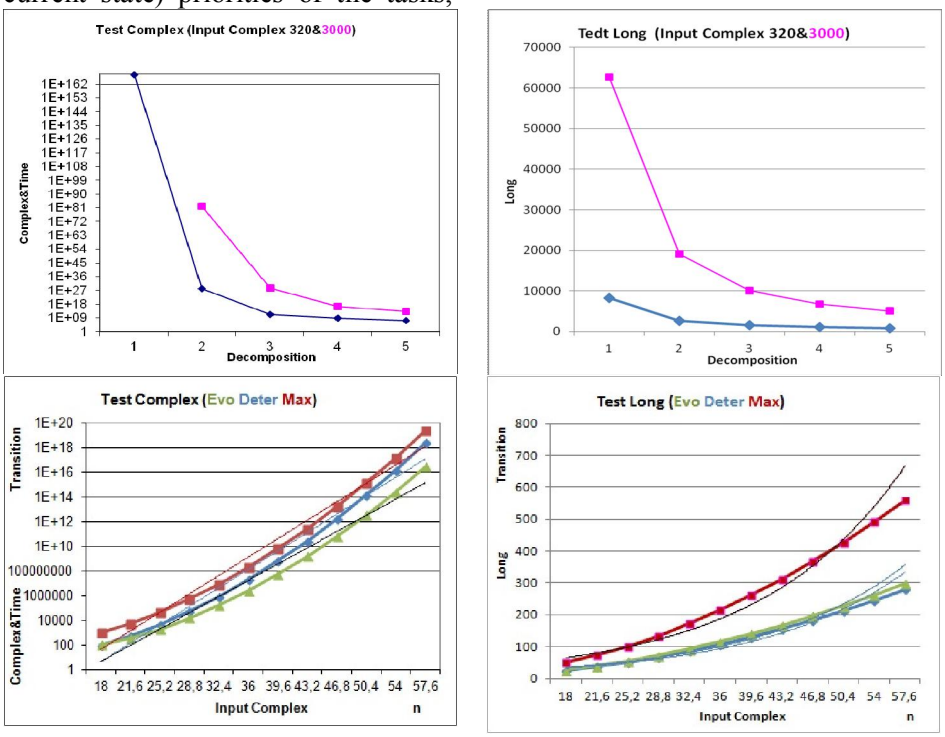


Fig. 3 – The dependence of the complexity and length of the tests on the coefficient of expansion and the complexity of the input components.

6. CONCLUSION

The proposed agent model of the behavioral testing allows to execute the distributed synthesis of behavioral tests on the basis of deterministic and evolutionary models and methods, and also to implement them during the operation and test DIS control taking into account the test conditions of the DIS components.

Combined methods due to the decomposition of testing processes in the distributed MAS allow to reduce the time of synthesis of tests to 90%, increase the adequacy of representation of the structure and interactions of real DIS in case of allowable computing costs of the cluster level of laboratory network, used in the background.

The exponential complexity, restricting the use of behavioral tests, and dimensionality are polynomially reduced due to the component (DIS) and agent (MAS) decomposition.

The implementation of the MAS conceptual model, executed at the structural and functional and program-algorithmic level in the MS Visual.Net environment, showed the possibility and feasibility of further research and practical work in this direction.

7. REFERENCES

[1] A.S. Tanenbaum, M.V. Steen, *Distributed Systems: Principles and Paradigms*, third ed., Prentice Hall Press, 2013, 705 p.

[2] G. Coulouris, J. Dollimore, T. Kindberg, G. Blair, *Distributed Systems: Concepts and Design*, 5th ed., Boston: Addison-Wesley, 2011, 1067 p.

[3] I. Turchenko, V. Kochan, A. Sachenko, Neural-based recognition of multi-parameter sensor signal described by mathematical model, *International Journal of Computing*, (3) 2 (2004), pp. 140-147.

[4] V.I. Hahanov et al., *Design and Verification of Digital Systems on Chips. Virology & System Virology, Studies Benefits*, Kharkov Nat. Univ. Radioelektronics, Kharkov, Ukraine, A New Word, 2010, 527 p. (in Russian).

[5] H. Tahbildar and B. Kalita, Automated software test data generation: direction of research, *International Journal of Computer Science & Engineering Survey (IJCSES)*, (2) 1 (2011), pp. 99-120.

[6] V. Hrusha, O. Ossolinsky, A. Sachenko, R. Kochan, Distributed on-line temperature measurement and control system, *International Journal of Computing*, (6) 2 (2007), pp. 62-67.

[7] S.J. Russell, P. Norvig, *Artificial Intelligence: a Modern Approach*, Prentice-Hall, Inc. A Simon & Schuster Company Englewood Cliffs, New Jersey, 2010, 1095 p.

[8] Y. Shoham, K. Leyton-Brown, *Multiagent Systems. Algorithmic, Game-Theoretic, and Logical Foundations*, 2010, 532 p.

[9] G. Rojek, R. Cięciwa, K. Cetnarowicz, Algorithm of behavior evaluation in multi-agent system, in *Proceedings of the International Conference on Computational Science ICCS'2005*: 5-th Int. Conference: Atlanta, GA, USA, (May 22-25, 2005) Pt. 3-ed., in: Vaidy S. Sunderam [et al.], *Lecture Notes in Computer Science LNCS*, vol. 3516. Berlin; Heidelberg: Springer-Verlag, 2005, pp. 711-718.

