Towards monitoring systems development on the basis of the multilevel relative finite state operational automata

Michael Chervontsev[1,2][0000−0003−3897−5068], Saddam Abbas[1][0000−0001−9931−463X], Alexander Vodyaho[1][0000−0002−1933−0933], and Natalia Zhukova[3,4][0000−0001−5877−4461]⋆

[1] Saint Petersburg Electrotechnical University "LETI" (ETU) St. Petersburg, 197376, Russia aivodyaho@mail.ru

[2] Research and Engineering Center of Saint Petersburg Electro-Technical University, Saint-Petersburg, 197022, Russia chervontsev.mikhail@nicetu.spb.ru

[3] Saint-Petersburg Institute for Information and Automation RAS St. Petersburg, 199178, Russia nazhukova@mail.ru

[4] Saint Petersburg National Research University of Information Technologies, Mechanics and Optics ITMO University, St. Petersburg, 197101, Russia.

Abstract. In the paper the problem of monitoring of a complex technical systems state of is considered. It is noted that such systems include a large number of elements and their structures permanently change. Classification of problems of monitoring is given and possible approaches to their decision are considered. For solving these problems it is suggested to use model driven approach. As model of structure of a target system it is offered to use multilevel relative finite state operational automata and as behavior model it is offered to use the work flow graphs. The actuality of models is provided by means of analysis of log files. Authors have suggested an algorithm of multilevel relatively finite state operational automata synthesis earlier. For maintenance in actual state of behavior models one can use process mining algorithms. The example of usage of suggested approach is given.

Keywords: monitoring systems · automata based approach to monitoring systems development

1 Introduction

The modern stage of technology development is characterized by the permanent increasing of the complexity of technical systems and by increasing of the number of subsystems, number of hierarchy levels and by the complexity of links between the elements of the system and by the complexity of the systems behavior. The complexity of the systems behavior can be expressed in the expansion of the

set of implemented functions, in the complexity of realized functions and in the ability of systems to adaptation. Advanced systems realize elements of cognitive behavior, including ability to learning and self learning. The problem of monitoring the status of such systems becomes a complex scientific and technical task. The monitoring system (MS) has to give information about the current state of the target system (TS), consisting of a huge number of elements, the structure of which is permanently changing, with strict restrictions on the bandwidth of communication channels and the number of observation points. Obviously, modern MS must have a level of intelligence, at least no lower than the level of intelligence of TS, i.e. should be adaptive and preferably have cognitive abilities.

2 Known approaches for monitoring problems solving

The problem of monitoring of the objects states was faced by many generations of researchers and developers, in particular technical systems developers. In the early stages, this problem was investigated by specialists in the of control systems, in particular adaptive control systems [1], where it was considered as one of the subtasks of control theory. In later stages, this problem was faced by business process management system developers [2, 3], where it was no longer tightly linked to the problem of control systems development.

Nowadays there is currently no general theory of MS building. These problems are discussed using different architecture viewpoints and different perspectives for different classes of information systems (IS), although it is becoming increasingly clear that the MS represents a separate class of IS that can be considered as a subclass of information management systems.

Development of modern MS can be based on the following main IT technologies: 1) theoretical researches in the field of adaptive control systems [1]; 2) structural system dynamics studies [4]; 3) pragmatic approach to building monitoring systems on the base of ITIL/ITSM approach [5], 4) BPM systems [2]; 5) data fusion system [6]. For the developers of MS, the experience gained in the construction of MS for non-technical objects, in particular natural and biological objects, is of considerable interest.

From a theoretical point of view, the results obtained from studies in the field of adaptive management systems construction, in particular [1], as well as studies in the field of structural dynamics of systems [4] are the most useful for the construction of the CM. In addition, the works of the authors [7, 8] on the synthesis of multilevel automata models and the works [9] on the synthesis of multilevel business processes (BP) from the logs can be noted. The pragmatic ITIL/ITSM approach to building MS is very good for enterprise information systems but attempts to apply it to other classes of IS, for example, for the Internet of Things (IoT), as a rule, are not effective. The specifics of building MS for the IoT are discussed [10]. A number of technologies can be useful for the modern MS building: 1) data processing in sensor networks [11]; 2) algorithms of process mining [12]; 3) multivevel automata synthesis [7,8]; 4) work related to the use of software Agile architectures [13].

It should be noted that the above mentioned approaches only address separate aspects of MS development and do not form a holistic approach to development of MS that can be used for monitoring complex technical systems (CTS) with dynamic structure and adaptive behavior.

3 Suggested approach

The suggested approach is aimed primarily at solving the problems of effective management of CTS with dynamic structure and behavior through the implementation of effective monitoring procedures. This approach can be applied both to the implementation of monitoring procedures of some external TS, and to the implementation of self monitoring procedure of MS. The key ideas of the developed approach is are the follows.

1. A TS is regarded as a partially observable system whose state information is not fully available for a certain moment of time.

2. Available TS state information is stored in the form of models of the TS that includes 3 model groups: models that describe structural dynamics, models that describe TS behavior, and transformation models. The models actuality is supported by means of log files processing.

3. The models are essentially architectural models. Thus, both TS and MS can be considered as agile architecture systems.

4. The proposed approach can be positioned as a multimodel approach in the sense of [4], based on the use of dynamic models, and as a agile architectural approach, based on the use of many architectural points of view and architectural perspectives [13].

The 3 main features of the proposed approach distinguish it from the existing approaches: 1) the use of explicit models describing structural dynamics and business processes in the TS; 2) structural dynamics and business processes are considered as a single entity; 3) the use of an architectural approach to the development of the MS, which allows to use suggested approach for solving the task of monitoring the state of multilevel systems with dynamic structure and adaptive behavior, which include many thousands of elements.

