Michalska Katarzyna
Wrocław University of Technology, Poland

Modelling of computer systems – an approach for functional and dependability analysis

Keywords

dependability, modelling, computer systems, service

Abstract

The paper presents a method of analyzing dependability aspects of service oriented information systems based on functional and dependability modelling. Analysis approach is based on integrating different computer system models into one coherent model suitable for simulation. To simplify integration of the models author propose an automatic solution that is integrated with the tools chosen for system analysis. Analysis is done with a usage of open-source simulation environment that can be easily modified and extended for farther work. Based on the simulation results, with respect to defined dependability and functionality metrics, some alternatives can be chosen in case of system or service failure.
The paper presents developed software which implements described methodology and results of analysis for an exemplary business service oriented complex information system.

1. Introduction

In the era of e-shopping, e-banking, e-learning and e-services, Internet accessibility cause growing numbers of users and their needs. Trends of service personalization increase these requirements. To satisfy these needs, various concepts are proposed. One of them is a Service Oriented Architecture (SOA) concept, but the complexity of these systems, as much as their requires of dependability and management issues still need an improvement. This paper focuses on the complex information systems, where business service aspects are crucial for their owner (service provider). Unavailability of these systems causes large financial consequences and loss (not only in a sense of economic, but also as loss of a good name of the brand).
In area of monitoring and still specified and in most cases commercialized. In fact these solutions (tools, models) are so specified for the implementation, that they could not be used easily for any other design or even slightly more complex one.
Moreover in the area of description and measurement dependability and functionality aspects are treated as separate categories but not as a hybrid method. For example, in case of analysis, metrics are used only for a business level of abstraction or system infrastructure level. Hybrid metrics (for both levels) are still under research [7, 25].
In this paper we based on functional and dependability approach related with tree main parts: modelling, analysis and synthesis (*Figure 1*).

2. Service oriented information systems

Computer Information System (CIS) is described [13] as a 4-tuple:

$$CIS =< Z,HS,M,K > \quad (1)$$

where:
Z – tasks, HS – technical infrastructure (hardware, software, links), M – clients, K – chronicle of the system (understood as time functions of the system).

Since we propose to analyze the Information System from a business service perspective called *Business Service Information System (CISB)*, we have extended eq. (1) and define *Business service* (*BS*) as a set of business logic, that can be loaded

and repeatedly used for concrete business handling process (i.e. ticketing service, banking, etc).

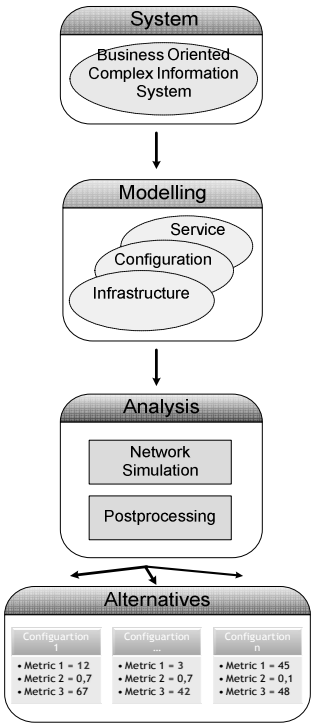

Figure 1. Research concept - overview

$$CISB =< Z, M, BS, HS, K > \quad (2)$$

Business service (BS) can be seen as a set of service components and tasks that are used to provide service in accordance with business logic for this process. Each business service business service components which consists of a set of activities that are the lowest observable entities level (requests and responses).

$$BS_i = \bigcup_j SC_j ; j \in N_{BS_i} \quad (3)$$

Service component (SC_i) is a service located on defined host (server) that determined service behavior, possibilities and requirements (i.e. authentication, data base service, web service, etc.). One host can have more than one service component.

$$SC_i = \bigcup_j T_j ; j \in N_{sc_i} \quad (4)$$

Technical infrastructure (HS) is considered as a set of hosts and computer network and is assumed to have the aspects of TCP/IP traffic are negligible. Each host is described by server name (unique ID), host performance parameter and a set of technical services (i.e. apache web server, MySQL database).