[10] R.M. Hierons, K. Bogdanov, J.P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Luttgen, A.J.H. Simons, S. Vilkomir, M.R. Woodward, H. Zeda, Using formal specifications to support testing, *ACM Comput. Surv.*, (41) (2009), pp. 9:1-9:76.

[11] M.J. Rutherford, *Adequate System-Level Testing of Distributed Systems*, A thesis submitted to the Faculty of the Graduate School of the University of Colorado, 2006, 158 p.

[12] K. Sinha, *Structural Complexity and its Implications for Design of Cyber-Physical Systems*, Massachusetts Institute of Technology, Engineering Systems Division, January 3, 2014, 342 p.

[13] N.K. Jha and S. Gupta, *Testing of Digital Systems*, Cambridge University Press, 2003, 1018 p.

[14] W.K. Lam, *Hardware Design Verification: Simulation and Formal Method-based Approaches*, Prentice Hall, 2005, 624 p.

[15] V.B. Kudryavtsev, I.S. Grunskii, V.A. Kozlovskii, Analysis and synthesis of abstract automata, *Journal of Mathematical Sciences*, (169) 4 (2010), pp. 481-532.

[16] I. Grunsky, O. Kurganskyy, I. Potapov, Languages representable by Vertex-labeled graphs, in *Proceedings of the 30-th International Symposium on Mathematical Foundations of Computer Science*, (2005), vol. 3618, pp. 435-446.

[17] A.A. Shchurov, R. Mařík, A formal approach to distributed system tests design, *International Journal of Computer and Information Technology*, (3) 4 (2014), pp. 696-705.

[18] R.M. Hierons, H. Ural, The effect of the distributed test architecture on the power of testing, *The Computer Journal*, (51) 4 (2008), pp. 498-510.

[19] O. Baldellon, J.-C. Fabre, M. Roy, Minotor: monitoring timing and behavioral properties for dependable distributed systems, in *Proceedings of the 19-th IEEE Pacific Rim International Symposium on Dependable Computing*

PRDC'2013, Vancouver, Canada, (December 2013), 10 p.

[20] G. Jervan, R. Ubar, Z. Peng, P. Eles, Test Generation: A Hierarchical Approach, *Chapter in System-level Test and Validation of Hardware/Software Systems, Springer Series in Advanced Microelectronics*, (17) (2005), pp. 67-81.

[21] A.M. Khamis, M.R. Girgis, A.S. Ghiduk, Automatic software test data generation for spanning sets coverage using genetic algorithms, *Computing and Informatics*, (26) (2007), pp. 383-401.

[22] J. Hudec, E. Gramatová, An efficient functional test generation method for processors using genetic algorithms, *Journal of Electrical Engineering*, (66) 4 (2015), pp. 185-193.

[23] P. Bernardi, E. Sanchez, M. Schillaci, G. Squillero, M. Sonza Reorda, An evolutionary methodology to enhance processor software-based diagnosis, in *Proceedings of the IEEE Congress on Evolutionary Computation*, Vancouver BC, (July 16-21, 2006), pp. 859-864.

[24] M. Harman, P. McMinn, a theoretical & empirical analysis of evolutionary testing and hill climbing for structural test data generation, in *Proceedings of the International Symposium on Software Testing and Analysis*, London, (9-12 July 2007), pp. 73-83.

[25] S.K. Singh, S. Sabharwal and J.P. Gupta, A novel approach for deriving test scenarios and test cases from events, *Journal of Information Processing Systems*, (8) 2 (2012), pp. 213-240.

[26] Z. Houhamdi, B. Athamena, Structured integration test suite generation process for multi-agent system, *Journal of Computer Science*, (7) 5 (2011), pp. 690-697.

[27] C. Nguyen, A. Perini, P. Tonella, Goal-oriented testing for MASs, *Int. J. Agent-Oriented Software Eng.*, (4) (2010), pp. 79-109.

[28] S. Yu, J. Ai, Software test data generation based on multi-agent, *International Journal of Software Engineering and its Applications*, (4) 1 (2010), pp. 67-74.

[29] A.S. Sugak, A.N. Martynyuk, Evolutionary network model of testing for distributed information systems, Elecrotehnical and Computer Systems, Odessa, Ukraine, 16(92) (2014), pp. 71-77. (in Russian).

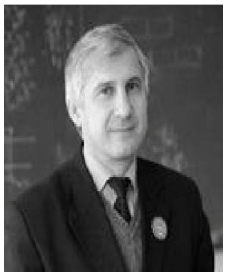
[30] A.S. Sugak, A.N. Martynyuk, Multi-agent systems of behavioral diagnosis for distributed information system, Elecrotehnical and Computer Systems, Odessa, Ukraine, 19(95) (2015), pp. 187-194. (in Russian).

***Anna Sugak**, graduated Odessa National Polytechnic University, Information systems Department. Now she works as assistant of the Department of Computerized Control Systems at Odessa National Polytechnic University.*

Research interests: monitoring and diagnostics of computing devices, synthesis test for discrete devices.

***Ph.D., Oleksandr Martynyuk**, graduated Odessa Polytechnic Institute, Electronic Computing Devices Department. Now he works as associate professor of Department of Computer's Intellectual Systems and Networks at Odessa National Polytechnic University.*

Research interests: protocol analysis and verification of computer networks, test synthesis for computer systems.



***Professor Oleksandr Drozd**, graduated Odessa Polytechnic Institute, Electronic Computing Devices Department. Now he works as Professor of Department of Computer's Intellectual Systems and Networks at Odessa National Polytechnic University.*

Research interests: functional (working) diagnosis of computing devices, accuracy of approximate calculations, dependability of critical system and application.