Journal of Future Internet
2014 Vol.1, No.1, pp.1-15
ISSN(e): 2409-790X
ISSN(p): 2413-8401

VERIFICATION METHOD OF COMPLEX WEB-SERVICES

Tkachova Elena[1] --- Issam Saad[2] --- Abdulghafoor Raed Yahya[3]

[1]*Telecommunication systems, Kharkov National University of Radioelectronics, Ukraine*

[2]*Kharkov National University of Radioelectronics, Ukraine*

[3]*Odessa National Academy of Telecommunications named after O.S.Popov, Ukraine*

ABSTRACT

In the article proposed formalisms for defining rules of Web-services orchestration and choreography that allows to combine elements of a distributed system in a single system. Proposed approach allows to perform formal association of individual system components into a single unit. Proposed verification method for distributed systems based on the model approach. E-networks are models for verification of distributed systems. Web-service's safety check is performed by finding deadlock conditions or any deviation from the specification process. This method allows taking into account the asynchronous nature of complex services and also performing stateful inspection: check for different Web-service consistency, check orchestration and choreography compatibility requirements for each service.

Keywords: Web-service, E-network, Verification, Distributed system, Service-oriented architecture, Formal language.

Contribution/ Originality

This study uses new algebraic methods for the analysis of complex Web-services. The novelty is in the use of formal grammars, which solved the problem of validation and evaluation of the behavior of the resource allocation of info communication protocols.

1. INTRODUCTION

In recent years the main trend in the business field is to transfer many processes in the global multiservice networks. Web-services are used to distribute different type of services over the Internet. Technical realization of such services is able due to using Web-services within the service-oriented architecture (SOA) paradigm [1, 2]. Web-services architecture becomes complex greatly. Complex and distributed services have been increasingly used. The main purpose of the developers in this case becomes a flexible and proper combination of different elementary (single) services in to composite (combined) services. Elementary services are used as a component of higher level services, including the possibility of recursion-combining composite servicesin to more complex structures.

SOA paradigm allows describing the relationship and interaction among consumers, service providers and service brokers. This allows using the abstract environment functioning services. Also important benefit is possibility of submit the requests to multiple services simultaneously

within the SOA to provide services with required quality, reliability and flexibility which needed modern distributed systems. Additional conditions such are ability to manage exceptions, the definition of integrity execution cycle service, error handling and possible timeouts work. All of this ensures availability of resources.

Web-services are implying flexible, dynamic and highly adaptive distributed system components that work to provide ever-changing needs of users. Developers and customs face a number of problems in developing distributed systems which based on SOA requirements. Composition of services provided by different vendors is not always possible, a number of difficulties arise in the implementation of long-term distributed operations (this is related to a small block of time necessary to perform resource). When analyzing existing specifications, there is a lack of common concepts and standards for facilities and structures of interaction between different components of a complex service, etc.

The main authors devoting his attention to solving the problem of providing quality Web-services are J. Alonso, C. Peltz, Casati, as well as corporations such as W3C, Oracle, Hewlett-Packard, and many others. The main solving concept is development of the formal description of the service's interaction in distributed multiservice systems [2-4].

This paper devoted to analyzes the existing description languages used for the composition and interaction of services, their principles of operation of the joint, as well as the technique of verification for distributed systems, which is based on the model approach and using E-network.

2. WEB-SERVICES ORCHESTRATION AND CHOREOGRAPHY

Orchestration in general term, means «modeling aimed, internal business processes». Web-services choreography means «interactions specification between autonomous business processes» or sequencing conditions for independent participants exchange messages [1, 2].

Web-services technologies in many cases still do not have common definitions and executing scenarios: lack of universal language for describing service components, uniform standards and performance distributed systems specifications. The terms orchestration and choreography describe two aspects of creating business processes from composite Web- services. According existent approach [3], these concepts can be divided into private (internal) and public (external) interaction.

Orchestration of business processes related to particular concepts and describes the process that occurs between the services (as a set of atomic services) which are controlled by one of the parties of participating in service, and defines the logic selection processes. Services orchestration implies a capacity to monitor the implementation of software applications (e.g., applications, components, services) to achieve the desired result. Build a logic-based representation of the process standardizes the composition across the organization(party providing the service) [3, 4].

In orchestration, the involved Web-services are under control of single endpoint central process (another Web-service). This process coordinates the execution of different operations on

the Web-services participating in process. This differs from choreography, which is more collaborative and allows each involved party to describe its part in the interaction. Choreography tracks the message sequences among multiple parties and sources - typically the public message exchanges that occur between Web-services - rather than a specific business process that a single party executes [3]. Only the central process (coordinator of the orchestration) is conscious of this aim, thus, the orchestration is centralized through explicit definitions of operations and the invocation order of Web-services.