Chronicle of the system (*K)* are the time functions on each level of abstraction.

Clients (*M*) consists of a set of users where each user is defined by its allocation (host), number of concurrently ruing users of given type, set of activities (a sequence of task calls - name of task and a name of service component) and inter-activity delay time (modelled by a Gaussian distribution).

Taks (*Z*) are the input data specified by the clients in case of business service usage (i.e. selection of a service imply its components with a specified choreography).

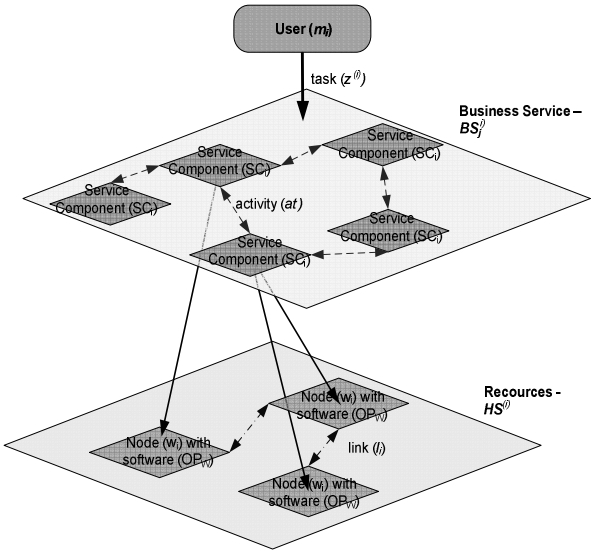

Figure 2. Business service oriented information system – levels of abstraction

3. Analysis of business service oriented information systems

There are various methods of system analysis. Some researches try to do it using graphs [9], others choose some simulation techniques. In this paper we consider system behaviour that will be as close to reality as is can be. Usage of simulator allow us to mimic the behaviour of a system. In literature, two main types of simulators can be found: a continuous time and discrete event based simulation [12], [22].
Continuous simulation requires a representation of the system using differential equations [9]. This type of simulator is predominately related with

electric power studies. For this reasons they will be excluded from further research. The other group of simulators are discrete event that describes the system behaviour as a series of events. Classically discrete event simulators are basis for telecommunication and IT analysis tools. The set of the most popular simulators of this kind is as follows: OPNET [5,18] and NS-2 [14], both well known by stakeholders, as well as QualNet [21], OMNeT++ [16], SSFNet/PRIME SSF [23], and SGOOSE [9].

Experiments reported in this paper were performed using the SSFNet simulation environment developed by the Renesys Corporation with support from DARPA. SSFNet has large number of protocols models and network elements; moreover open-source code allows modification. In this paper Java based version of SSFNet were used since Java language allowed much faster development then a usage of C++.

SSFNet simulator consists of three major parts: SSF engine, DML language [8] and SSFNet models. The SSF (Scalable Simulation Framework) is public-domain standard for discrete-event simulation implemented in C++ with Java and C++ interface. Scalable Simulation Framework is a base for higher level - the SSFNet. SSFNet module is a collection of Java packages for modelling and simulation of networks and Internet protocols. Moreover SSFNet uses public-domain standard called DML (Domain Modelling Language) to configure simulation scenarios.

For the purpose of this work some extensions were developed, mainly connected with support for traffic generation (models of user behaviour), simulation of business level services, implementation of resource consumption for requests processing. Since fault and failures models are integral part of dependability analysis the SSFNet was extended to in-corporate errors. Errors were introduced in different levels beginning from link failures, network adapter failures to software component failures [23]. Additional modules of the tool required the extension of its input language (DML) used in standard SSFNet version, but the most important extension was implementing Monte-Carlo approach [10] based on running simulation several times and calculating results based on averages values. In this way during each simulation the parameters described in by stochastic processes, in our cases it was the traffic generation which modelled user behaviour in a random way, have different values (according to set-up distributions) having an influence on the system behaviour. The capability of multiple runs of simulation was added to standard SSFNet package by changes in several SSFNet classes (setting up random seed and clearing all static collections). Results of simulation are recorded in specified output file that allows further post- processing in case of dependability metrics.

