Proceedings of the 11th International Conference "Reliability and Statistics in Transportation and Communication" (RelStat'11), 19–22 October 2011, Riga, Latvia, p. 382-389. ISBN 978-9984-818-46-7
Transport and Telecommunication Institute, Lomonosova 1, LV-1019, Riga, Latvia

RELIABILITY AND FUNCTIONAL ANALYSIS OF WEB SYSTEMS BY MONTE-CARLO SIMULATION

Tomasz Walkowiak

Institute of Computer Engineering, Control and Robotics
Wroclaw University of Technology
ul. Janiszewskiego 11/17, 50-372 Wroclaw, Poland
E-mails: Tomasz.Walkowiak@pwr.wroc.pl

The paper describes a reliability and functional analysis of Web systems. The analysed systems are modelled as a set of tasks that use data, obtained in an interaction with other tasks, to produce responses. System reliability is described by a failure and repair process of system elements. Whereas repair time model takes into account working hours and weekends of maintain crew. The reliability is measured by a system availability calculated by a Monte-Carlo based simulator. Next, functional analysis of a web system is measured by a functional availability, i.e. the probability that a client will receive response within a given time limit. The metric is calculated by simulation software developed by authors. The web system simulation takes into account the consumption of computational resources (host processing power). Finally, the service availability is defined, which compromises the reliability and functional parameters as well as an input load of the system. Simulation results of for a testbed system are given.

Keywords: web system, reliability, availability, Monte-Carlo simulation

1. Introduction

The Web systems are currently becoming the core infrastructure of almost all business activities. They belong to a class of complex systems as a result of the large number of components and their complicated interactions. As more and more web systems are being designed and implemented it's vital to have means for analysis the system reliability and ways of selecting the best (according to some criteria) configuration of the system components and system maintenance.

Reliability is mostly understood as the ability of a system to perform its required functions for a specified period of time [1]. It is mostly defined as a probability that a system will perform its function during a given period of time.

The classical web system reliability analysis [5] is based on Markov or Semi-Markov processes [1]. The typical structures with reliability focused analysis are not complicated and are based on serial-parallel or k of n systems. It is mostly assumed that a system is stationary and therefore one could calculate stationary reliability as the asymptotic value of reliability. Therefore, the method is based on a definition of system operational states. Next, the calculation of the probability that the system is being in a given state is performed. Assessing the reliability states as operational or failed one could calculate the reliability as the expected value of the system being in operational states. The main drawback of the classical approach is the fact that it is based on a very strict assumptions related to the life or repair time and random variables. The assumed distributions of the analysed system elements are idealized and it is hard to reconcile them with practice. Such reliability approach is not able to support the time-depending analysis, which seems to be very important in some practical applications. Therefore, we propose to weaken the assumptions of exponential distribution of repair time and take into account more practical situations like working hours and weekends. Moreover, we focus on a business service realized by web-system [4] and functional aspects of the system, i.e. performance aspects of business service realized by a web system. We assume that the main goal, taken into consideration during design and operation of the web-system, is to fulfil the user's requirements, which could be seen as some requirements to perform a user's tasks within a given time limit.

To deal with reliability and functional aspects of web systems we propose a common approach [2] based on modelling and simulation. Modelling is focussed on a process of execution of a user request, understand as a sequence of task realised on technical services provided by the system [9], whereas, the simulation is responsible for reliability and functional analysis. It is based on a time event simulation with Monte Carlo analysis [3].

The organisation of the paper is as follows. We start with modelling of the web system on the task level (section 2). Next, reliability analysis of a web system is given section 3. It is followed by a functional analysis performed by a simulator tool. The tool allows calculating the user request time and therefore to calculate the functional availability. Finally, the client availability is defined by compromising reliability

(failures and repairs), functional aspects (probability that a user will receive a request within a given time limit for a given system input load) and input load (the number of users changing in time during a week). We conclude with a short summary and plans for future works.

2. Web System Model

As it was mentioned in the introduction we decided to analyse web systems from the business service point of view. Therefore, we model a process of execution of a user request, understand as a sequence of task realised on technical services provided by the system.