Choreography, in contrast, does not depend on a central orchestrator. Each Web-service that participates in the choreography has to know exactly when to become active and with whom to interoperate. Choreography is based on collaboration and is mainly used to exchange messages in public business processes. All Web-services which take part in the choreography must be conscious of the business process, operations to execute, messages to exchange as well as the timing of message exchanges [3, 4].

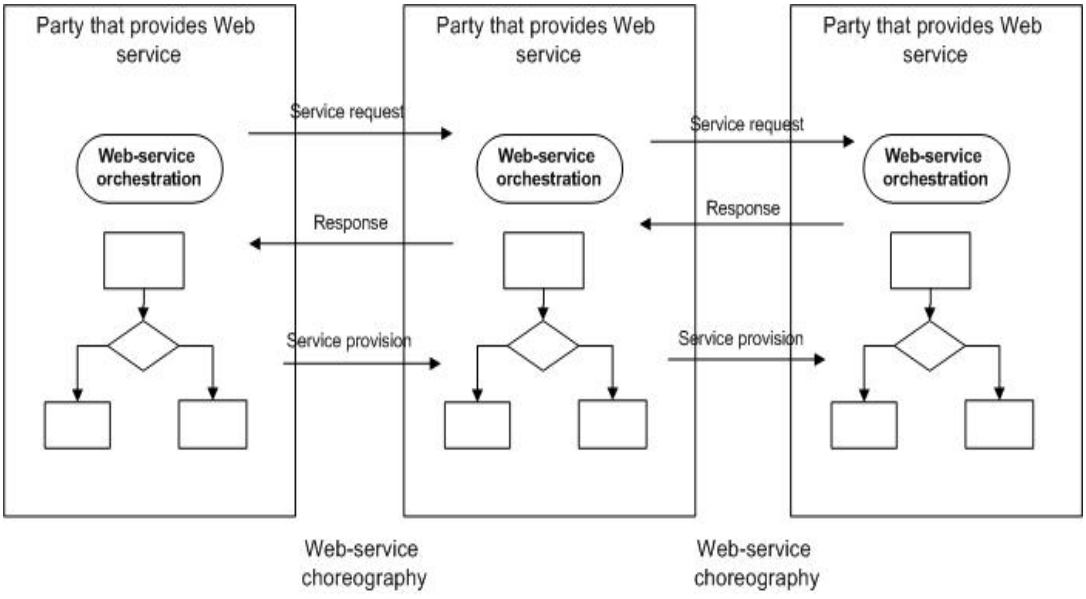
An example is the process of establishing connection of various application layer protocols (SIP). Important in the interaction of business processes is to comply with the correct sequence of their execution. Table 1 shows the major components that are formally harmonizing concepts and service composition.

Table-1. Web-services orchestration and choreography in distributed systems

Choreography	**Orchestration**
Public business processes. It is publicly available to all participants in the interaction.	Private business processes. It is confidential, accessible only to a certain process or service.
Describes the interaction between services that includes a set of protocols which work together.	Logical relationship between the processes during the service formation.
Key components are messages and his structure, asynchronous and synchronous communication services, service messages.	Key components are participant and his role, variables and properties that define the interaction between participants, error handlers, events, and logical connectives between events.

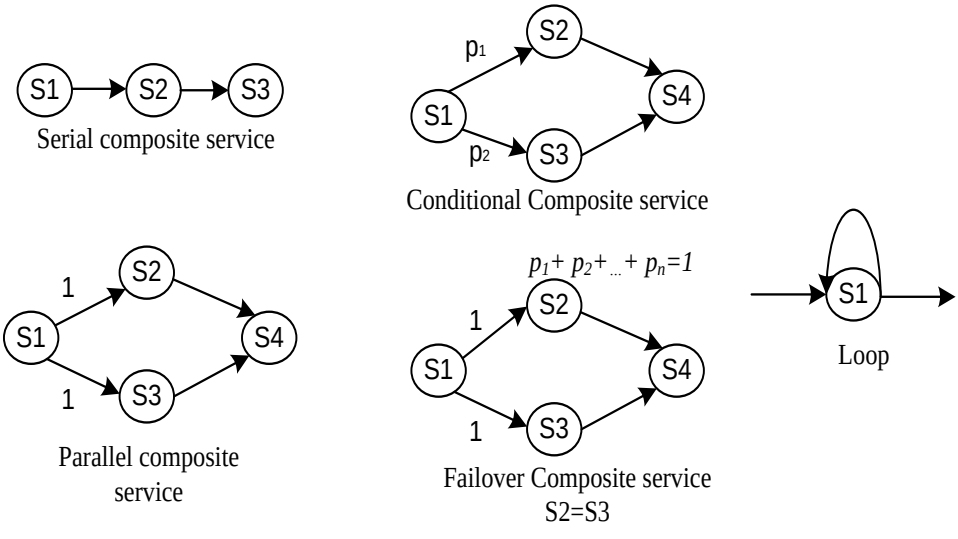
Developers should consider the limitations imposed by existing coordination protocols interaction of end users and the internal rules of operation services. Figure 1 shows example of Web-services orchestration and choreography.

