Towards Building of Cable TV Content-Sensitive Adaptive Monitoring and Management Systems

Vasiliy Yu. Osipov, Natalia A. Zhukova, Alexander I. Vodyaho, Andrey Kalmatsky, and Nikolay G. Mustafin

***Abstract*—**The problem of building of cable TV content-sensitive monitoring and management systems is discussed. Alternative approaches to solving this problem are analyzed. Architecture of perspective content-sensitive monitoring and management system with extended functionality is suggested. Requirements for automatic scripts generation are formulated and possible solutions are suggested.

***Keywords*—**Cable television, monitoring, content-sensitive systems, agile architecture

I. Introduction

Nowadays for information telecommunication domain one can see an evident trend to enlarging scales of IT systems under development and sphere of their application. One of the examples of such type of IT systems is digital cable TV (DCTV). Modern DCTV incorporate many thousands and even millions of subscribers and a number of subscribers permanently growth. Modern DCTV systems are distributed systems which include data communication networks front end and back end equipment.

Back end equipment is realized on the base of server clusters and is resident in operator data centers. Front end equipment is resident on client side and includes TV receivers, TV tuners and Set-top box (STB), which support base and extended TV functionality. DCTV monitoring and management is not a trivial task. As a rule, operation management systems (OMS) are used to solve such problem. OMS generalized structure is shown in Fig. 1

Data collected from STB are accumulated in data center servers. As a rule, one server works with a number of subnets. Each segment has its own physical or virtual local server (LS). STBs connected to receivers are resident on client side.

Generalized OMS has to solve 3 main groups of problems:

1) Processing data received from TV tuner. These problems are typical problems of technical support. The main of them are following:

- monitoring status of separate elements, groups of elements, infrastructure status and notification operator about incidents;
- registration information about errors, errors identification, localization and estimation possible approaches to error corrections;
- correction of errors;
- detection of reasons of error situation appearance;
- network and sub network situation assessment;
- mining of correlations between incorrect operation of separate elements and network and sub network situation and incorrect action of a user;
- prognoses of appearance of unwonted situations.

2) Assessment states of objects indirectly coupled with objects to be analyzed. This group includes following main tasks: assessment of status of data communication network, STB status, quality of external providers services (AVN server) etc.

3) Business data processing. The main tasks to be solved are following:

- mining information about subscriber preferences;
- forming user groups;
- prediction user behavior and estimation risks of loosing subscriber;
- modeling market dynamics in the case of service modernization and appearance of new services.

It is necessary to mention that similar problems have been solving for managing different network infrastructures. Nowadays a number of reference solutions (frameworks) of such kind have been developed, including a number of frameworks such as ITIL (IT Infrastructure Library) [1], eTOM (Enhanced Telecom Operations Map) [2] and others [10, 11]. But unfortunately one can not use them for managing DCTV networks. ITIL is oriented on usage for managing corporative information systems, eTOM – is oriented to be used by service providers. Both ITIL and eTOM give only description of functional descriptions. They can answer the question ***What is to be done,*** but they do not answer the question ***How is to be done***. Creation of the special framework for OMS would be very desirable, but the problem is that market of OMS is relatively small. But in any case ITIL and eTOM are to be taken into account by OMS developers.

Comparing OMS with ITIL and eTOM one can make following conclusions: i) OMS is a large or even very large scale network, ii) as a rule OMS network infrastructure is heterogeneous, iii) OMS must show very low total cost of

ownership (TCO), iv) OMS must show high levels of availability.

As it was mentioned above a typical DCTV has hundreds of thousands subscribers. For permanent monitoring of such system it is necessary to have big number of operators. Taking into account hard requirements to TCO, DCTV owners try to minimize number of operators as much as possible.