For the needs of this research, we provided two metrics of information system dependability, i.e. availability and response time. Due to a randomness of a user behaviour the calculation of these metrics was done based on Monte-Carlo approach by repeating simulation of the same system N times over analyzed period T. Therefore, all defined below metrics are calculated as an average over all batches of simulation.

The availability function [2] *A(t)* for the system is a probability that system is working properly in time *t:*

$$A(t) = P\{\text{system is working in time } t\}$$

In Business Service Oriented Complex Information Systems with have more than one level of abstraction, we can suppose, that the system is available as a probability that in time *t* all requests came from users to the system and services are supported correctly. On this basis we can estimate that, the business service availability (*BSA*) can be computed on the basis of observed system uptime in the analyzed period *T* over *N* simulation as:

$$BSA = \frac{1}{NT}\sum_{i=1}^{N} t_{up}^{i}$$

whereas t_{up}^{i} is a time of service being working in *i*-th simulation.

Above formula requires defining what does it mean that service is working. Since we are looking on the system form the client perspective, we assume that service is working if and only if it responds to the client with a proper response. The downtime starts when for some request there no proper response (the time of starting of response is used). It finished when for any request there is a proper answer (also a request send time is used in this case) [25].

In a very similar way we can calculate availability of the server (*SA*) as:

$$SA = \frac{1}{NT}\sum_{i=1}^{N} t_{up}^{i}$$

In this case we can calculate another level of abstraction in CISB, that is hardware one. In this case, t_{up}^{i} is a time of server being working in *i*-th simulation.

Response time in a CISB is a time to pass between request to the system and correct response. In case

of CISB it depends from network recourses, infrastructure capabilities and system overload.
Proposed metric analysis the business service response time (BSRT) and is intended to be a numerical representation of client's perception of particular service components quality. It is calculated for each tasks separately as an average delay between the starting time of user response ($t_{i\_request}$) and getting answer ($t_{i\_response}$) from the service (i.e. only requests that were properly answered are taken into account).

$$BSRT = \frac{1}{N_{request}} \sum_{i=1}^{N_{request}} (t_{i\_request} - t_{i\_response})$$

It is important to state that it is calculated for all properly answered requests ($N_{requests}$) in all simulation runs. BSRT is considered to be very useful for ranking system configurations.

4. Modelling of business service oriented information systems

Necessity of standardize languages that are (and will be used in a future) for describing various aspect of computer systems, was one of the reasons for creating organizations involved in this matter. These are:

- OASIS (ang. *Organization for the Advancement of Structured Information Standards*) [17],
- BPMI (ang. *Business Process Management Initiative*) [3],
- OMG (ang. *Object Management Group*) [15],
- W3C (ang. *World Wide Web Consortium*) [28],
- WfMC (ang. *Workflow Management Coalition*) [29],
- OAGi (ang. *Open Applications Group*) [14].

Criteria	Service description languages					System description languages		
	WSCI	WSCL	UML2	BPMN	WS-CDL	SDL	CIM	DML
Easy to learn	+/-	-	+	+	-	+	-	+
Extendable syntax	-	-	+	+	+	+	+	+/-
XML format	+	+	-	-	+	+	-	-
Multilevel Description	-	-	+	-	+	+	+	+
Stable version	-	-	+	+	+/-	+	+	+
Open source tools	-	-	+	+	+	-/+	-	+/-
Open source	+	+	+	+	+	+	+	+

Table 1. Language comparison − exemplary criteria

Taking into consideration two main aspect of the system under test, we have examined many languages and models from technical and nontechnical factors (*Table 1*). Based on eq. (2), we have considered formats like [7]:

- WSCI (*Web Service Choreography Interface*) [30],
- WSCL (*Hewlett-Packard's Web Service Choreography Language*) [31],
- UML2 (*The Unified Modelling Language 2: Activity Diagram*) [26],
- BPMN (*Business Process Modelling Notation*) [4],
- WS-CDL (*Web Services Choreography Description Language*) [11],
- SDL (*System Description Language*) [1, 7],
- CIM (*Common Information Model*) [6],
- DML (*Domain Modelling Language*) [8].