4 MS generalized model

In general, the MS consists of a TS and a monitoring subsystem. The monitoring subsystem receives reports about events occurring in the TS. The monitoring subsystem may provide requests for the state of the TS or its elements. The monitoring subsystem works with models that are generated from status messages (logs) and provides information to users according to their roles and requests. Systems of different nature can be conceded as TS, but the subject of this work is CTS. Within the framework of the developed approach, the structure and behavior of the TS are subject to the following main restrictions: 1) from the point of view of the monitoring subsystem, the TS is passive, i.e. it does not react in

any way to attempts to pick up logs, it cannot prevent the attempts of pick up logs, it does not change its state and behavior because of log pickupping; 2) log sources are linked to certain subsystems and are not connected with each other in any way; 3) all logs contain structured data which are represented in XES (eXtensible Event Stream) format or can be converted to XES format [14]; 4) the TS is not a hard real time system; 5) all control actions are formed on the basis of the TS model; 6) all logs must be converted to XES format as part of preprocessing; 7) all noises in logs shall be cleaned at the preprocessing stage.

Thus, the following types of tasks are not solved within the framework of the suggested approach: 1) in the case when logs from the TS encapsulate unstructured data (texts in natural languages, voice messages); 2) in the case when the TS is an active component, that is, it may prevent the process of log pick up by suppression or distortion of logs, or when the TS changes its structure and/or behavior when it detects an attempt to pick up the logs; 3) if the message semantic is unknown, i.e. when the problem is to understand the languages which is used for communication between TS elements (humans, animals, robots, etc.).

The MS can be considered as a log processing system. On the basis of logs and context information, TS models are built, on the basis of which information is presented to users according to their role and their requests. At the conceptual level MS can be defined as $S_m = \{X, Y, M, F\}$, where X is a set of queries about TS state, Y is a set of reactions to queries, M is a set of models, F is a set of functions that implement mappings $X \xrightarrow{F} Y$, or $X \xrightarrow{F} M \xrightarrow{F} Y$. The generalized structure of the MS which includes 3 automata is shown in Figure 1.

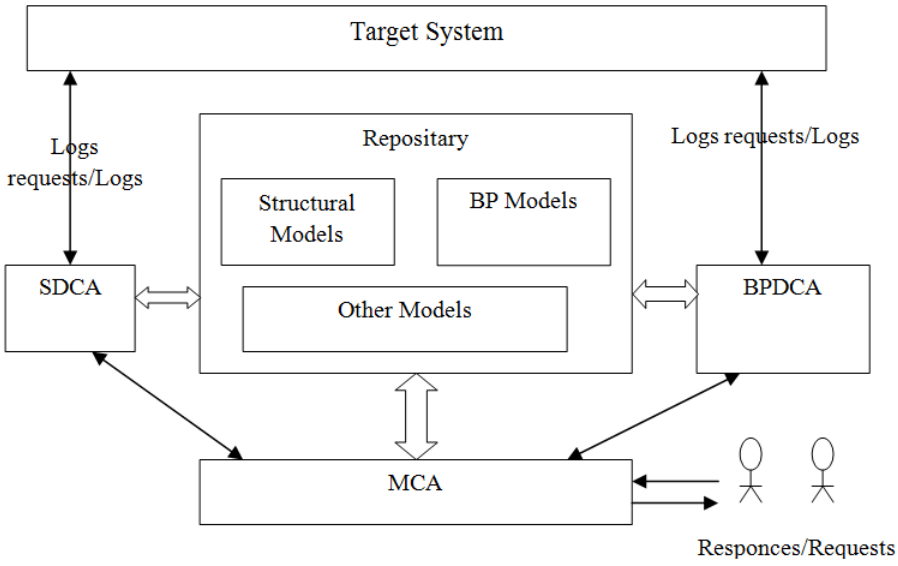

Fig. 1. The generalized structure of a MS.

MS includes 4 main elements: MS = $\{R, SDCA, BPDCA, MCA\}$, where: R - repository, SDCA - structural dynamics control automate, BPDCA - BP dynamics control automate, MCA- monitoring control automate. The repository is designed to store models, logs, scripts, contextual and other information.

Generalized algorithm of MS realized by MCA:

Step 1. Start background monitoring processes (polling)

Step 2. Waiting requests from other automata or a user

Step 3. If there is a request, from SDCA, then process the request and go to step 2

Step 4. If there is a request from BPDCA, then processing and go to step 2

Step 5. If there is a request from a user, then processing and go to step 2.

Generalized algorithm of SDCA operation:

Step 1. Waiting for log from TS hardware or software platform or request from MCA

Step 2. If the log from the TS platform is received, then the log cleaning, processing and go to step 1

Step 3. If a request for MCA is received, processing and go to step 1

Generalized algorithm of BPDCA operation:

Step 1. Waiting for log from BP or request from MCA

Step 2. If the log is received, then the log cleaning, processing and transition of item 1.

Step 3. If a request for APC is received, processing and transition of item 1.

In general case MS can be a part of monitoring and management system. The generalized structure of such a system is shown in Figure 2).

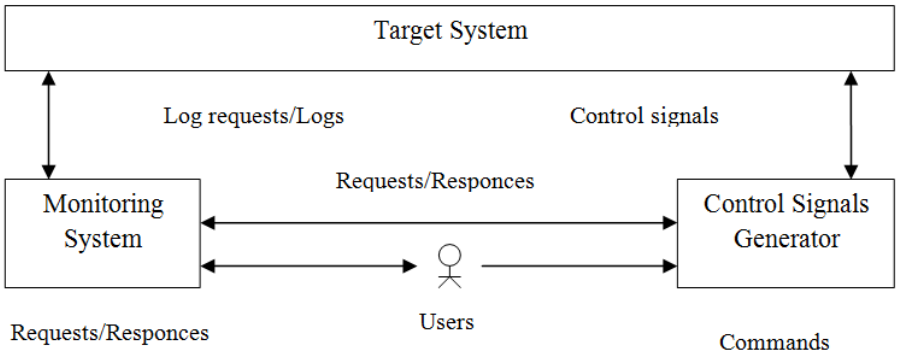

Fig. 2. MS as a part of Management System.

System according to Figure 1 includes 4 main elements: TS, MS, control signals generation subsystem and users. Monitoring requests can be generated either by a user (manual control) or by the control signals generator (automatic control or mixed mode). MS receives requests from users or from a control signals

generator about TS or its elements status in terms of a domain specific language. MS transforms requests into log requests script and on the basis models and information from logs generates answers to the requestor.