Fig-1. Structural diagram of interacting Web-service compositions



Complex services are composed of several atomic services united in accordance with defined structure (scenario). These scenarios represent the execution order of atomic services in the composition of the complex. In Figure 2 depict five major operators which can be described by the complex structure of the service: serial, parallel, conditional, loop and failover.

Fig-2. Operators for complex Web-services



Technical requirements for correct Web-services execution in accordance with the SOA concept are [3, 4]:

1. Flexibility. Separation between the logic of the process and services.

2. Simple and structured activities. The process of harmonization of services should support actions for access other Web-services, and to describe process semantics. Simple action can be considered as a component that interacts with something out of the process, while the structured action controls the overall execution of the process by specifying the composition and procedure.

3. Recursive composition. A separate business process may interact with a plurality of Web-Services. The process can be represented as a Web-service aggregation process in a higher level.

4. Serial transaction processing and correlation queries. Long to execute services should ensure transactional integrity and exception management, as well as open access to many resources.

3. OVERVIEWOF EXISTING SOLUTIONS FORBUSINESS PROCESS MODELING

The Web-services specifications offer a communication bridge between heterogeneous computational environments used to develop and host applications. The future of E-business applications requires the ability to perform long-lived, peer-to-peer collaborations between the participating services, within or across the trusted domains of an organization.

The first language for the sequential description, parallel and multi variant workflow specification is BPEL4WS (BPEL). BREL allows simulate the behavior of Web-services in the business processes interaction [4, 5]. It is an integral part of the grammar-based XML, which is intended to describe the control logic in coordination Web-services involved in the workflow of a business process.

WSCI. In general, the WSCI defines the interaction or exchange of messages between Web-services. The specification provides message correlation, ordering rules, exception handling, transactions, and dynamic interaction [6]. In WSCI separate process cannot manage the interaction. Each elementary action WSCI is a single process associated with a specific operation WSDL. WSCI specification extends WSDL, allowing the execution order of operations available WSDL. In other words, WSDL for each available service describes the entry point, and WSCI defines the interactions of operations WSDL.

BPML is a language for describing business processes based on XML. BPML provides structure for describing the actions and workflow business process [2, 5]. Along with the structural actions for organization branching sequential and parallel processing cycles and synchronization process provides the basic steps for sending and receiving messages, call services.

The language was designed for controlling the duration of processes, and includes long-term storage function. Data exchange between the participants being in XML, using the definitions of roles and partners, such structures BPEL. BPML also supports recursive composition for forming the composite business processes. BPML supports both long and short-term transactions using rules to control compensation notion of context, which is similar to that used in BPEL [7].

The formal grammars are another language, which is widely used in the formalization of the run sequence of various processes [8].They allow precisely describe sequence of events, and to

the execution of parallel processes. The advantage of formal grammar is step by step of Web-service implementation, can represent Web-service's choreography and orchestration in one formal notation. In this paper we applied formal grammar in verification approach.

4. ORCHESTRATION AND CHOREOGRAPHY WEB-SERVICES COORDINATION

Today there are no standardized communication protocols of services collaboration, which are represented by different developers. In general, collaboration protocol relates to the organization of communication between several members of the business process, while the composition of services is a private (closed) process, controlled by a separate party. BPEL language supports both executable processes, and the organization of interaction between them. BPML and WSCI can work together so that BPML simulated execution of the business process, and WSCI - choreography of Web-services [1, 4, 7].

Formal representation of elementary Web-service is as follows: $S: Inp \rightarrow Out$.

In this case S – a service; Inp – a finite set of input parameters; this parameters affect to business process performance, $\{Inp1, Inp2, Inp3 \ldots Inpn\}; Out$ – finite set of output parameters$\{Out1, Out2, Out3 \ldots Outn$.