Since XML format was one of the crucial aspect of our the choice, we focus mainly on this category. As it can be seen in *Table 1*, there few more advantages of using languages under. In our research two of them were chosen: WS-CDL and SDL.
System Description Language (SDL) has been defined by the POSITIF Project [1] to provide a formal description of networked ICT systems. This language consists of a description in case of:

- system topology − physical (hardware) and logical (software) infrastructure;
- the network configuration and black-box functionality of each node;
- security policy of each node.

SDL meta model can be examined in two parts:

- *core* − description of various network elements, their connections and software (i.e. network devices, links, interfaces, logical elements, etc.);
- *extensions* − description aimed to information not related to physical network infrastructure (i.e. building, floor, room, alarm, etc.).

```
<computer id="WebServer">
        <interface connector="RJ45"
    datalinkprotocol="Ethernet CSMA/CD"
    id="eth0`" protocol="Unknown"
    technology="Ethernet">
            <address
            netmask="255.255.255.0"
            type="ipv4"
            value="192.168.1.1"/>
        </interface>
        <service id="web_server">
            <sap addr="192.168.1.1"
            port="80"  transport="tcp"/>
            <software idRef="apache"/>
        </service>
        <os idRef="Fedora" running="yes"/>
        <sw idRef="tomcat"
running="false"/>
</computer>
```

Figure 3. Exemplary description of a host in SDL

Creation and formal verification of models can be done using tool called SDLDesigner [1].

Web Services Choreography Description Language (WS-CDL) has been defined by W3C group. It is used to define behaviour of the services inside ICT network. WS-CDL focuses on describing the business protocol among different roles. Same as in case of SDL language, WS-CDL model can be divided in two parts:

- static − description of static relation i.e. communication types, channel types, parameters types;
- dynamic − description of cooperation between communication actors and their time correlations.

Important issues of WS-CDL are discussed in [11]. Creation and formal verification of models can be done using Pi4SOA tool [20].

```
    <interaction  channelVariable="tns:USR2DNS"
name="ZadanieAdresuIP" operation="GET">
        <participate fromRoleTypeRef="tns: USR "
relationshipType="tns:getUserIP"
toRoleTypeRef="tns:DNS"/>
        <exchange action="request"
informationType="tns:Opaque" name="request">
            <send
variable="cdl:getVariable('Response','','')"/>
            <receive
variable="cdl:getVariable('Request','','')"/>
        </exchange>
        <exchange action="respond"
informationType="tns:Opaque" name="response">
            <send
variable="cdl:getVariable('Response','','')"/>
            <receive
variable="cdl:getVariable('Request','','')"/>
    </exchange>
</interaction>
```

Figure 4. Exemplary description of activity using WS-CDL

Unfortunately existing languages (although still appearing as new solutions) are not fully sufficient for whole system description, but since XML format allows us to create new schemas and documents we can define our own elements and languages. In this paper author define simple language for mapping WS-CDL components into SDL elements (as pairs). Since every system can be described by more then one configuration, therefore using Operational Configuration Language (OCL) we can model various simulation configuration scenarios using one simple language.

```
<node type="host" name="WebSerwer"
                    network_name="DMZ1">
        <componet_name name="INTRANET">
```