One can define following partial case of the system as shown in Figure 2: 1) the system operates in automatic mode (user is absent); 2) the user can only observe the behavior of the CA; 3) the user controls the system manually.

The MS can also be seen as a stand-alone management system that handles monitoring requests for some external system.

5 Requirements to models

The model requirements that to the operation of automata can be formulated by the following way.

General model system requirements. The general requirements for models to be used in CTS MS can be defined as follows: 1) models must have ability to describe CTS with dynamic structure and adaptive behavior; 2) models must be functionally complete, i.e. they must be able to answer questions from all users within the scope of the tasks; 3) models must be executable, i.e. it must be able to use them in run time and preferably in non-rigid real time mode; 4) models must have implementations of acceptable complexity, even for TS consisting of a large number of elements.

Requirements to structural models. The main requirement for structural models is the ability to describe with their help the dynamic structure of TS in terms of elements, connections and their current, past and future states of structural elements of TS. They must give the possibility to describe the constantly changing multilevel structure of arbitrary complexity, and it must be possible to build them automatically from logs.

Requirements for behavior models. The main requirement for behavior models is the ability to describe with their help BP and determine BP state for current moment of time, for past and future moments of time. They must allow observe BP execution in run time using log information. They also must allow change BP structure when TS structure changes.

Transformation model requirements. These models are used to form representations and they can be conceded as brokers between the model system and users. The main function of this subsystem is to translate queries formulated by interested parties into the language of queries to the corresponding models and, accordingly, to make the inverse transformation.

6 Automata models usage in MS

The models used in the frames of model based approach to MS construction are essentially architectural models. Different formalisms such as UML diagrams, Petri networks, graphs, particularly workflow graphs, ontologies, state machines can be used as TS models [12, 13]. Taking into account the requirements to

structural models, the most promising is the use of multilevel relative finite state operational automata as the models.

The main argument in favor of this approach is the possibility of automatic synthesis of such automata by logs [6, 7]. Structural models describe TS in terms of subsystems, links between them, in terms of inputs and outputs, and properties (attributes).

In order to describe dynamic behavior it is necessary to able to describe multilevel managed adaptive processes. One of the key model requirements is the ability to generate automatically process models from logs.

Automata models also can be used for solving this problem. Thus, a hierarchical stochastic automate with a reconfigurable structure can be used to describe the monitoring process in its most general form. In fact, it is a set of automata that can be used for simulation.

One can define the following main types of TS: 1) simple TS with fixed structure; 2) complex TS with fixed structure; 3) simple TS with variable structure; 4) complex TS with variable structure; 5) simple cognitive TS; 6) complex cognitive TS. Table 1 shows the types of automata that can be used to model separate TS classes.

The simplest Mealy machine is defined as $A = (X, Y, S, T, S_0)$, where X and Y are finite sets of input and output signals, S is not more than countable set, S_0 is initial state), T is table of transitions. Such machines are studied in detail, algorithms of their synthesis are known, but with their help it is possible to describe directly only the simplest MS with a fixed structure of small complexity. A hierarchical (multilevel) automate is defined as an oriented graph H = {V_0, V_k, L, T} having an initial vertex V_0, nested vertices V_k, and a plurality of arcs L.

Table 1. Types of automata that can be used in MS.

	TS structure	Automata type	Necessity of automatic automata synthesis
1	Simple TS with fixed structure	Elementary finite state automate	No
2	Complex TS with fixed structure	Multilevel finite state automate	No
3	Simple TS with variable structure	Automate with variable structure	Yes
4	Complex TS with variable structure	Multilevel automate with variable structure	Yes
5	Simple cognitive TS	Stochastic automate with variable structure	Yes
6	Complex cognitive TS	Multilevel stochastic automate with variable structure	Yes

Each can be mapped to either automate A or graph H, and the vertex V_0 only to graph H. Hierarchical MS are typically organized as a tree. When the state of the system changes, very often only the separate low level subsystems change their state. Hierarchical automata can in principle be reduced to conventional Mealy machine. The use of hierarchical automata in some cases makes it possible to simplify the procedure of their synthesis. They can also be useful when working with distributed MSs. If the structure of the TS is not constant, then the automate has to be rebuilt. This is not possible in all cases. *Variable*

Table 2. Types of automata with variable structure

	Inputs	**Outputs**	**States**	**Transitions**	**Comments**
0	0	0	0	0	Classical automate with fixed structure
1	0	0	0	1	Transition table changes
2	0	0	1	0	-
3	0	0	1	1	Individual states can appear and disappear and the transition table can be changed
4	0	1	0	0	Only the output function changes, the logic of the automate operation remains unchanged
5	0	1	0	1	The function of the outputs and the logic of the operation of the automaton, which is described in terms of the same states, change
6	0	1	1	0	-
7	0	1	1	1	Changes the function of outputs and the logic of the automate operation, which is described in terms of the same states
8	1	0	0	0	Only the output table changes, the logic remains the same
9	1	0	0	1	The logic that is described in terms of old states and the output function changes
10	1	0	1	0	-
11	1	0	1	1	Automate inputs and operation logic, which is described in terms of new states change
12	1	1	0	0	-
13	1	1	0	1	The logic of the operation, which is described in terms of previous states changes
14	1	1	1	0	-
15	1	1	1	1	The automate is completely replaced with a new automate

structure automate can be defined as an automate whose transition and output functions are explicitly time dependent, $A_{i+1} = \mathrm{F}(A_i, \mathrm{t})$ or as $\mathrm{A} = (\mathrm{X(t)}, \mathrm{Y(t)}, \mathrm{S(t)}, \mathrm{T(t)}, s_0)$.

In the present work, it is assumed that the MS operates with discrete states and works in discrete time. As TS operates in discrete time and works with discrete states, it can be considered that at each time t_i the automate is in some particular state s_i and then it can be defined as A = (X(s_i), Y(s_i), S(s_i), T(s_i), s_0). If the variable structure automate works for a very long time then the number of states in general becomes infinite. It makes the procedure of synthesis of such automate very difficult. This problem can be solved using 2 approaches: by generating a new automate "on the fly" by means of correcting automate structure every time when a new log arrives or by reducing this problem to the special case when the number of states is a countable set. Thus, it would be useful to consider special cases when a new automate is not change its state to a new state, but only the individual elements (attributes) of the description are changed. In this case, the variable-structured automate may be defined as a set of automata, Ai = Ai-1, Δi-1, i, where A is a set of operations, realization of which can transform the automat to the desired automate.