Expanded view service encompasses description of the conditions that are required to service perform (preconditions), and changes subject area. Implementation of Web-service generates these changes. The setOut is dividedinto two subsets $Output$, and $Effects$in this case. $Output$is directly output parameters; $Effects$ is effects of service, can change the state of the service environment.

$$S: Pre \rightarrow Output \cup Effects \quad (1)$$

Formula (1) represents a necessary condition for the fulfillment service. A necessary and sufficient condition has the form:

$$S: Pre \cup Input \rightarrow Output \cup Effects \quad (2)$$

Sets Inp; Pre; $Output$; $Effects$, given in formula(2) are given in the specification of the design of a specific service. Sets depend on the subject area of service provided.

Using formal methods for the composition of services in accordance with the concept of SOA and BPEL specification language can be defined by the following set of expressions:

$P := Stop \mid Skip$ – start or pass service composition process;

$\mid inv!ch\tilde{} \rightarrow P$ – call for service or process that is involved in providing the service;

$\mid inv?\tilde{}x \rightarrow P$ – identification of caller service. This service participates in providing complex service;

$\mid ch!exp \rightarrow P$ – input interface(input value of request processing);

$\mid ch?x \rightarrow P$ – output interface (output value);

$\mid x := exp; P$ – service assignment;

$\mid if\ b\ P\ else\ Q$ – conditional services branching;

$|\ P \blacksquare Q$ – logical choice of action in service composition;

$|\ P\ \Delta\ Q$ – interrupt a service or process;

$|\ P\ |||\ Q$ – services interaction;

$|\ P;\ Q$ – sequential execution.

P, Q – Web-service components. These components are involved in negotiating and holistic view of service. It can be basic services or processes that involved in providing the service, the symbol ■can represent anylogical operation ($\cup,\ \cap,\ !,\ ||,\ \&$).

Choreography services can be described by global protocol that governs the conduct of the participants interacts with each other. Each participant has their own processes, thus directs the coordination of interaction that meets with their requirements. Elements of Web-services can be represented as follows:

$X ::= Stop|Skip$ – inaction and termination of service;

$|svr(A,B,ch\tilde{}) \rightarrow$ X – call for service in accordance with certain performance conditions;

$|ch(A,B,exp) \rightarrow$ X – opening a communication channel with a specific interface;

|x := exp; X – service assignment;

|if b X else J – condition for implementation;

$|X \blacksquare J$ – one or several messages selecting;

$|X;J$ – sequential messages execution.

There X, J – processes of Web-services providing.A, B – input conditions.

For example, consider a communication channel between two users: the sender (S) and the receiver (R). The equipment is provided with an output buffer of the sender (out), and the recipient - input (in). Both buffers have unlimited capacity. If S sends a message m to R, then it puts the message in your output buffer out. The receiver receives the message by removing them from their input buffer in [9].

Any service can be exposed to a number of critical factors (e.g., service may be blocked after the one of the participant's buffer will be overflowing). Errors formula of service performing can be represented as follows:

$$X(inv!\ ch?1(start:mess1) \rightarrow (mess1:RETR(A,B,exp)) \rightarrow (S:messDelete \mid x)U(mess1:DELE) \rightarrow$$
$$\rightarrow X(mess2ch?x) \rightarrow DeliveryDetails(exp;P) \rightarrow (Stop)$$

The above formula describes process of informing the undeliverable message 1when the input buffer is full:

$$G((m:out \wedge m:in)U\neg(m:out)) \mid\rightarrow G(m:out \wedge \neg m':out \wedge F(m':out)$$
$$\rightarrow F(m:in \wedge \neg m':in) \wedge F(m':out)))$$

Formula (3) presents the composition and interaction of Web-services: the requirement of «channel preserves the order of transactions». Formula describes the orchestration of services to

the following condition "after receiving confirmation that the message has been delivered to the recipient, is it from the buffer. Out".

5. VERIFICATION APPROACH

Web-services verification is differ from the verification of telecommunication protocols. Web-services can change the value of current parameters in real time execution; consider an asynchronous mode for interaction and the ability to access multiple resources of a distributed system [7,8].

The main requirements for Web-services in distributed system are [3, 4, 10]:

Interface Compatibility – focusing on semantics of correlating invocations against receiving and message replies between partner processes.

Compatible interfaces can be defined as follows (formal notation):

$$Input : O \to message \cup Input(o) = M(O), there\_o - [inv?m], [inv!\ m]$$
$$Output : O \to message \cup Output(o) = M(O), there\_o - [inv!\ ch?], [inv!\ ch\ \ ]$$

There O is an orchestration or process of services agreement for Input and Output interface.

m are factors. m-factors determine type of services. *ch* is a service challenge, this service need for

work beginning

Safety Compatibility – assurance that the composition is deadlock free and is checked against partial correctness of transitions. It provides a reliable layer for the exchange of information between parties, guaranteeing the delivery of information with an exactly-once semantics.