Generally speaking users of the system are generating tasks which are being realized by a web system. The task to be realized requires some services presented in the system. A realization of the system service needs a defined set of technical resources. Moreover, the services have to be allocated on a given host. Therefore, we can model a web system (*WS*) as a 4-tuple [8, 9]:

$$WS = \langle Client, BS, TI, Conf \rangle \tag{1}$$

Client – finite set of clients,
BS – business service, a finite set of service components,
TS – technical infrastructure,
Conf – information system configuration.

During modelling of the technical infrastructure we have to take into consideration functional and reliability aspects of web systems. Therefore, the technical infrastructure of the web system could be modelled as a pair:

$$TI = \langle H, N \rangle, \tag{2}$$

where
H – set of hosts (computers); N – computer network.

We assume that aspects of TCP/IP traffic are negligible and therefore we model the network communication as a random delay [9]. Therefore, the N is a function, which gives a value of time of sending a packet form one host (v_i) to another (v_i). And it is given by a Gaussian distribution with a standard deviation equal to 10% of the mean value.

The main technical infrastructure of the web systems are hosts. Each host is described by its reliability parameter (mean time to failure and mean time of repair) and functional parameters:

- server name (unique in the system),
- host performance parameter – the real value which is a base for calculating the task processing time (described later),
- set of technical services (i.e. apache web server, tomcat, MySQL database), each technical service is described by a name and a limit of tasks concurrently being executed.

The *BS* is a set of services based on business logic, that can be loaded and repeatedly used for concrete business handling process (i.e. ticketing service, banking, VoIP, etc.). Business service 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 [11]. Therefore, *BS* is modelled a set of business service components (*BSC*), (i.e. authentication, data base service, web service, etc.), where each business service component is described a name, reference to a technical service and host describing allocation of business service component on the technical infrastructure and a set of tasks. Tasks are the lowest level observable entities in the modelled system. It can be seen as a request and response form one service component to another. Each task is described by its name, task processing time parameter and optionally by a sequence of task calls. Each task call is defined by a name of business service component and task name within this business service component and time-out parameter. System configuration (*Conf*) is a function that gives the assignments of each service components to a technical service and therefore to hosts since a technical set is placed on a given host. In case of service component assigned in a configuration to a load balancing technical service the tasks included in a given service component are being realised on one of technical services (and therefore hosts) defined in the load balancer configuration. The client model (*Client*) consists of a set of users where each user is defined by its allocation (host name), replicate parameter (number of concurrently ruing users of given type), set of activities (name and a sequence of task calls) and inter-activity delay time (modelled by a Gaussian distribution).

Summarising, a user initiate the communication requesting some tasks on a host, it could require a request to another host or hosts, after the task execution hosts responds to requesting server, and finally the user receives the respond. Requests and responds of each task give a sequence of a user task execution (choreography) as presented on exemplar Figure 1.