Variable structure automata can be considered as a class of automata for which different elements can be changed and other elements remain unchanged. The main types of automata with variable structure are presented in Table 2, where 0 corresponds to unchanged elements and 1 corresponds to variable elements. Some combinations have no practical implementations. This applies primarily to cases where states appear and disappear, but the transition table does not change. The situation with the appearance of new input signals without changing the transition table is essentially the appearance of unidentified automata and signals. The reasons may be different. First, such a signal may occur due to a time delay or failure in the log generation subsystem on the MS side, second, it may be a really new signal for which the automate needs to be reconfigured. If the signal disappears, some of the automate states may become unavailable. These cases require separate consideration. Other combinations have real implementations. Changing input signals is a simple operation that can be implemented in terms of operations **"Add Signal"** and **"Remove Signal"**. Behavior changes (transition tables) can be implemented in terms of operations **"Delete Transition"** and **"Add Transition"**. Thus, the automate reconfiguration can be implemented in the form of a scripts including the operations listed above.

The task of synthesis of multilevel automata with variable structure is discussed in detail in the works of the authors [7, 8].

A stochastic automate (Moore) can be defined as A = (X, Y, S, M(x), s_0), where X is the finite set of input signals, Y is the finite set of output signals, S is the counting set of states, s_0 is the initial state or probability distribution of initial states, M(x) = $||m_{ij}(x)||$, q is a function-function distribution.

It is known fact, that by means of increasing automate complexity it is possible to build an equivalent deterministic automate. In this case, the inputs are described by the stochastic matrix family X(t) = $||x_t(t)||$.

From the point of view of MS construction probabilistic machines are interesting, first of all, from the point of view of organization of the training process, by means of probability adjustment.

7 Structural Dynamics Models

The structural dynamics control subsystem at the logical level can be represented by 2 main subsystems: structural dynamics model (SDM) and SDM control automate (SDMCA) (Figure 3). The physical implementations of the system according to Figure 3 can be very different. They can be realized as a distributed system, or as distributed ontologies, SDM and SDMCA can be implemented in the frame of one system or as separate unites.

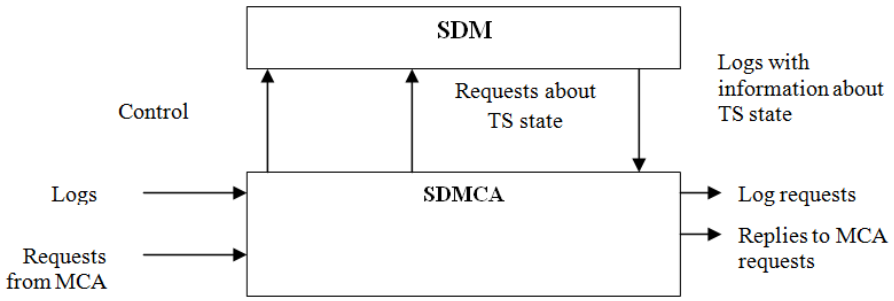

Fig. 3. Logical structure of structural of dynamics management subsystem.

SDMCA receives information about state of individual TS elements changes, which comes in the form of logs. On the basis of received information model corrections are realized. If it is necessary, then SDMCA sends request for additional information. It can be either simple queries or scripts. The SDMCA may receive requests on the state of the TS elements from the MCA and generate responses to these requests.

The IDA can be defined as:

$$M_{sd} = (G_{sd}, F_{sd}, Q_{sd}),$$

where G_{sd}, is a set of sets of graphs, F_{sd} is a set of graph conversion operators, Q_{sd} is the language of TS state requests. The G_{sd} graph is a multilevel graph and describes the TS structure at some hierarchy level in terms of the subsystems and the relationships between them:

$$G^i_{sd} = (V_i, L_i),$$

where V_i is the set of vertices belonging to the i-th level and Li is the set of links belonging to the i-th level. Through V_{ij} we will denote the j-th element belonging to the i-th level, L_{ij} - j-th link belonging to the i-th level, L_{ij}. The set of sets G_{sd} can be divided into 3 subsets: 1) a subset of fully serviceable G_{sds} configurations; 2) a subset of partially serviceable G_{sdp} configurations; 3) a subset of defective G_{sdr} configurations.

Each subset of G_{sd} includes a countable set of configurations. Each element of the static model is vertex V_i and the link L_{ij} can have an arbitrary number of attributes. The number and types of attributes depend on the specifics of the subsystem to which the vertex or link is mapped.

Attribute values can be read at an arbitrary point in time in a log format, defined indirectly or with the help of logical inference. The attributes contain parameters characterizing the technical state in terms of "green-yellow-red, GYR" [4], where "green" means that the element is fully serviceable, "yellow" means that the element is partially serviceable, "red" means that the element is defective. The parameter may also be unavailable. In this case, its value can be determined through a logical inference. An arbitrary number of log available points can be associated with each element. Reference and working (current) models are distinguished. These models can have different attributes.

In particular, the working model describes the TS in concrete moment of time. The reference model can be linked with the interval and the parameters can be colored in the GYR colors. Elements (subsystems) can appear and disappear at an arbitrary moment in time because of different reasons i.e. on/off, break-fix, user login/exit.

Models of new elements may be known and stored in the repository, but may not be known. Relationships between elements can also have attributes.

More formally, a model describing structural dynamics represented in the form of multilevel relative finite state operational automate can be defined as follows.

Automate = {Set of Input Messages, Set of Output Messages and Logs, Set of Sets of Automate State, Set of Transition Tables}, Machine State = {General State, Current TS Structure}, General State = {Serviceable, Faulty, Partially Serviceable}, Current structure = {Elements, Links}, Structure element = {Attributes, Links}, Attribute = {Set of Name-Value pairs}, Links = {Siblings, Parents, Children}.