Figure 5. Exemplary mapping description in OCL

Since SDL language give us only host description of
a client, we proposed XML based language that contain reference to SDL model with extensions concerning SSFNet client model requirements. For this reasons Workload Description Language (WDL) has been defined. It is used to provide user description for analysis proposes. It mainly focused on the simulation load - client number, size of their request, description of their behavior in case of using business service (activities), etc. If WDL language is not given, user is obligate to put this data in a framework for model integration (section 5). Please note that WDL is based on XML format, therefore creation and formal verification of WDL models can be done using any XML editing tools with respect to proposed XML Schema.

```
<Client clientID=" client2">
    <ClientDescription>malicious client
    (Bob)</ClientDescription>
    <ClientAction actionType="attack attempt"
    actionName="request main page">
            <PatienceTime>20</PatienceTime>
    ….
    </ClientAction>
</Client>
```

Figure 6. Exemplary description of client description using WDL

Figure 7 shows a concept of proposed language usage with respect to system levels (eq. (2,3,4)).

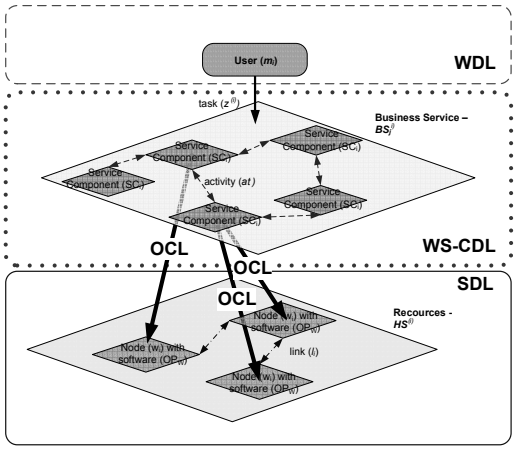

Figure 7. Usage of modelling languages for system description

5. Integrated environment

Usage of three modelling languages and an input language used in SSFNet simulator (Domain Modelling Language) cause a need of integration. In spite of the fact, that DML is simple text format it is difficult to read and write larger number of network nodes for simulation. Moreover extensions done in SSFNet entailed DML modification and SSFNet output format. Since DML text format is easy to transform to XML format, we propose an extended simulation output – called XDML. Creation of XDML languages give as many processing possibilities. First of all (as show on *Figure 8*) we can translate any language to an input format of this analysis module. In our case we can pick information from SDL, WS-CDL, OCL and WDL to create an XDML with a very small user interaction. Secondly an XML format is easily processed by Java and XML techniques (i.e. XSLT, DOM, SAX, JAXB technique) which is helpful in creating model (in our case Information System) visualization: showing the structure of the network and it's element.

For this reasons special tool called *Integrated Analysis Environment (IAE)* was implemented (*Figure 9*).

Each network element has several functional parameters and user can graphically edit this information. It is worth to mention that every SDL device type is translated into suitable XDML one and every WS-CDL service description is interpreted and translated into service components and tasks. Moreover in proposed framework user is able to put its own variables and attributes based on XDML specification or use extend models (i.e. consumption model, operational configuration model) to simplify its work.

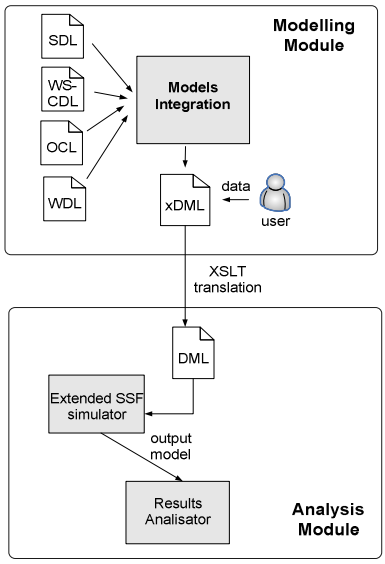

Figure 8. Integrated Analysis Environment – framework concept

Since analysis model is mainly based on Java interface of SSFNet, it helps to integrate SSFNet environment with created postprocessing module.