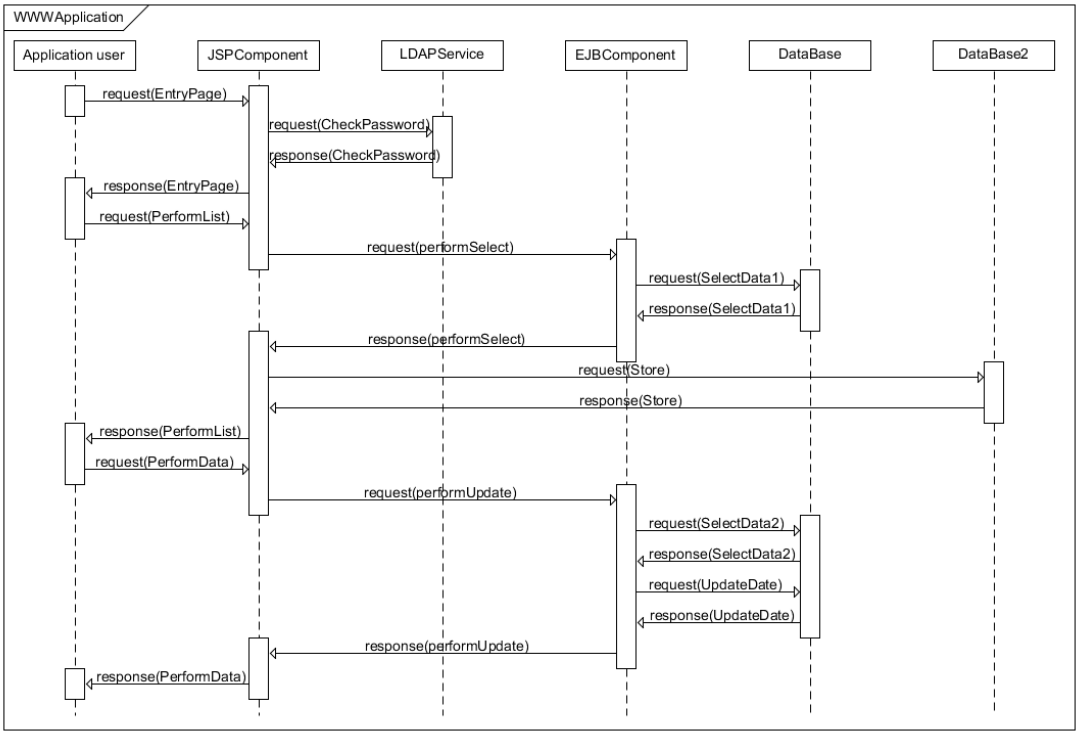

Figure 1. Testbed choreography

The request is understood as correctly answered if answers for each requests in a sequence of a user task execution were given within defined time limit (parameter of each request in *BS* model) and if a number of tasks executed on a given technical service is not exceeding the limit parameter (parameter of *TI* model).

The user request execution time in the system is calculated as a sum of times required for TCP/IP communication (modelled by a random value) and times of tasks processing on a given host. The task processing time is equal to the task processing time parameter multiplied by a number of other task processed on the same host in the same time and divided by a the host performance parameter. Since the number of tasks is changing in simulation time, the processing time is updated each time a task finish the execution or a new task is starting to be processed.

3. Reliability Analysis of Web System

3.1. Reliability model

There are numerous sources of faults in Web systems. These encompass hardware malfunctions (transient and persistent), software bugs, human mistakes, viruses, exploitation of software vulnerabilities, malware proliferation, drainage type attacks on system and its infrastructure (such as ping flooding, DDOS). We propose to model all of them from the point of view of a resulting failure. Looking at the model presented in the previous section it means that a host is not working. If a host is taking part in the choreography (a given business service component is placed on this host) the whole business service is not responding, therefore the web system is not in a ready state.

We assume that system failures could be modelled by a set of failures. Each failure is assigned to one of hosts and represents a separate and independent working-failure stochastic process. The occurrence of failures is described by a random process. The time to failure is modelled by the exponential distribution like in the Markov approach. But in case of repair time a more sophisticated assumption are taken into consideration.

As it has been mentioned in the introduction the classical reliability analysis is assuming the stationary of the process. However, the experience of the author shows that the most probable day that there is a need to repair a web system is Monday. It is mainly due to a fact that during weekends nobody is maintaining the system but failures could happen. To consider this fact we propose to model in detail maintaining crew working hours.

Therefore, we assume that working hours of system maintainer are from 8 am to 4 pm, Monday to Friday. And only during these days any failure could be discovered and any repair operation could be performed. Of course any other system of working hours (like for example two shifts) could be analysed in a similar way. Moreover, we assume that new elements (to replace failed one) needs to be delivered from external localization by a post courier. Therefore, the element to be replaced will be available on the next working day at 8 am.

Taking into account all these assumption we could distinguish following elements of repair time:

- failure discover time – equal to zero if a failure happens during working hour, in other cases it is equal to a time period to a beginning of next working day (8 am);
- failure analysis time – a time needed to discover the failure reason, it is model by a random variable with a truncated normal distribution, if the failure analysis will be overlapping with not working hours, it is enlarge by a time period to next working day (by 16 hours or even 66 hours in case of Friday);
- element delivery time – with a given probability it is equal to zero (it allows to model an operating system or software component failure), in other cases equal to a time period to next working day (modelling the delivery of the element be a courier);
- element replacement – a time required to replace a failed element or a software reinstallation/ repair in case of software failure, modelled by a random variable with a truncated normal distribution within working hours (like in failure analysis time).