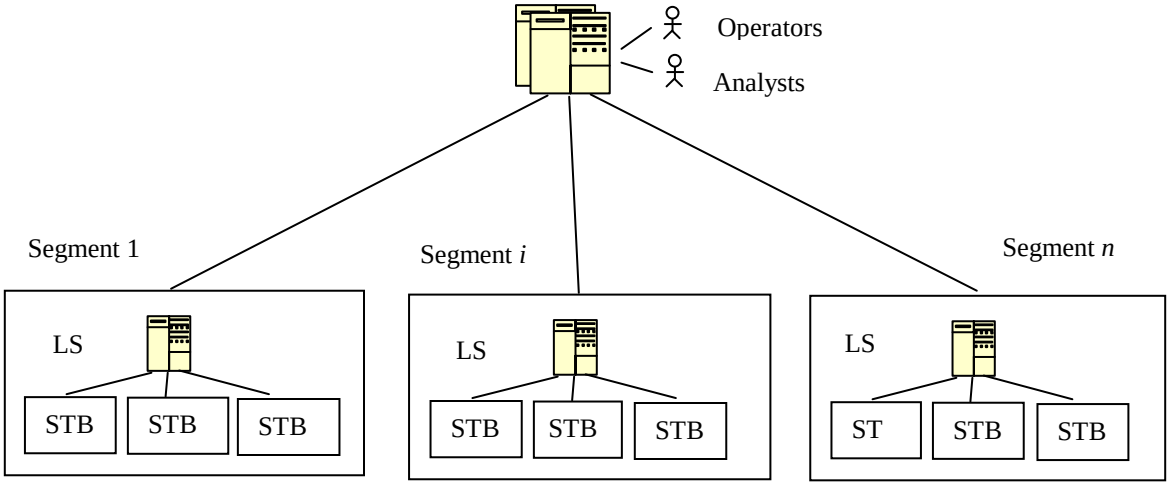

Fig. 1 Generalized OMS structure

Data communication networks used as transport subsystems for DCTV have heterogeneous structure and very often have low speed segments. So problems in low speed segment can be the cause of a global problem in a network. Equipment for DCTV produced by different company is not standardized and because of it, as a rule, DCTV developers cannot use general solutions for all types of STB. Different equipment has different characteristics, different architecture and different interfaces and build-in processors have different performance and different memory size. Bandwidth of LAN segments is not unlimited also.

DCTV market is very competitive one. DCTV owner tries to minimize TOC first of all by means of minimizing expenses for STB. Essential part of subscribers use old models of receivers and they don't want to pay for new equipment. Telecommunication companies also don't want to pay for new equipment.

As a rule a lot of errors and faults of different kind appear in a process of operation of old DCTV equipment. DCTV provider must be able to repair equipment as soon as possible.

The process of operation of OMS shows high dynamics. Load on the network and equipment is changing permanently depending on behavior of subscribers, network status, and it is very difficult to make prognoses about future state of DCTV and because of it one can meet a satiation when additional load created by monitoring system may cause system crash. In DCTV always presents risk of collapse when one can observe spontaneous growth of traffic between LS and STB because of repeating attempts of LS to receive information about STB status.

OMS developers meet following main challenges: i) very long life cycle of OMS development and installation. As a rule this process takes about 6 months. This does not allow upgrade software very often, because of high cost of each release; ii) the scale, DCTV structure, real tasks of the real operator define business logic to be realized by STB, so it is necessary to develop many types of OMS and each project to be implemented needs a lot of resources.

There are strong dependences between scripts and activities of operator and direct and indirect dependences between network components, their status and algorithms. List of errors of the STB is defined by the stack of protocols to be loaded in the STB. STBs of different types use different stack of protocols, which can be developed by OMS developer or by STB developer. Difference between stacks may be essential. There are no standards for error message formats.

Main requirements to OMS are the following: i) ability to work with big data (volume of collected data may reach dozens of Gigabytes), ii) agile business logic (ability for run time adaptation to context and operator actions, iii) low requirements to technical characteristics of STB and to network parameters, iv) low TCO, v) easy porting to different hardware and network platforms, vi) ability to be easily upgraded according to new changing requirements or appearing new requirements.

In order to satisfy all these requirements new architecture solutions are to be used. In the article an approach to building content-adaptive of DCTV OMS is discussed. Suggested approach allows build agile systems which can satisfy enumerated above requirements. In the section 2 of the article alternative approaches to building OMS are discussed. In the section 3 OMS business processes are analyzed. In the section 4 next generations content-adaptive OMS are discussed. In the section 5 architectural solutions of content-adaptive OMS are discribed. In the section 6 requirements to the script language, which can be conceded as a key element of the suggested approach are analyzed. Conclusions and directions of future research and development are discussed in section 7.

II. Alternative approaches to OMS building

Taxonomy of alternative approaches to OMS building is shown in Fig. 2.