This requirement generally refers to the choreography. Interactions function correctly if for any multiple processes has a constant amount of executed process on the future:

$$X = \{m(P, A, B), \phi_{input\_output}(o)\} \to \phi_{input}(x) = \phi_{ioutputt}(y) \quad (3)$$

There ϕ defines the execution process logic 0, x and y is a variables, dependent on input conditions A, B x and y have the following type of relationship:

$$receive[o](x) \Rightarrow reply[o](y)$$

Liveness Compatibility – assurance against starvation of progress (that the service process eventually terminates) and those messages received are served on a first-come-first-served basis. Requirements must be satisfied with respect to the quality of service (delay time, the sequence of transactions). It provides a reliable layer for the exchange of information between parties, guaranteeing the delivery of information with an exactly-once semantics.

Formally, this requirement may be represented as follows:

$$M(S) = (S_0 \to S, L, pre), \quad (4)$$

S is a set of states services or processes, providing a composite service, S_0 is an initial state (service challenge). The rule of services interaction formula: $\to \subseteq S_0 \times pre \cup S \times pre \cup L$

An example of interaction and following the successful implementation of business processes. In the implementation of banking services participant has the following list of operations: $o1[?getF; !ListF]$ and returns a list of following transactions$o2[?getCho]$ and provide participant two possible transaction, $o3[?cancel]$ - denial of service, or $o4[?payment]$ – send payment confirmation:

$$A = (PA = fC : o2g; InA = fo1; o2; o3; o4g; BA(x; y))$$
$$with\ BA(x; y) = rec[o1](x); rep[o1](x); rec[o2](x); scope[time, f(rec[o4](y);$$
$$inv[C : o2]); (rec[o3](x); empty)g].$$

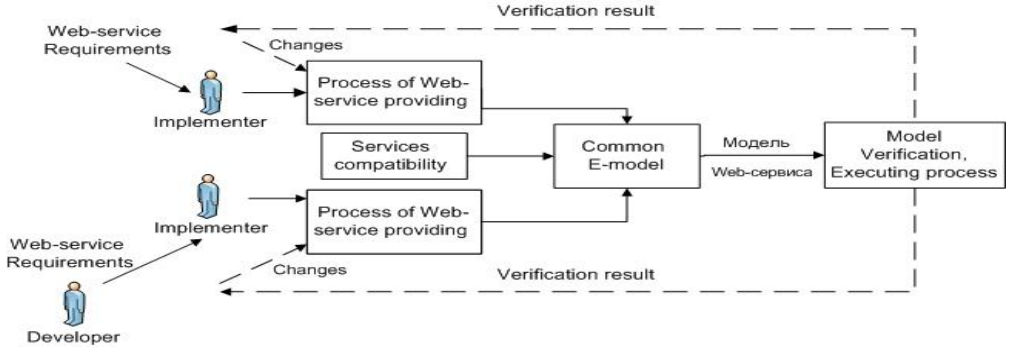
More over modeling approach is used for verification multiservice distributed systems that have real-time application. Using this verification approach, the properties and characteristics that should have a distributed system is formally defined. Verification takes into account the structure and behavioral properties of processes.

SOA has a complex, distributed structure that can be dynamically changed. E-networks are the most suitable apparatus for simulation networks that constructed on SOA concept. One of the advantages of representation of the functioning of multiservice networks (or Web- services) in the form of graphs E-networks is to provide convenient and visual modeling algorithm. One of the most important properties of E-network is modeling protocol to verification on safety and liveness properties [11].

As models for distributed systems verification is proposed to use the E-network. Web-service's safety check is performed by finding deadlock conditions or any deviation from the specification process. Safety check can be performed in the analysis of properties such as boundedness and coverability. Web-service's liveliness checking is performed by finding reachability and liveness state in E-network model. This positions lead to outcomes that match the requirements of the specification [11, 12]. In other words, service orchestration reaches its completion; verification of this property can be reduced to coverability and reachability.

The process of verification of distributed systems involves close interaction between the provider (organization providing process)and developer ofWeb-service [13, 14]. Important point in the construction of a common service model is the representation of correct interaction of internal and external processes of the various participants. Figure 2illustrates the design of the Web-services verification process [15].

Fig-3. Verification approach



In the verification process Web-services, perform the following steps :

1. Formal representation of the composition process of each service providing;

2. Formal representation of the interaction service composition;

3. Building a model for complex Web-service witch based on E-network;

4. Analysis of the model Web-service properties such as boundedness, liveness, reachability, coverability.

5. Service verification based on the model approach: check for different Web-service consistency, check for compatibility requirements orchestration and choreographyfor each service.