3.2. System availability

We propose to measure reliability aspects of web systems by system availability metric. It is defined as the probability that the system is ready (provides responses) at a specific time:

$$A_S(t) = \Pr(ready(t)) \ . \tag{3}$$

Model described in the previous section gives a periodic availability function with a period equal to one week. The model was a base for a development of a reliability simulator. It was implemented in C++ within the Scalable Simulation Framework (SSF)[6]. SSF is an object-oriented API – a collection of class interfaces with prototype implementations. For the purpose of simulating reliability sates we have used Parallel Real-time Immersive Modelling Environment (PRIME) [7] implementation of SSF due to much better documentation then available for original SSF and additional methods of synchronization between processes. The main reason of using SSF was the fact that it was also used by authors for a development of functional simulator of Web system (section 4.2).

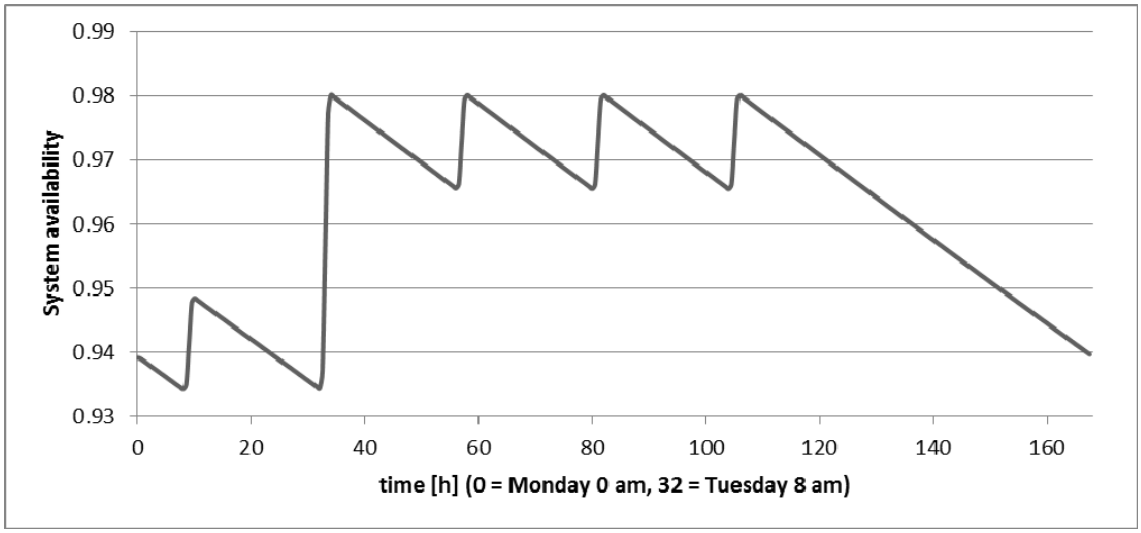

Figure 2. System availability for a testbed system

Simulation experiments were performed on a testbed web system consisting of six hosts. There were five computers with business components communicating according to the choreography presented on Figure 1 and one router with a firewall. Since, today's computer devices are not failing very often, the intensity of failures was set to one year per year. Whereas repair parameters were set as follows:

- mean failure analysis time – 3 hours,
- mean element replacement time – 1 hour,
- probability of hardware failure – 0.9.

The achieved results for simulation of simulating 10,000 weeks 10,000 times are presented on Figure 2.

4. Functional analysis of web system

4.1. Functional simulation [9, 10]

As it was mentioned in the introduction the presented approach is focused on a process of execution of a user request, understand as a sequence of task realised on technical services provided by the system.

Therefore, the main aim of simulation was to allow calculation service response time. The user initiate the communication requesting some tasks on a host, it could require a request to another host or hosts, after the task execution a host responds to requesting server, and finally the user receives the respond. Requests and responds of each task give a sequence of a user task execution – choreography (see Figure 1).

The user request execution time in the system is calculated as a sum of times required for TCP/IP communication (modelled by a random value) and task processing times on hosts according to some choreography.