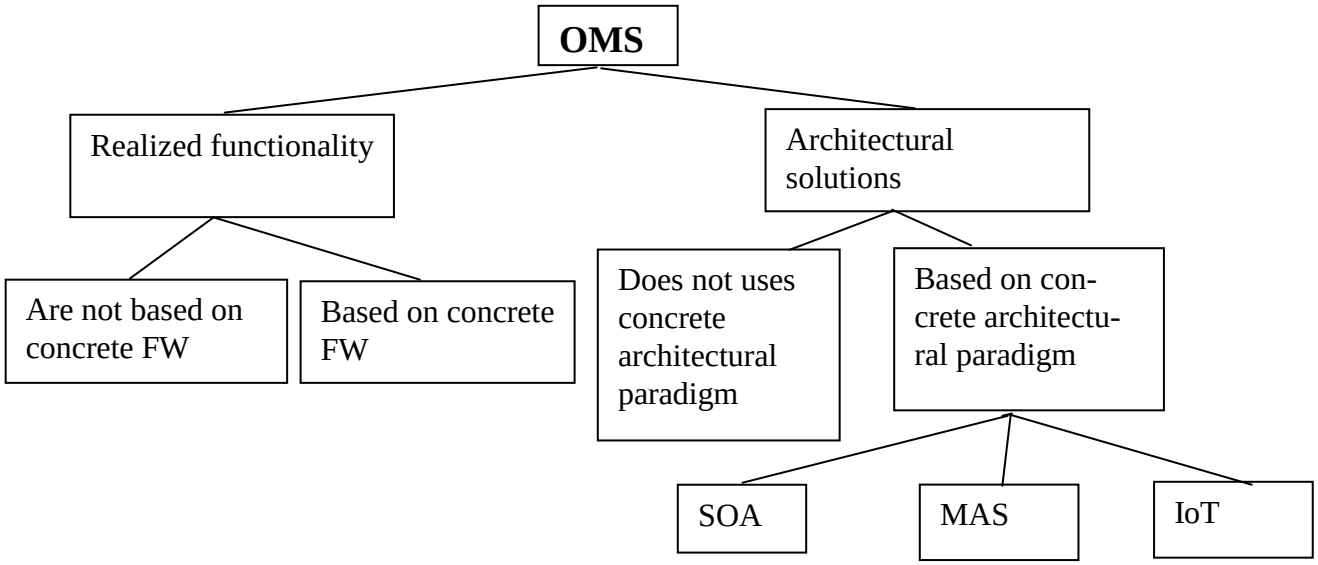

Fig. 2 Alternative approaches to OMS building

It is suggested to build OMS with the help of reference solution (framework, FW) or can realize needed functionality according to product owner requirements. The most part of existing OMS are built in such a way. They do not use any paradigm, but use a combination of technologies. By the most part developers use such architectural styles as Code on Demand, Event Driven Systems and service oriented architecture (SOA). Alternative approach is building OMS on the base of concrete architectural paradigm such as SOA, multi agent systems (MAS), Internet of Things (IoT) etc.

Nowadays there are no industrial standards for OMS building. Such standard protocols like SNMP [3] are used seldom, because, on one hand, they are to complex, on the other hand, OMS suppliers by the most part suggest platform dependent software components without SNMP support. So it is evident that nowadays process of OMS unification is in the very early phase.

The generalized structure of modern OMS is shown in Fig 3 [4].

LS is responsible for forming log files and supports log files storage operation. STB is a box on client side. As a rule it is rather simple device with a cheap processor with limited amount of memory inside. In real DCTV equipment produced by different companies are used. They have different instruction sets and error messages have different formats, so an operator has to us special instruction or script to define STB type. STB software includes diagnostic subprograms for receiver. These subprograms do not have build in adaptation mechanism. Monitoring system runs on the main server. It includes a toolset (OMS Tools) for solving separate monitoring tasks. As a rule it is a tool for working with log files. An OMS operates in a following way. Operator controls a number of STBs working at main server. They can receive information about status of concrete STS or group of STBs, send instructions, investigate system responses and analyze log files, such as instruction for finding specific patterns, sending specific diagnostic instructions, visualization and estimation status parameters. All OMS have GUI interface.