The following information comes to automate inputs: 1) requests TS state (current state requests, past state requests, future state requests); 2) requests for reconfiguration (requests for reconfiguration in order to improve performance, requests for reconfiguration in case of TS element occurrence or disappearance.

Logs may by generated in different ways: 1) logs generated by built-in monitoring and diagnostics elements of TS subsystems; 2) logs generated upon request from SDCA; 3) logs generated by diagnostic scripts launched by SDCA.

SDCA is a processor (engine) providing access to the model by performing a set of operations. SDCA is responsible for monitoring the TS state and control TS structural dynamics. The SDCA input receives logs, which can be messages

(responses to requests) or signals. The SDCA may provide requests for the state of the TS elements and provide control signals for reconfiguration or parameter adjustment. All end-user requests for TS element status pass through MCA. The SDCA controls the current model using commands of restructuring the current model. The main commands of model management are following: 1) replace the model; 2) to remove vertex; 3) add vertex; 4) to add link; 5) remove link; 6) remove attribute; 7) add attribute; 8) set attribute value.

SDCA inputs may receive requests for current, past and future state of TS or its elements. The main types of requests are of the following type: in what state is the i-th element of the TS, in what state was the i-th element of the TS in t moment of time, which subsystems are in a partially serviceable state, etc.

The automate operation the can be described as follows. The automate changes its state when new log or signal, which encapsulate information about the change in the state of the TS element, appears. Automate has memory for saving information about all previous States and traces. When a request about the state of the TS or TS elements is made by a user, the automate generates the required representation on the basis of the current state vector and/or historical vectors. The machine can query the status of the TS elements. This action does not change the automate state.

It should be noted that in general, the number of states of the automate can be infinitely large, for example, in the case of IoT systems. In this case the automate is to be built in run time mode. In many particular cases, the number of automate states is finite, for example when we deal with reconfigurable technical systems.

The generalized SDCA algorithm is as follows.

Step 1. Starting the polling process.

Step 2. Waiting for input information.

Step 3. When information about the change in the state of the TS element appears, the following actions are performed:

Step 3.1. Adjust TS model

Step 3.2. Reconfigure TS if necessary

Step 3.3. Send restructuring information to BPDCA and to interested users.

Step 3.4. Adjust and restart the polling process.

Step 4. If a request about TS state arrives then:

Step 4.1. If it is possible, the response is generated on the base on the state vector, and is send to the user.

Step 4.2. If it is impossible then special diagnostic script is generated and launched. The response is formed on the base of the script execution results.

Step 5. If at the input we have a request about the reasons for the incorrect operation of BP, then the diagnostic script is generated and launched, the results of script operation are used to form a response.

8 Behavioral TS models

Dynamic behavioral TS models are models of BP running in the TS. A wide range of models and languages, such as Petri networks, automata models, work flow graphs, L, YAWL etc can be used [2, 12]. Information about BP current state can be obtained by logs, in XES format [14].

Taking into account earlier listed behavior models requirements, a workflow graph model has been selected. Three components of the work flow graph are of interest: the control flow graph, the data (object) flow graph, and the resource graph. The reference model is a work flow graph whose vertices are loaded with information identical to the information present in the logs in the form Name-Values form. If possible, GYR intervals can be set. More formally, the pattern of behavior can be described as follows:

Reference Model = {Vertex Set, Arc Set, Markup Set}.

$< Vertex >::=< Vertex - name >< Parameter - Value > .$

The contents of the **Parameter-Values** field can be specified in 3 ways: one parameter-one value or one parameter - Interval of values, one parameter of 3 interval (Green, Yellow, Red).

BPDCA operates as follows. Logs are input to BPDCA. At the output we have BP control signals, which perform BP adjustment. It is assumed that the reconfiguration is realized inside the TS and the TS control unit selects one of the reconfiguration options from the list of possible ones. The BPDCA may request and receive current state information from the SDCA. The BPDCA informs the SDCA about violations in the BP implementation process and receives information about the TS structure change. In addition, the BPDCA input receives requests about the current status of the BP, to which it responds.

Generalized BPDCA algorithm is as follows.

Step 1. Logs and signals, which are generated by BP, are transmitted to BPCA input

Step 2. Logs are compared to reference logs

Step 2.1. If all values are in the green zone, then continue

Step 2.2. If values are in the red or yellow zone, a message is sent to the SDCA.

Step 2.3. If the TS restructuring signal is received, the possibility of BP restructuring is considered.

If it is possible, then the reconfiguration is performed and a message is sent to the MCA. Markups are implemented using standard methods [2] for workflow methods.

It should be noted that, unfortunately, it is impossible to present the reference model in the form of traces, since for a correctly operating process it is sometimes impossible to determine exactly the order of arrival of logs. This can be done, for example, with the help of data flow models.

9 Representations formation models

The presentations formation models (RFM) describe how users interact with the MS using Domain Specific Languages (DSL). Formally, the RFM can be defined as $M_{pf} = \{TRM, DSL\}$, where TRM is a set of models built on the M_{pf} model, and DSL is a set of languages for interaction with MS, focused on different categories of users $\mathrm{DLS} \xrightarrow{RQ} \mathrm{M}$ and $\mathrm{M} \xrightarrow{RS} \mathrm{DLS}$.

RFM can be realized as a set of services for translation requests and responses into the DSL of different users. As a model describing the operation of the PFM, it is proposed to use a modified data fusion model based on a well-known JDL model [15], which includes 6 Levels.

Level 0. Signal/Feature Assessment. The main task solved at this level is to estimate and predict the values of the individual parameters of TS elements.

Level 1. Entity assessment. The main task solved at this level is to estimate and predict the values of individual parameters and the state of individual entities (objects).

Level 2. Situation Assessment. The main task solved at this level is to assess and predict the state of the structure and parameters of relations between elements of some part of the TS and their impact on the state of the objects themselves.

Level 3. Impact Assessment. The main task solved at this level is assessment and prediction of future states of the TS, its parts and formation of alternative variants of the system response.

Level 4. Process Assessment. The main task at this level is to evaluate and predict the performance characteristics of the TS and compare them with the desired performance indicators.