The task processing time calculation method has to take into consideration the fact that the number of tasks running on a given host changes in time. Let $\tau_1, \tau_2, ..., \tau_e$ be a time moments when a task ($task_j$) with some execution time parameter ($et(t_j^i)$) is starting or finishing processing on a host on which the task is allocated ($h_i = al(task_j)$). With each host, a performance parameter is associated (($per(h_i)$)). It is a real, relative value representing a host processing power. It is equal to one for some reference host. Let $num(h, \tau)$ denotes a number of task being processed at time τ on host *h*. It is not taking into account tasks which requests tasks on other hosts and waits for a response. Therefore, τ_e – the time when task $task_j$ finishes its execution – has to fulfil a following rule:

$$\sum_{k=2}^{e} (\tau_k - \tau_{k-1}) \frac{per(h_i)}{num(h_i, \tau_{k-1})} = et(task_j). \tag{4}$$

Having above equation the task processing time is equal to:

$$pt(task_j^i) = \tau_e - \tau_1. \tag{5}$$

Therefore, service response time for a single user performing some choreography *c* (consisting of a sequence of *n*-tasks $c = < task_1, task_2, ..., task_n >$) could be calculated as:

$$urpt(c) = \sum_{i=1}^{n} \left(delay(al(task_{i-1}), al(task_i)) + pt(task) + delay(al(task_i), al(task_{i-1})) \right), \tag{6}$$

where

- $delay(h_i, h_j)$ is a time of task request transmission from host h_i to host h_j. It is modelled by a random value with Gaussian distribution.
- $al(task_0)$ is understood a location of a user.

It is important to mention that *urpt()* that could be seen as a random value since it has different values for each user interaction with the system.

As it was mentioned in the previous section we have developed a web system simulator [9, 10, 11] within a SSF framework. The main advantages of SSF in case of simulating Web systems is its process oriented approach. The process in SSF could be seen as independent threads of control. It can be blocked waiting for an event to arrive or for a given period of simulation time. Since the real Web servers are built as multithreaded applications processes available in SSF net simplifies Web server implementation in the simulator. Moreover, the implementation of processes in SSF is very efficient, much faster than a usage of general purpose multithreading library. A small test done by authors comparing Java threads to SSF processes showed that process context switching (which happen frequently in event simulation – at least once per one event) in SSF is more than 10 times shorter.

4.2. Functional availability

To measure functional aspects of web system we propose to use the client availability metric. It is defined as a probability that service response time will be less than a given time limit ($t_{\max}$) for a given load (number of user requests per second) of the system:

$$Af(n) = \Pr(urpt(c) \le t_{\max} \mid nuser = n) \mid \quad (7)$$

This metric is a numerical representation of clients' perception of particular business service quality. It measures the probability that the user will not resign from active interaction with the service due too long service response time.

The example results for the testbed system with choreography from Figure 1 with a time-limit set to 10 s and concurrent task limit set to 200 are presented on Figure 3. The mentioned before time-outs and a maximum number of tasks results in dropping some of requests and therefore in decrease of the availability parameter when number of user requests enlarges.

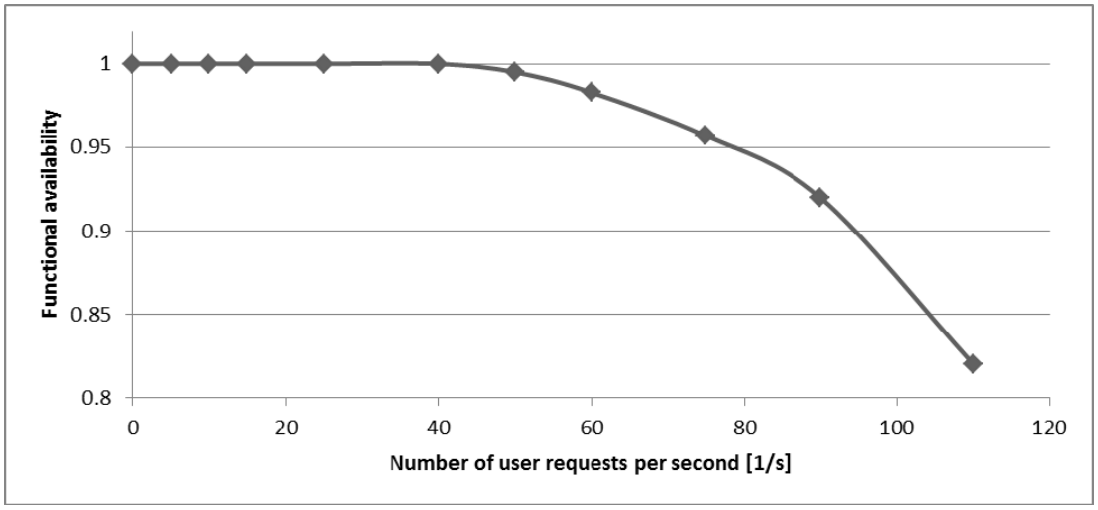

Figure. 3. Functional availability for a testbed system in a function of number of user requests per second