It allows adding some additional features concerning Monte-Carlo simulation (i.e. progress bar) and a module responsible for plotting calculated metrics.

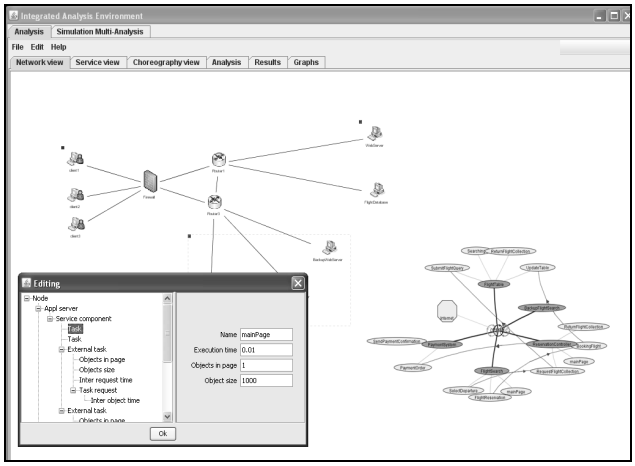

Figure 9. Integrated Analysis Environment – framework overview

5. Postprocessing results - test case scenario

For the case study analysis we propose an exemplar service system illustrated on *Figure 10*.

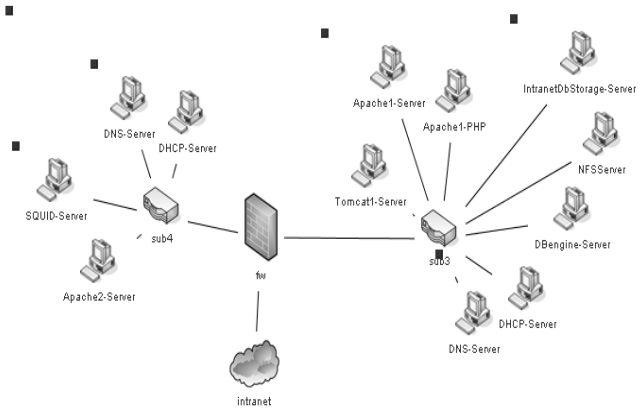

Figure 10. Testbed overview

The system is composed of tree networks: one is a client network, other service provider networks. For service realization system contains few servers i.e.: *Appache1-Server*, *DNS-Server* and *DBengine-Server*. Essentially the testbed system implements two main service scenarios: "*GetMainPage*" and "*LookNewsList*" that can interact with each other as shown on *Figure 10*. Each scenario is described using specified set of service component and interaction between them. In both scenarios user ask for a Wab Page, what is connected with request of DNS address and response from DNS component (located on a host DNS-Server, see *Figure 11*). In a second scenario "*LookNewList*" user after receiving DNS address request for some data from using *DBComponent* located on *DBStorage-Server*.

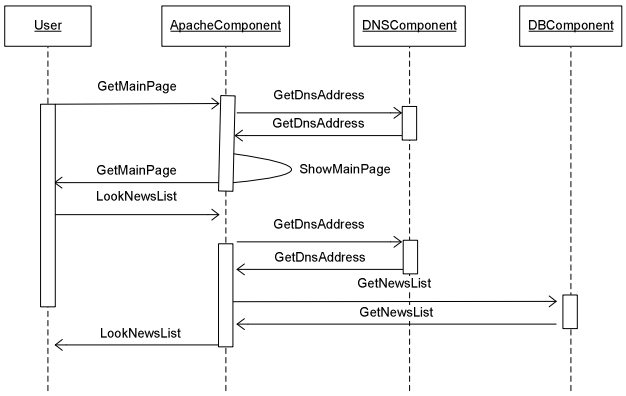

Figure 11. Service state chart

Since simulation allows to observe different parameters of observed model we focused mainly on a metrics proposed in section 3. In order to perform some interesting experiments we consider two hypothetic situations that may accrue in our testbed system: first, that all elements are working properly, second some failures were introduced (the failure of *DBengine-Server* staring at 1 000 s and finished at 5 000 s.).