Level 5. Human-machine interaction (HMI). (Cognition Refinement). Functions related to the improvement of the man machine interface are implemented at this level. This level is responsible for virtualization and collective decision-making.

The JDL model assumes a bidirectional process. Bottom-up movement is a process of information mining from raw data, and then knowledge extraction from information and decision-making. A reverse motion is a stream of queries that are formulated by interested persons in terms of the subject area. Note that functions related to different levels can be implemented in an arbitrary order [6].

In fact, JDL model defines what is "done" rather than "how is done" i.e. the JDL model is a functional model. It was never seen as a process model or as a technical architecture model. The classic JDL model is a fairly general model that is not directly related to any of the subject domains.

Visual data-fusion model. This model is a further development of JDL model. The authors solved the following tasks: i) minimization the amount of information provided to the user saving the needed level of quality; ii) provide information strictly in accordance with the user request and role; iii) provide information taking into account the current situation. The main distinguishing feature of this model from classical JDL model is that suggested MJDL model is domain oriented model and because of it this model can be populated by concrete services

and stakeholders. So suggested MJDL model may be conceded as functional-process model.

The generalized structure of the proposed MJDL is shown in Figure 4

The proposed MJDL model as well as the classic JDL model has 6 levels. The main users (stakeholders) include 4 groups: service engineers (responsible for the technical state of the TS infrastructure), operator (monitoring the state of separate TS subsystems, if necessary, they can realize control actions), managers responsible for the technical state of large subsystems or the TS), business analysts and top managers who are primarily interested in the overall state of the TS. These user groups work at different levels of the MJDL model. Separate levels of MJDL model realize following functionality.

Level 0. The main task solved at this level is generation requests for logs and processing "raw" logs, processing, evaluation and prediction of values of TS elements individual parameters. It can be said that individual logs are processed at this level. Typical problems solved at this level are the elimination of noise such as random logs, loss of logs, inability to receive the required log, etc.

Level 1. The task of this level is characteristics evaluation of individual elements of TS and prediction the values of individual parameters and the state of individual TS subsystems. At this level functions related to the generation of information about individual objects are implemented. It may be, for example, information about the technical state of individual subsystems of a CTS, the location of the object, the speed of movement and the direction of movement.

Level 2. Evaluation of the overall TS state. The main task solved at this level is to estimate and predict the TS state. This layer implements functions related to generating situation information in a particular context in terms of entities (subsystems), relationships between them and events.

Level 3. Reaction definition. This level is present if we speak about a monitoring and management system. The main task solved at this level is evaluation and prediction of future states of the TS and its parts in terms of utility/cost, generation of alternative versions of control signals generation. At this level, the situation is assessed, which may include an assessment of the dynamics of the situation, assumptions about possible actions of objects external to the CA, threats and their own vulnerabilities.

Level 4. Efficiency assessment. The main challenge at this level is to evaluate and predict the performance characteristics of both the TS and the MS itself and compare them with the desired performance indicators. At this level, the CM itself is monitored, in particular to improve its timing parameters.

Level 5. Human-machine interaction. Functions related to the implementation of HMI procedures are implemented at this level. In addition, this level is responsible for virtualization and collective decision making. Also knowledge management mechanisms are implemented at this level, i.e. such questions are solved: who requests information, who has access to information, for what purpose the information will be used, etc.

The use of the proposed MJDL model allows solve 2 main problems: creation of common terminology, which specialists in different fields can use in the de-

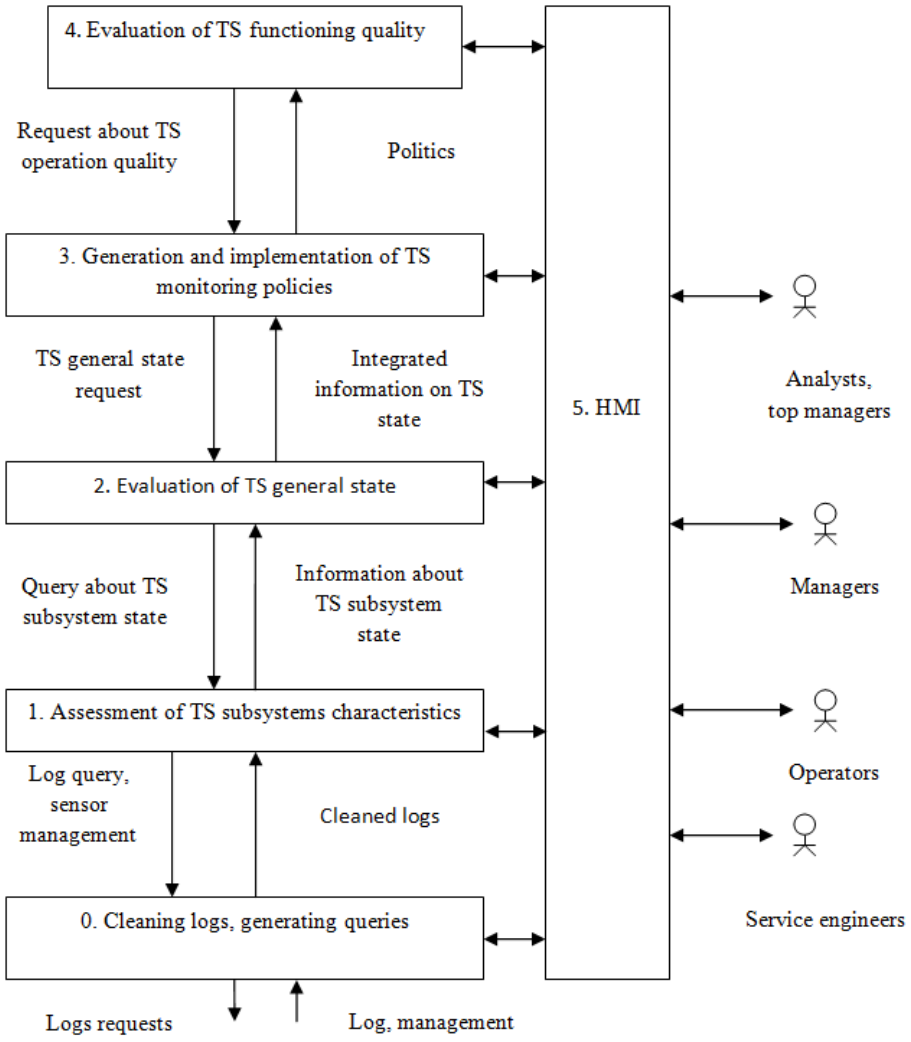

Fig. 4. MJDL model.

sign of different MS for different subject domains. In this context, the task of harmonizing inter-layer interfaces, that is, standardizing the way information is presented, comes to the fore.