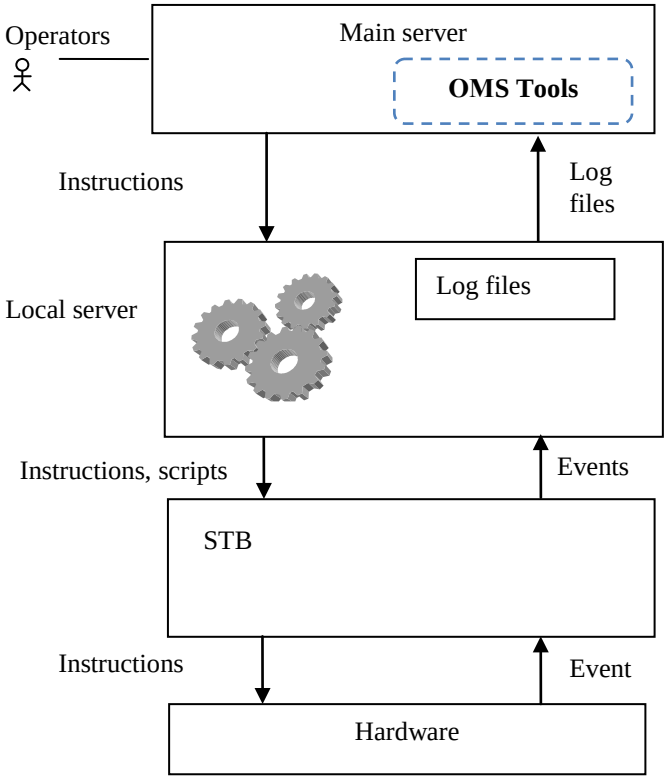

Fig. 3 The generalized structure of modern OMS

III. OMS OPERATION

As it is shown in fig.1 DCTV is 3-tier system. Lower tier is presented by STBs, middle tier is present by LS and on the upper tier we have the main server. Classification of possible approaches to business process organization in OMS is

presented in Fig. 4.

One can distinguish 3 main approaches to data processing in frames of OMS: main serve based processing, STB based processing and LS based processing. It is necessary to remember that typical OMS includes one main server on the base of cluster, dozens or even hundreds of LS and many thousands of STBs. Nowadays OMS are implemented on the base of dedicated network infrastructure. One can predict that in the future for implementation of OMS network infrastructure of smart house would be used and LS would be realize as virtual servers in fog environment [5].