5. Reliability and Functional Analysis of Web System

5.1. Input load and client availability

It is obvious that a number of users of a web system changes in time. It depends on a type of provided service. But usually on weekends, there are fewer users then on weekdays. Also one could notice changes in traffic during a day – with a peak load in a middle of day and minimum load at night. Therefore, the input load variations could be described by a function *nuser*(*t*) similar to that presented on Figure 4.

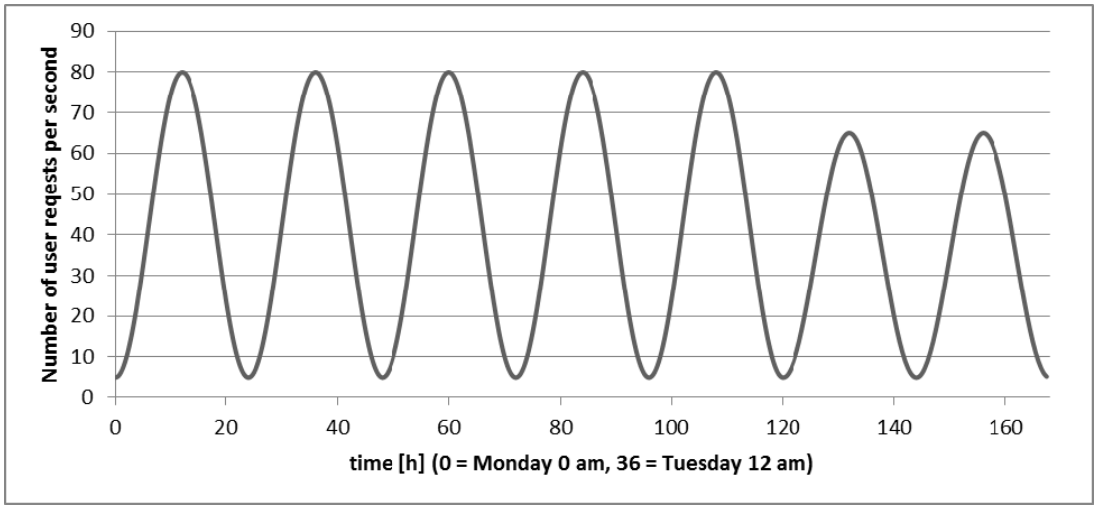

Figure 4. Input load for a tesbed system

Using the function *nuser*(*t*) the changes of functional availability in time could be easily calculated as

$$A_{ft}(t) = \Pr(urpt(c) \le t_{\max} \mid t) \models A_u\left(nuser(t)\right). \quad (8)$$

The results for a testbed system and assumed input load (form Figure 4) are presented on Figure 5.