6. RESULTS VERIFICATION OF AGENT-BASEDCONTROL PROTOCOL

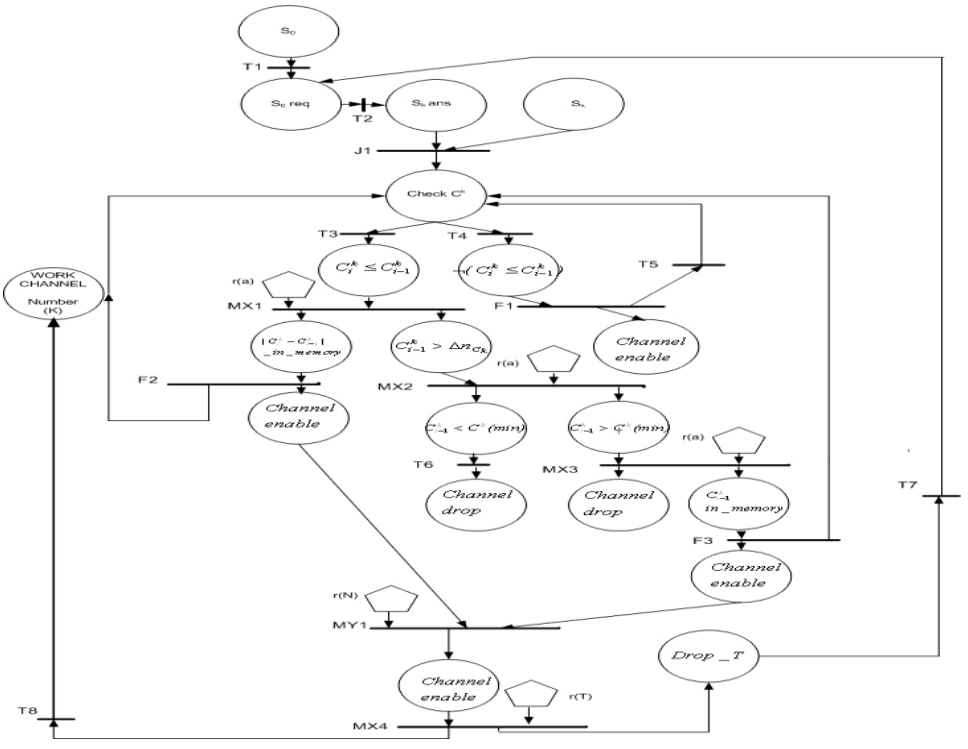
As a practical implementation of the proposed method we built the formal notation that describes the behavior of load balancing protocol in multiservice network. Considered distributed system (multiservice network) using an agent-based control that build on E-model.

The main goal of control agent is to implement protocol, which is responsible for overload balancing in communication channels within formation exchange between components of a distributed system.

E-network model for determination numbers of unloaded communication channels with allowable bandwidth is depicted on Figure 4.

The meanings of transitions in E-models are: *T1* – querying the control agent to the incident channels; *T2* – sending request for communication channel; *J1* –channel response(answer is possible only when you know the necessary attributes of the channel); *T3*, *T4* – the channel capacity change checking; *MX1*, *MX2*, *MX6*, *MX5*, – the communication channel performance checking with decreasing bandwidth; *T6* – revalidation response; *T7* – sending a message about the communication channel suitability; *F1*– the quality of service compliance checking; *F2*, *F3* – entering values of bandwidth decrease in memory; *MX4*– verification of clock period compliance; *MY1* – metric system verification; *T8* – new request for bandwidth check; *T9*– issuance of the operating channel numbers.

Fig-4.E-network graph for communication channels with the allowable bandwidth



Positions correspond to the following states: s_o - the initial state(the definition of communication channels control agent); $s_o req$ - generating request for discovery incident communication channels; $s_k ans$ - response generating; s_k - providing information about communication channel; $Check\_C^k$ - definition of communication channels with the allowable bandwidth; $|C_k^i - C_{k-1}^i|\_in\_memory$ - fixing the value of change the bandwidth between two time interval; $Channel\_enable$ - match cannel as active (allowable bandwidth); $Channel\_drop$ - match channel as ignore; *WORK CHANNEL* - definition of working communication channels; *Drop T* - reset clock period counter.

Set of final states should be clearly defined using the proposed approach. The final step in the procedure is a condition WORK CHANNEL, which keeps the communication channel number corresponding requirements.