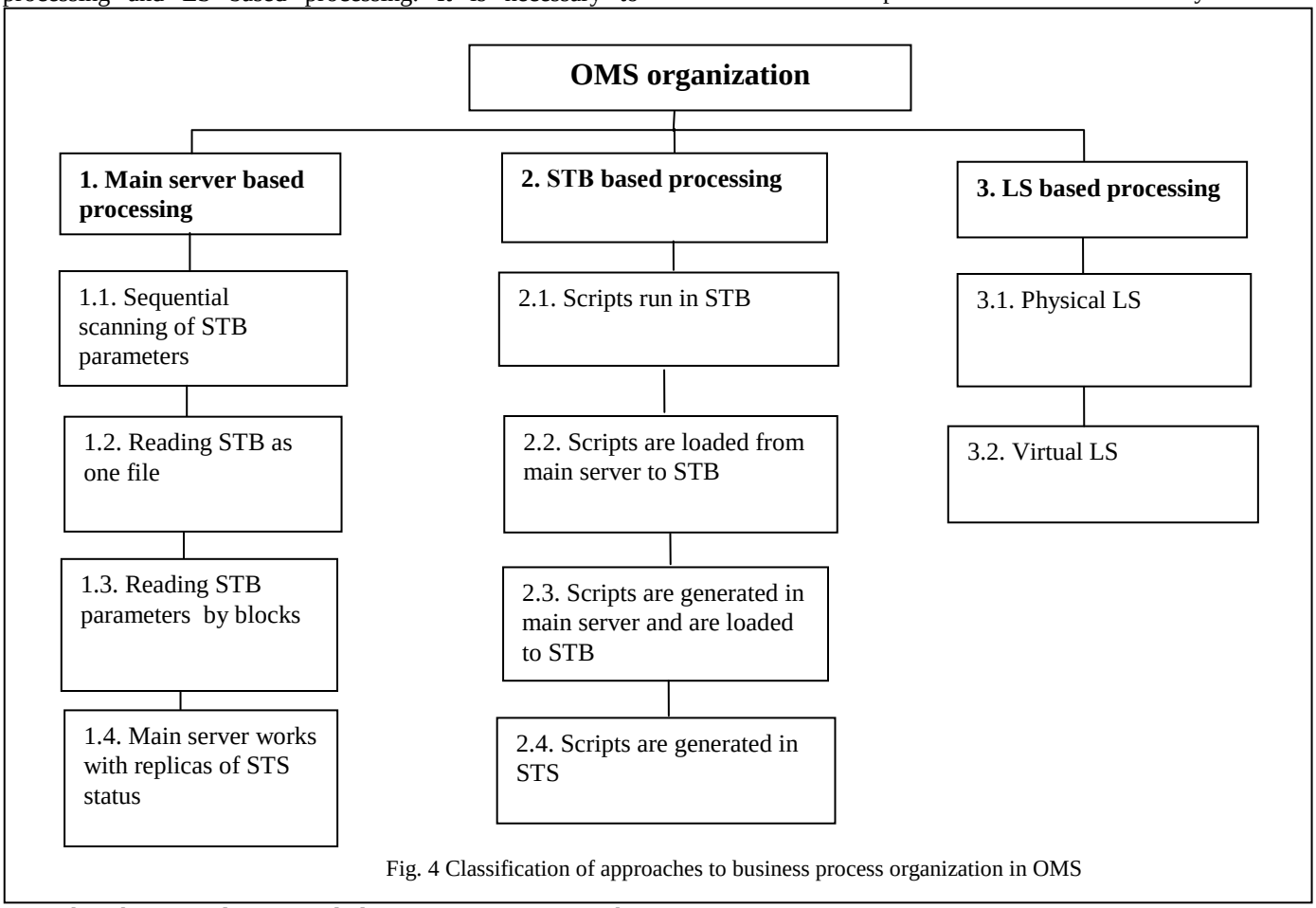

Fig. 4 Classification of approaches to business process organization in OMS

According to Fig. 4 there are 10 basic variants of business process organization. Variants 1.1-1.4 are main server based, variants 2.1-2.4 are STB based and variants 3.1, 3.2 are LS based. Let us discuss them in details.

Variant 1.1. Scripts are saved and run in the main server. STB status parameters are requested by need. In order to reed parameters one can use sockets, remote procedure call (rpc), REST or other protocols from microservice stacks [6]. This approach is a simplest one and nowadays it is rather popular because of simplicity. But it has 2 serious disadvantages. It is necessary to have big volumes of memory for saving all scripts. Volume of traffic between main serve and STBs may be rater big, and taking into account low bandwidth of separate network segments OMS performance may be rather poor. LS in this variant is used as log file storage.

Variant 1.2. After receiving error message main server reeds all parameters in one file. Reading parameters in bulk mode allows reduce response time, but length of the file may be very big and it may include a number of parameters which are not needed in particular situation. In real systems this variant is not used because it is less effective in comparison previous one.

Variant 1.3. In comparison with previous one this variant assumes that information about STB status is read by blocks. Structure of each block is correlated with concrete type of event. This approach is not bad, but its practical implementation is not simple because of many types of STBs. Additional problems take place with appearance of new types of STB.

Variant 1.4. This approach is very similar to variant 1.1. The main difference is that STBs status information are replicated to main server. Log files are also placed in main server data base. Different replication mechanisms can be used. The simplest variant is when STB informs about each status change by sending message. Usage of this approach allows minimize response time, but does not influence other characteristics.

Variant 2.1. All scripts are resident in STB. Technically it is possible to do only for STBs of new models with big volume of build in memory. Additional disadvantage of this approach is that it is necessary to have a lot of copies of the same scripts.

Variant 2.2. On the main server side in the script storage there are all necessary scripts for all error situation types and

all STB types. After receiving error message from STB or monitoring subsystem the system selects proper scripts and sends them to STB where they are executed by virtual machine (VM). Results of script execution are to be saved in script storage. This variant is rather interesting because scripts are not very sophisticated and one can use rather simple script language. So, virtual machine is also may be rather simple. For this variant there are 2 problems: a programmer has to write or adapt VM for each type of STB and analyst has to write scripts for all error situations and STB types.

Variant 2.3. On the main server side there is script generator. After receiving error message from STB or monitoring subsystem the script generator generates proper scripts and sends them to STB where they are executed by VM. Generated script and results of script execution may be saved in script storage and can be used in the same situation. Theoretically one can use information about results of script execution for adaptation script generator, but this question is out of the scope of this paper. It is the most interesting variant because it opens perspective of essential reducing TCO by means of reducing expanses for script generation.

Variant 2.4. This variant is very similar to previous one. The main difference is that script generator is installed on STB side. For new models of STB it is realizable, but for old models it is not realizable. Usage of both variants in the frames of one OMS leads to increasing of system heterogeneity and it is not good from the point of view of TCO. So this variant is less interesting then previous one.