The results of a comparison of the two configurations (without failures and failed) are given in *Figure 12-14*. The simulations was performed for simulation time of 10 000 seconds and repeated 100 times. As it could be expected, it shows that in case of failures performance and availability drops down for every level of abstraction.

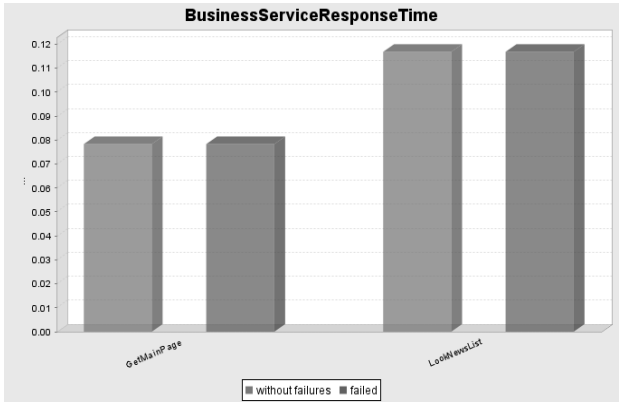

Figure 12. Business Service Response Time - metric results

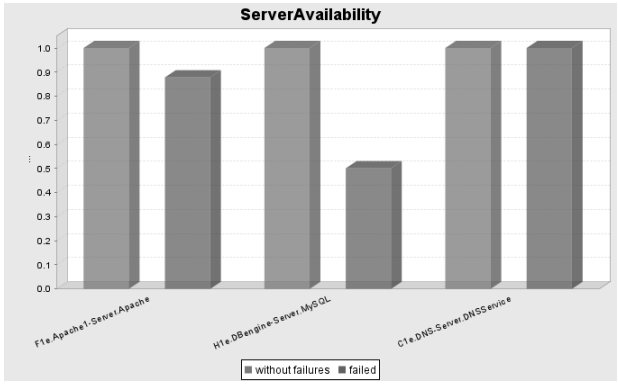

Figure 13. Service Availability - metric results

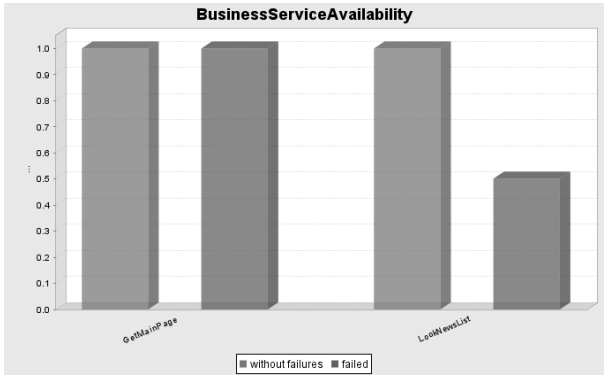

Figure 14. Business Service Availability - metric results

5. Conclusion

As described in this paper usage of a standard languages for model description (i.e. WS-CDL and SDL) and graphical interface (IAE) integrating whole modelling and analysis process makes the tool useful for Business Service Information System designers and administrators. Proposed method and tool could be used as powerful tool in order to examine any kind of system configuration without building it. Moreover proposed environment can be simply modify and extended in case of new dependability (especially availability) metrics, as much as additional languages that could improve effectiveness of system analysis. Based on proposed analysis we can help system administrator to reconfigure the system when needed without a significant lost of service performance for the user. In result it will reduce cost of building the technical infrastructure.

Work on extending a set of dependability models (i.e. improvement of policy model, failure model and workload) is currently ongoing.

References

[1] Aime, M. & Atzeni, A. & Pomi, P. (2007). *AMBRA* - Automated Model-Based Risk Analysis. The 3rd Interna-tional Workshop on Quality of Protection (QoP 2007) (Affiliated with ACM CCS) Alexandria, VA, USA.