10 Typical tasks of monitoring. Variants of problem statement

Depending on what is known about the TS and what needs to be known, 4 global tasks for MS can be identified.

Task A. Verifying the relevance of the model using log file information. Corresponds to the case where it is known both the static model of the TS and its dynamic model, i.e. models relating to all levels of interest are known.

Task B. Build a behavior model using a structural model and logs. In this case, the TS structure is known, but there is no knowledge about BP in TS. This problem is similar to the classic process mining task [12] when we want to restore the BP structure either from scratch from logs or from a known higher level dynamic model and logs.

Task C. Building a structural model by behavior model and logs. BPs running in the TS are known or can be restored by logs. For example, in technical systems, this may be an option where it is necessary to determine the cause of the malfunction from an anomaly in behavior.

Task D. Construction of structural and behavior models using descent approach. In this case there is no complete information on both system structure and BP. There is only a set of logs.

11 Generalized algorithms of typical monitoring problems solving

Tasks Type A (Model conformance checking). The very fact that the model is irrelevant can be detected either through the appearance of messages (logs) or when internal control circuits are activated. It is assumed that the proper polling procedures are started. For Type A of problem statement there are a number of special cases

Task A1. Verify that the structure model and behavior model are correct. The indication of incorrect models can be: a message from SDCA generated by build in internal control schemes or a reaction to a request to rebuild the BP, a message from BPDCA about appearance of a yellow or red log, The reaction of MS can be a request to SDCA for running monitoring script in order to identify the origin of incorrect operation.

Task A2. Correction of models. The structure model is adjusted when it becomes known about changing TS element state. It can be done on the base of information from the log or by means of launching diagnostic script. Correction of the behavior is performed in the following cases: when a log with information about the restructuring of the TS, when a yellow or red log appears or when

a request from the user about restructuring comes, for example, in order to improve the efficiency of the TS.

Task A3. Specifies the point of divergence between reference and real models. This task usually occurs in a MS running in post mortem mode. A typical approach to solving this problem is to view logs from the BP, i.e. from the BPDCA. We can start viewing logs both from the beginning or from the end. One can track the control flow graph, resource graph, and object flow graph for yellow-red logs. It is possible to track logs coming from SDCA for appearance of yellow-red logs. This approach usually allow detect the time and cause of the error.

Task A4. Construction of a picture of the development of an emergency situation. This task usually occurs in post mortem systems. The model of emergency situation development can be represented as a pair of ordered and time-aligned chains of yellow-red logs from the SDCA and BPDCA Using this information one can build causal links between events.

Task A5. Identify risks of emergency situations and present situation development options. Alarm risks are defined in terms of yellow logs both from SDCA and BPDCA. The forecast is based on the assumption that yellow logs can turn into red log. So one can define the structural element responsible for the appearance of the yellow log, if the emergency situation is detected through the log from the BPDCA.

Tasks of type B. Building a behavior model from structure model and logs. This is a well known process mining task [12].

Task B1. Build behavior model on the basis of structure model and logs. Control flow graph recovery is a process mining task in classical statement. (Restoring a resource graph and an object flow graph are separate tasks.)

Task B2. Restore the structure of individual BP from logs. This is a special case of classical process mining task when it is necessary to build BP model using logs only from certain BP.

Task B3. It is a task of finding certain patterns of unwanted behavior. The solution of this problem assumes usage of a pattern library. This problem as a rule is a security problem.

Tasks of type C. Building a structural model by behavior and logs. Since the structural model is a multilevel relative finite state operational automate, this task is the essence of the task of synthesis of such automate from logs [7,8].

Task C1. Receiving actual information about changing of TS structure.

The main idea of solving this problem is running polling processes and process logs in run time. Also it is necessary to process in run time messages from the TS build-in control systems which are responsible for TS reconfiguration. In some cases it would be enough only to correct models but in very often it is necessary to rebuild the model from the stretch.

Task C2. Building a structural model from a behavior model. This is the task of synthesis multilevel relative finite state operational automate from logs. This task can also be defined as the task of determining the structure of a black box

in terms of finite state operational automate on the basis of known behavior in a minimum number of steps. Algorithms of synthesis of such automata are described in details in publications [6, 7].

Task C3. Identify a faulty TS element within the minimum time. This task assumes that some TS subsystem is known to be malfunctioning, but the specific element causing the fault is unknown. It is necessary to generate or select the diagnostic procedure which can find the faulty element.

*Task C4. Defining a TS element which is a "bottleneck".*The term "bottleneck" can be defined in different ways. If bottleneck is an unreliable element, it is the task 5. If a bottleneck is defined in terms of performance or throughput, this task is solved by receiving proper logs from running benchmarks. These can be both logs from structural elements and logs received from BP.

Tasks type D. Building of structural and behavioral models using descent approach. This fairly common task can be solved in different ways depending on which logs are available. If logs relating to all levels are available, the multilevel relative finite state operational automate can be build (Task C2) and then a behavior model can be build. If logs related only to certain levels are available, one can use the "step by step descent" method, that is, one can build only of the level of detail of the available logs.

The generalized algorithm is as follows.

Step 1. Define a behavior model at the top i_0 level.

Step 2. Using the black box definition structure procedure (Task C2), define the top i_0 layer structure.

Step 3. For each i_0 level model element, using generated test scripts build lower level models in the same way.

Thus, the algorithm is reduced to the sequential application of the black box structure determination procedure and the construction of BP by logs. The case where logs themselves are available, but their structure and semantic are unknown, is a separate task, which is not considered in this work. This task type can be defined as a separate type (Type E). A type E task can be formulated as a task of defining a log structure by a structural model and a behavior model.