Variant 3.1 and Variant 3.2 can be conceded as LS centric. In modern OMS LS by the most part are used only as a log file storage. It the frames modern approaches for OMS building the idea of realization of processing on LS is not very fruitful. The only reason to use LS main processing element is reduce loading on main server, but now main server is realized on the base of powerful clusters and good quality lines connect main server and LS.

But situation would be different if STB would be used as an element of smart house build on the base of Internet of Things (IoT) paradigm. This paradigm is tightly coupled with fog computing paradigm [5] which assumes presence of low level virtual computing environment. In this context one can use two strategies for integrating OMS into smart house environment: integrating physical LS into fog infrastructure (Variant 3.1) or realize LS as virtual server (Variant 3.2).

Results of comparison of different approaches show that nowadays the most interesting for practice is Variant 2.3 which assumes presence of script generator on main server side because of at least 2 reasons. The first reason is that it allows minimize number of scripts to be developed. The second one is that it opens perspectives of building adaptive script generators with can take into account previous experience while generating scripts.

IV. Content adaptive OMS

Content-adaptive OMS (CAOMS) can be defined as a run time system for DCTV monitoring and management which can realize content-adaptive solving problems of diagnostics and repairing.

As it was mentioned above the most perspective approach for building OMS is dynamic script generation on main server side and loading them for execution into the STB. Such approach can be conceded as separate architectural substyle - dynamic code on demand, (DConD), usage of which would allow solve the problem of preparing and saving of big number of scripts. Usage of VM on STB allows use single script language for all types of STB.

Algorithms of STB monitoring must take into account behavior of operators and context which are permanently changing. Additional problem is that it is very difficult to develop script library that cover all situations because as a rule there is no full classification of error situations is unavailable.

In existing OMS systems scripts are realized as monolithic code, VM are not widely used.

CAOMS structure. CAOMS realizes 2 main features: i) analysis of data received from STSs in run time, ii) generation and loading scripts into STB which can analyze situation and make repairing operations if it is possible. Generalized CAOMS structure is shown in Fig. 5. In comparison with existing OMS, CAOMS has 3 additional components: script generator, data analyzer and VM. Data analyze is to be used to process error events messages and log files.

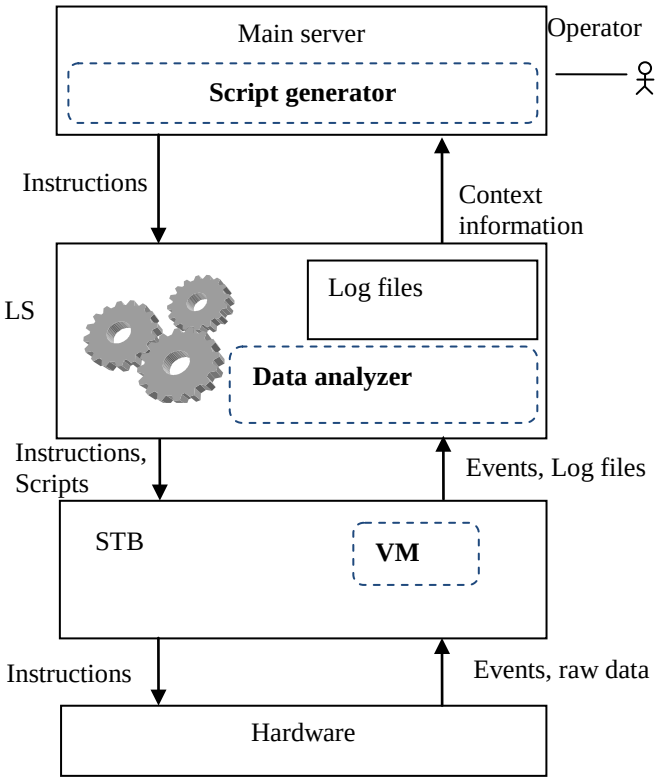

Fig.5 Generalized CAOMS structure

LS is responsible for solving 2 tasks: forming log files and loading scripts into STB, where they are executed by VM.

Logic of CAOMS operation. Logic of CAOMS operation assumes:

1. Cyclic monitoring of receiver status, this activity is realized by a daemon running on the main server. This daemon is responsible for collective monitoring status of the STB

network of the cable operator. Monitoring of the status of receiver is executed in accordance with algorithms that are built in VM. These monitoring scripts are loaded into VM on the stage of tuning of STB. When an error is detected error message is formed and send to server. Program structure depends upon STB model and network structure. This software is to be upgraded seldom. Sometimes it is necessary to stop or run a process of monitoring depending on situation i.e. the load on the network.