[2] Avižienis, A. & Laprie, J-C. & Randell, B. (2000). Fundamental Concepts of Dependability. 3rd Information Survivability Workshop (ISW-2000), Boston, Massachusetts, USA.

[3] Business Process Management Initiative; *http://www.bpmi.org/*

[4] BPMN (*Business Process Modelling Notation*); *http://www.bpmn.org/*

[5] Chang, X. (1999). Network simulations with OPNET, *Proc. Simulation Conference*, Vol. 1, 307-314.

[6] CIM (*Common Information Model*); *http://dmtf.org/standards/cim*

[7] DESEREC Project Home Page, *www.deserec.eu*

[8] Domain Modeling Language (DML) reference Home Page, *http://www.ssfnet.org/SSFdocs/dmlReference.html*

[9] Duflos, S. & Diallo, A.A. & Le Grand, G. (2007). An Overlay Simulator for Interdependent Critical Information Infrastructures. Dependability of Computer Systems, DepCoS-RELCOMEX '07, IEEE Press, 27-34.

[10] Fishman, G. (1996). *Monte Carlo: Concepts, Algorithms, and Applications*. Springer-Verlag, New York

[11] Hongli, Y. & Xiangpeng, Z. & Zongyan, Q., Geguang, P. & Shuling, W. (2006). A formal Model for Web Service Choreography Description Language (WS-CDL). *Proc. of ICWS 2006. IEEE Computer Society*.

[12] Nicol, D. & Liu, J., Liljenstam, M. & Guanhua, Y. (2003). Simulation of large scale networks using SSF. *Proc. of the 2003 Winter*, Vol. 1, 650 – 657.

[13] Michalska, K. & Walkowiak, T. (2008). Hierarchical approach to dependability analysis of information systems by modeling and simulation. The First International Workshop on Dependability and Security in Complex and Critical Information Systems, DEPEND 2008, August 25-31, 2008 - Cap Esterel, France, *IEEE Computer Society Press* 2008.

[14] NS-2 Simulator Home Page, *www.isi.edu.nsnam/nas/*

[15] Object Management Group; http://www.omg.org/

[16] OMNeT++ Simulator Home Page, *www.omnetpp.org*

[17] Organization for the Advancement of Structured Information Standards; *http://www.oasis-open.org/*

[18] OPNet Modeler Home Page, *http://www.opnet.com/*

[19] Open Applications Group; *http://www.openapplications.org/*

[20] PI4SOA project web page: *http://sourceforge.net/projects/pi4soa*

[21] QualNet Simulator Home Page, *www.scaleble-networks.com*

[22] Riley, G.F. (2003). Large-scale network simulations with GTNetS. Proc. of the 2003 Winter, Vol. 1, 676 – 684.

[23] SSFNet Simulator Home Page, *www.ssfnet.org*

[24] Walkowiak, T. & Michalska, K. (2010). Performance analysis of service-based information system with load balancer - simulation approach. *Dependability of networks*. 155-168.

[25] Michalska, K. & Walkowiak, T. (2009). *Simulation approach to performance analysis information systems with load balancer*. Information systems architecture and technology: advances in Web-Age Information Systems. 269-278.

[26] UML2 (*The Unified Modelling Language 2: Activity Diagram*); *http://www.uml.org/*

[27] Wang, S. & Liu, K.Z. & Hu, F.P. (2005). Simulation of Wireless Sensor Networks

Localization with OMNeT. *Mobile Technology, Applications and Systems*, 1 – 6.
[28] World Wide Web Consortium*; http://www.w3.org/*
[29] Workflow Management Coalition; *http://www.wfmc.org/*
[30] WSCI (*Web Service Choreography Interface*); *http://www.w3.org/TR/wsci/*
[31] WSCL (*Hewlett-Packard's Web Service Choreography Language*); *http://www.w3.org/TR/wscl10/*