Behavior of the system is as follows (in accordance with the developed method and the selected type of language):

$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a = 0) \rightarrow$$
$$((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i) \_ in \_ memory) \rightarrow$$
$$\rightarrow (Check\_C^k) \vee ((C_k^i - C_{k-1}^i) \_ in \_ memory)(Channel \_ enable \mid r(N \mid N < N(\mathbf{max})) \rightarrow$$
$$\rightarrow (Channel \_ enable \mid r(T \mid T < T_z))WORK \_ CHANNEL$$
$$\cup$$
$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a = 0) \rightarrow$$
$$((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i) \_ in \_ memory) \rightarrow$$
$$\rightarrow (Check\_C^k) \vee ((C_k^i - C_{k-1}^i) \_ in \_ memory)(Channel \_ enable \mid r(N \mid N < N(max)) \rightarrow$$
$$\rightarrow (Channel \_ enable \mid r(T \mid T < T_z))WORK \_ CHANNEL$$
$$\cup$$
$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (C_i^k < C_{i-1}^k) \Leftrightarrow$$
$$r(a \mid a = 0) \rightarrow ((C_i^k - C_{i-1}^k) > \Delta n_{c^k}) C_{i-1}^k < C^k(min)(Channel \_ drop)$$
$$\cup$$
$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (C_i^k < C_{i-1}^k) \Leftrightarrow r(a \mid a = 0) \rightarrow$$
$$((C_i^k - C_{i-1}^k) > \Delta n_{c^k}) C_{i-1}^k > C^k(min) \rightarrow$$
$$\rightarrow (Channel \_ enable \mid r(N \mid N < N(max))(Channel \_ enable \mid r(T \mid T < T_z)) \Leftrightarrow$$
$$\Leftrightarrow WORK \_ CHANNEL$$
$$\cup$$
$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a = 0)$$
$$\rightarrow ((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i) \_ in \_ memory) \vee$$
$$\vee (Check\_C^k)(C_k^i - C_{k-1}^i) < \Delta n_{c^k}) C_{i-1}^k > C^k(min) \vee ((C_k^i - C_{k-1}^i) \_ in \_ memory) \rightarrow$$
$$\rightarrow (Channel \_ enable \mid r(N \mid N < N(max))(Channel \_ enable \mid r(T \mid T < T_z)) \Leftrightarrow$$
$$\Leftrightarrow WORK \_ CHANNEL$$
$$\cup$$
$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a = 0) \rightarrow$$
$$((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i) \_ in \_ memory) \vee$$
$$\vee (Check\_C^k)(C_k^i - C_{k-1}^i) < \Delta n_{c^k}) C_{i-1}^k > C^k(min) \vee ((C_k^i - C_{k-1}^i) \_ in \_ memory) \rightarrow$$
$$\rightarrow (Channel \_ enable \mid r(N \mid N < N(max))(Channel \_ enable \mid r(T \mid T > T_z))(Drop\_T) S_0 req.$$

There T_z – tags lifetime, C_i^k, C_{i-1}^k – bandwidth between two time intervals (*i. i-1*), $C^k(\mathbf{min})$ - minimum bandwidth in single channel, $N(\mathbf{max})$ – metric of network segment.

This approach shows liveness and boundedness properties of E-model. As can be seen from the proposed description, achieving a state *WORK CHANNEL* is possible only in a few situations and presented in formulas below:

$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a = 0) \rightarrow$$
$$\rightarrow ((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i)\_in\_memory) \rightarrow$$

$$\rightarrow (Check\_C^k) \vee ((C_k^i - C_{k-1}^i)\_in\_memory)(Channel\_enable \mid r(N \mid N < N(max)) \rightarrow$$
$$\rightarrow (Channel\_enable \mid r(T \mid T < T_z))WORK\_CHANNEL;$$

$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a = 0) \rightarrow$$
$$\rightarrow ((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i)\_in\_memory) \vee$$

$$\vee (Check\_C^k)(C_k^i - C_{k-1}^i) < \Delta n_{c^k}) C_{i-1}^k > C^k(min) \vee ((C_k^i - C_{k-1}^i)\_in\_memory) \rightarrow$$

$$\rightarrow (Channel\_enable \mid r(N \mid N < N(max))(Channel\_enable \mid r(T \mid T < T_z))WORK\_CHANNEL;$$

$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (C_i^k < C_{i-1}^k) \Leftrightarrow$$
$$r(a \mid a = 0) \rightarrow ((C_i^k - C_{i-1}^k) > \Delta n_{c^k}) C_{i-1}^k > C^k(min) \rightarrow$$
$$(Channel\_enable \mid r(N \mid N < N(max))(Channel\_enable \mid r(T \mid T < T_z))WORK\_CHANNEL.$$

The following formal chain (behavior chain)can detect deadlocks and unreachable state in our E-network. In this way we receive information about problem in reap protocol's work.