2. When main server receives error message from STB, the script generator generates the script of additional diagnostic. Automatic building of scripts can be realized on the base of formalalisms known as operational conditionally finite state automata [7]. The result of syntheses is a sequence of operators on script language.

- S_4^{STB} – collected diagnostic information is send from STB to main server;
- S_4^{BE} – main server has received diagnostic information from STB;
- S_5^{BE} – diagnostic information collected in STB is processed;
- S_6^{BE} – script with reactions to error situation is formed by main server (an operator may take part in this process);
- S_5^{STB} – script with reactions is executed in STB;
- S_F^{STB} – correct STB status.

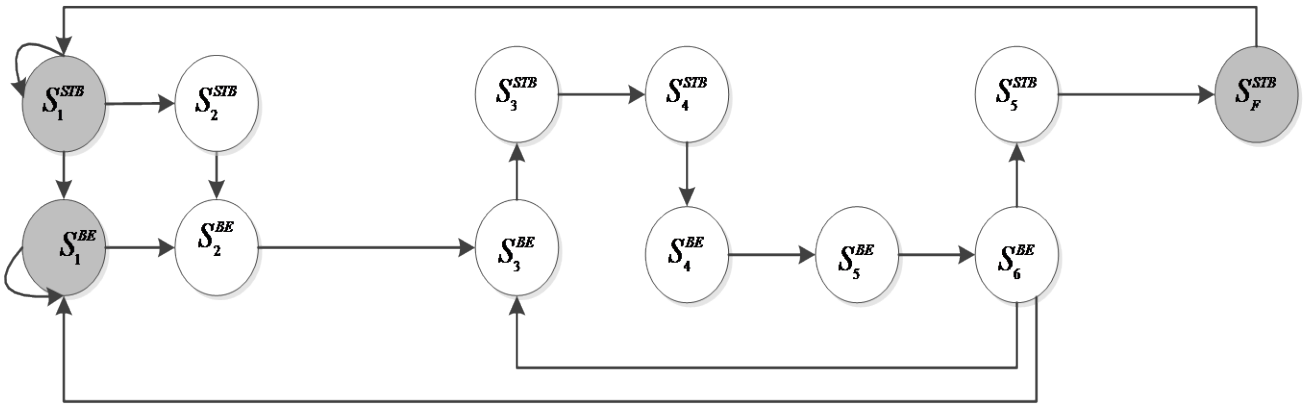

Fig.6 The graph of generalized states of monitoring and repairing process for CAOMS

3. Generated script is send to STD and executed by VM. The main task of script execution is to collect diagnostic data. Results are sent to main server, log files are sent to LS.

4. Main server analyzes results of diagnostic scripts execution, realizes procedure of situation assessment according to approach suggested in [7].

The graph of generalized states of monitoring add repairing process for CAOMS is presented in Fig. 6 where

- S^{STB} graph vertexes which correspond to dedicated receiver states, S^{BE} correspond to server dedicated states, arcs correspond to transitions between states;
- S_1^{STB} – planed monitoring of receiver;
- S_1^{BE} – planed monitoring of the receiver status realized by center server;
- S_2^{STB} – error situation in STB, notification message is send to main server;
- S_2^{BE} – in the process of planned monitoring in the main server STB error is fixed;
- S_3^{BE} – diagnostic script for STB detailed testing is generated on main server side;
- S_3^{STB} – diagnostic script is loaded and executed in STB, necessary information is collected;

V. OMS CAOMS ARCHITECTURE

According to developed approach to CAOMS building each STB includes Agent module which is responsible for forming context or VM operation (Fig. 7). Compiled modules loaded into VM define logic of its operation.