12 Example of suggested approach usage

The suggested approach was used for development of a number of real IS. One of such systems is a cable television network management system. Modern cable television networks have hundreds of thousands and even millions of subscribers, and the number continues to grow. The system itself is a distributed system that includes data networks, server and subscriber equipment. The typical structure of a cable digital television (CDT) network is shown in Figure 5

In Figure 5, the following designations are used: Server - Data Center; LS – local segments of the network, STB - Set-top-box, which supports basic and extended functionality of user's TV receivers. Modern CDT networks has a number of specific features. Such features include: large and very large network size,

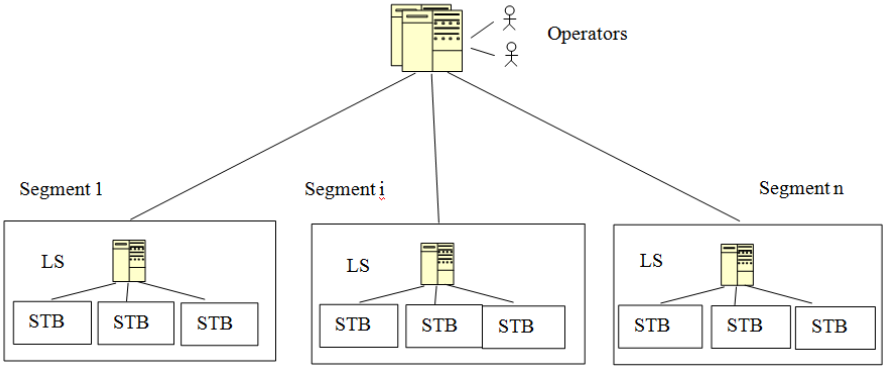

Fig. 5. The typical structure of CDT network.

high dynamic environment, heterogeneity of network infrastructure, strict requirements in terms of total cost of ownership, reliability, reaction speed etc.

Failure of even one elements of such a network can lead to avalanche effects when a local problem causes an avalanche-like increase in traffic between the local server and the STBs due to multiple attempts to receive/transmit data. It is obvious that all possible scenarios of behavior of such systems cannot be described in advance.

Monitoring of STB network interaction errors (so-called "last mile" errors) and receiver errors are particularly important. While monitoring CDT network it is necessary to take into account the following limitations when solving monitoring and control tasks: TV consoles, especially old models, have weak technical capabilities, data collection and transmission networks are characterized by low bandwidth, local servers, as a rule, have low performance. All these constraints define the context for MS development. In addition, the context is determined by end-user behavior.

CDT network operators have enough big toolbox for collecting and processing information about subscribers but the operators do not use these tools intensively. The problem is that existing restrictions do not allow realize processing of these information because this will lead to increasing network traffic and decreases the level of user services quality. For example it can increase the response time for user actions. In some cases the volume of data to be transferred can be many times more than real bandwidth of the network. Usage of model approach when models are placed in high performance central server and enough powerful region servers allow solve many mentioned above problems and essentially decrease the traffic inside the management network.

13 Conclusions

The main idea developed architectural model approach to MS development is that the TS is considered as partially observed IS, information (knowledge) of which for any moment of time is not available in full volume. Available information (knowledge) is stored in a form of dynamic models which include 3 groups of models: the models describing the structural dynamics, models describing behavior and models of transformation of representations.

The relevance of models is maintained by means of processing system events coming in the form of log- Thus, both MS and TS can be considered as agile architecture systems. Suggested approach is based on the algorithms of automatic synthesis of multilevel variable structure automata and process mining algorithms. Suggested approach was used in the process of development of a number of MS for different subject domains. Further direction of suggested approach development is solving problem of building MS for cognitive IS. This can be done by moving from hierarchical variable-structure automata to hierarchical probability variable-structure automata. This would allow realize learning procedures.

References

1. Sragovich V. Mathematical Theory of Adaptive Control World Scientific. Publishing Danvers, NJ, 2006. – 473 .
2. Dumas M., La Rosa M., Mendling J., Reijers H. Fundamentals of Business Process Management Second Edition. Springer-Verlag GmbH Germany, Berlin, 2018, – 527 p.
3. Weske M. Business Process Management. Springer-Verlag Berlin Heidelberg, 2012. – 403 .
4. Osipov V.Yu. Synthesis of effective programs for managing information and computing resources. Devices and control systems. 1998, No. 12. S. 24 - 27.
5. ITIL - IT Service Management. — URL: https://www.axelos.com/best-practice-solutions /itil
6. Blasch E., Bosse E., Lambert D. High-Level Information Fusion Management and System Design, Artech House Publishers, Norwood, MA. 2012. – 376 p.
7. Osipov V., Lushnov M., Stankova E., Vodyaho A. Inductive Synthesis of the Models of Biological Systems According to Clinical Trials. // International Conference on Computational Science and Its Applications (ICCSA 2017). Lecture Notes in Computer Science, vol. 10404. Springer, Cham. – Pp 103-115.
8. Osipov V., Vodyaho A., Zhukova N. About one approach to multilevel behavioral program synthesis for television devices // International journal of computers and communications. 2017. Volume 11, P. – 17-25.
9. Munoz-Gama J. Conformance Checking and Diagnosis in Process Mining. Comparing Observed and Modeled Processes, Springer International Publishing AG. Berlin Heidelberg, 2016. – 202 p.
10. Huang Jun, Hua Kun (Eds.) Managing the Internet of Things: Architectures, Theories and Applications. The Institution of Engineering and Technology, 2016. — 226 p.

11. Iyengar S., Brooks R., Distributed Sensor Networks. Taylor Francis, London. 2012-1140p.
12. Van der Aalst W. Process Mining Data Science in Action. Second Edition. – Heidelberg. Springer, 2016. – 468 p.
13. Rozanski N., Woods E. Software Systems Architecture. Working with Stakeholders Using Viewpoints and Perspectives. – Upper Saddle River, NJ. Addison-Wesley, 2012. – 715 p.
14. XES Schema Definition. http://www.xes-standard.org/