$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (\neg (C_k^i < C_{k-1}^i) Check\_C^k;$$
$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (C_i^k < C_{i-1}^k) \Leftrightarrow r(a \mid a = 0) \rightarrow ((C_i^k - C_{i-1}^k) > \Delta n_{c^k}) C_{i-1}^k < C^k(m^3 n)(Channel\_drop)$$

$$S_0 S_0 reg(S_k ans S_k) Check\_C^k (C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a = 0) \rightarrow$$
$$\rightarrow ((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i)\_in\_memory) \vee$$

$$\vee (Check\_C^k)(C_k^i - C_{k-1}^i) < \Delta n_{c^k}) C_{i-1}^k > C^k(min) \vee ((C_k^i - C_{k-1}^i)\_in\_memory) \rightarrow$$

$$\rightarrow (Channel\_enable \mid r(N \mid N < N(max))(Channel\_enable \mid r(T \mid T > T_z)) \cup$$
$$(Drop\_T) S_0 req \quad (6)$$

Formula (6) determines the potentially attain able state sand cycling opportunities in the model system, which can reduce the effectiveness of the functioning of the protocol and spend more time working on the identification of communication channels. Thus such approach gives ability to perform dynamic verification due to ability to check and intermediately change the single state of system's element.

7. CONCLUSION

With the great increasing of Web technology, more and more computation are established by Web services residing over the Internet. For accomplishing the goal of the computation, they should not only have "correct" functionalities, but also correct interactions with each other. With

the interaction becoming more complex, the problems related to specify and verify the interaction of the participants will become harder, too.

Thus, the processes in evolved in providing services can be classified into private and public that determines the interaction between system hosts.

To formally describe the requirements for Web-services developed a number of specification languages; the most used of these are BPML, BREL, WSDL, WSCI. However, questions remain open coordination composition and interaction Web-services, as well as their correct functioning.

Basic method to execute validation functioning Web-services is to verify. For verification of multiservice distributed network is proposed to use model-based approach. The paper proposes means of describing the composition and coordination of Web-services. Proposed approach allows to perform a formal association of individual system components into a single unit.

Proposed step by step verification method allows to perform a dynamic verification of the service by changing its structure or constituent elements, as well as repairing ortimeout.

REFERENCES

[1] The Standish Group, *The scope of software development project failures*: The Standish Group. Stanford. Available http://www.cs.nmt.edu/~cs328/reading/Standish.pdf, 2012.

[2] J. Westerman, "SOA today: Introduction to service-oriented architecture." Available http://www.information-management.com/news/7992-1.html.

[3] C. Peltz, "Web services orchestration and choreography," *IEEE Computer*, vol. 36, pp. 6-52, 2003.

[4] G. Alonso, F. Casati, H. Kuno, and Machiraju, *Web services—concepts, architectures and applications*. Berlin Heidelberg: Springer-Verlag, 2008.

[5] A. Airkin, S. Askary, and W. Fordin, "Web service choreography interface (WSCI) 1.0," *W3C Working Group*, 2002.

[6] C. Peltz, *Web services orchestration — A review of emerging technologies, tools, and standards*: Hewlett-Packard Company, 2003.

[7] S. Weerawarana and C. Francisco, "Business process with BPEL4WS: Understanding BPEL4WS," Part 1 // Research Report, IBM Developer Works2002.

[8] M. Y. Vardi, "An automata-theoretic approach to automatic program verification. M.Y. Vardi, P. Wolper," in *Proc. of the First Symposium on Logic in Computer Science*, 1986, pp. 322-331.

[9] A. Lafuente, "Directed search for the verification of communication protocols," A. Lafuente // Doctorial Thesis, University of Freiburg, Institute of Computer Science. pp: 157, 2009.

[10] C. S. Langdon, "The state of web services," *IEEE Computer*, vol. 36, pp. 96-99, 2010.

[11] M. Baldoni, C. Baroglio, and A. Martelli, "Verifying the conformance of web services to global interaction protocols: A first step," *International Workshop on Web Services and Formal Methods*, 2010.

[12] M. Bravetti, C. Guidi, and R. Lucchi, "Supporting e-commerce systems formalization with choreography languages," in *SAC '05: Proceedings of the 2005 ACM Symposium on Applied Computing*, New York, USA, 2005, pp. 831–835.

[13] J. Clarke, M. Edmund, and A. Peled, *Model checking*: MIT Press, 1999.

[14] A. Assaf, A. Sid, and B. Ben, "Web services business process execution language version 2.0, Editors," *OASIS Open*. Available http://www.oasis-open.org/committees/download.php/10347/wsbpelspecification-draft-120204.htm. [Accessed December 2004], 2004.

[15] W. Simon, "Specification and verification of composite web services," in *In Proocedings of the 8th Enterprise Distributed Object Computing Conference*, 2010.