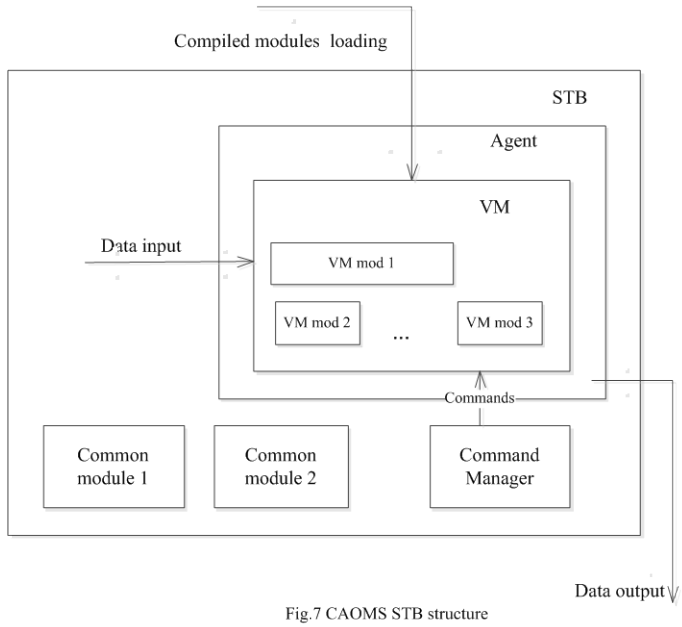

Fig.7 CAOMS STB structure

As it was shown above in section 3 for building CAOMS the

most interesting are variants 2.2 and 2.3 which assume loading ready or generated scripts from main server into STB.

There are a number of approaches for practical implementation of mentioned above variants: i) generated scripts are saved in a cache, ii) an operator can make scripts manually or edit generated scripts using special tool with GUI interface, iii) an operator can create new scripts on the base of existing scripts.

Usage of script languages is one of the key elements of CAOMS architecture. There are 3 alternative approaches for script language choosing: i) usage of standard general purpose language such as Java, ii) usage of standard script language such as Perl, Python, iii) usage of domain specific script language (DSL) [8]. Usage of standard languages allows use existing VM, but automatic generation of scripts is not a simple problem. Standard languages have a lot of features not needed to write diagnostic scripts. So usage of DSL is more attractive perspective. Diagnostic script DSL should have following features: i) be effective while processing text messages, ii) be able to realize logical processing, iii) support trigger operations iv) be able to work with business rules.

Supporting toolset has to satisfy following requirements: i) support automatic script generation, ii) be able to correct generated scripts, iii) be able to create scripts in manual mode.

It is possible to use different approaches to script generation. First of all, one can use tools for working with business rules, for example Jess [9]. In this case an operator writes test in terms of business rules. So, operator can do it in business terms. The problem is that it is necessary to add new rules when new models of STB appear. This approach has all limitations of expert system approach.

It seems that more effective is an approach based on usage of ontologies. In this case for every new model of STB its ontological description is formed. This description is used to generate scripts. Ontological description for new STB models can be developed on the base of existing models.

VI. Conclusion

Nowadays main problems in the sphere of OMS development have roots in severe requirements to TCO and necessity to deal with STBs of old models, working in heterogeneous structures. Usage of CAOMS can be conceded as an effective solution of the problem of reducing TCO. For the modern CAOMS automatic script generation on the main server side is the most effective solution.

Future directions of CAOMS development should be coupled with such paradigms as smart houses and IoT. In this context STB can be conceded as an element of smart environment. Script generation can be moved to LS which can be realized as virtual server as done in fog environment. For successful migration of CAOMS to the new IoT it is necessary special methodology of migration to this platform. The process will take at least some years. During this period one part of subscribers will work with new equipment and other part will work with old and very old equipment. So, agility of suggested approach would be very useful for solving the high heterogeneity problem.

References

[1] ITIL. Official Site. https://www.axelos.com/best-practice-solutions/itil

[2] TM Forum. https://www.tmforum.org/

[3] An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. https://tools.ietf.org/pdf/rfc3411.pdf

[4] Ciciora W., Farmer J., Large D., Adams M. Modern Cable Television Technology: Video, Voice, and Data Communications. Second Edition, Elsevier, 2004.

[5] Fog computing. http://www.webopedia.com/TERM/F/fog-computing.html

[6] Newman, S. Building Microservices. O'Reilly Media. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA, 2015.

[7] Osipov V.Yu. Automatic Synthesis of Action Programs for Intelligent Robots. Programming and Computer Software, 2016, Vol. 42, No. 3, pp. 155 – 160.

[8] Kelly S., Tolvanen J. Domain-Specific Modeling. Hoboken, New Jersey John Wiley & Sons, Inc., 2008.

[9] Boyer J, Mili H. Agile Business Rule Development. Berlin Heidelberg, Springer-Verlag 2011.

[10] Brezillon P., Gozalez A. J. (Eds). Context in Computing. A Cross – Disciplinary Approach for Modeling the Real World. Springer Science + Business Media New York, 2014.

[11] Sungur T., Breitenbucher U., Leymann F., Wieland M. Context-sensitive adaptive production processes. Procedia CIRP, 41 (2016) 147 – 152.