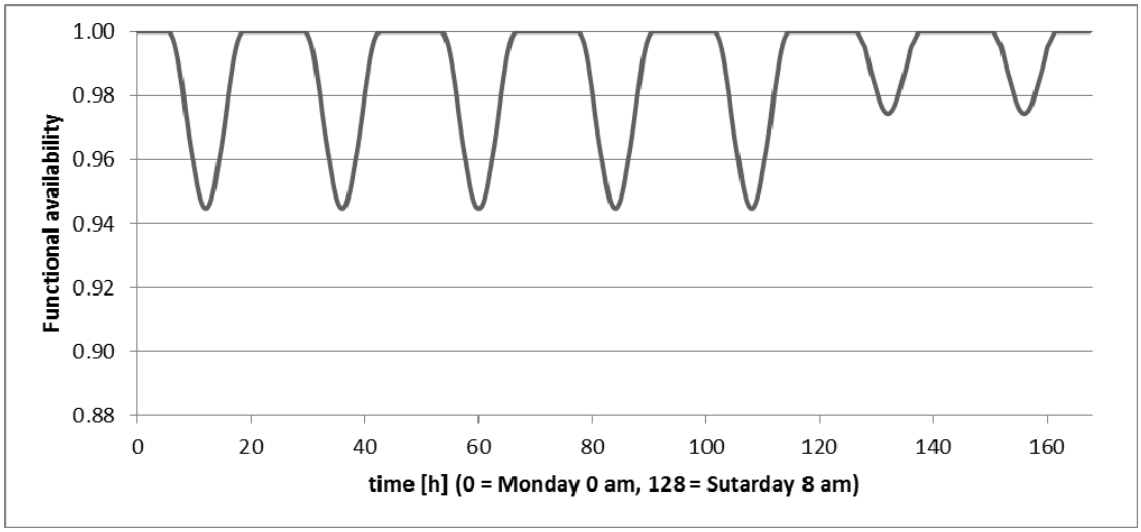

*Figure 5.*Functional availiabiliy for a tesbed system in afunction of a time (for one week)

5.2. Client availability

Form the web service client point of view reliability and functional reasons of getting no answer are undistinguished. Therefore, to measure client availability we propose to combine system and functional availability as given in below formula:

$$A_c(t) = \Pr(operational(t)) = A_S(t) \cdot A_f\left(nuser(t)\right) \quad (9)$$

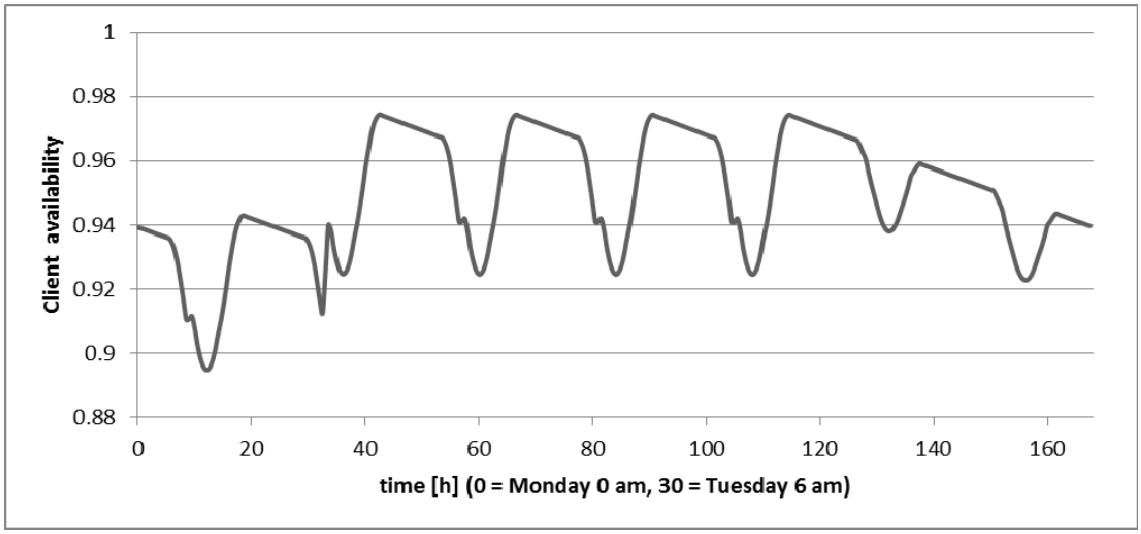

*Figure 6.*Client availiabiliy for a tesbed system in afunction of a time (for one week)

The client availability changes in a week for a testbed system with an input load from Figure 4 are presented on Figure 6. It could be noticed that availability varies in time following changes in input load and system reliability (due to a fact that system maintainer is not working 24 per day).

6. Conclusions

In this paper we have presented a reliability and functional analysis of Web systems based on modelling and analysis approach. Developed simulation software allows analysing the reliability (understood as the ability of a system to perform its required functions) for a given configuration of a Web system. Following reliability parameters are taken into account as follows: intensity of failures, mean failure analysis time and mean element replacement time. The reliability analysis stresses upon working hours of system maintainers. Functional parameters describe the web system choreography (interaction between tasks), system structure (number of hosts, host performance, service components placement on hosts and each task execution time) and input load (changes of user requests number in a time during a week).

Using the tool Web-based system configuration and maintenance procedures could be easily verified and different configuration could be compared. It makes the solution a powerful tool for increasing system availability and by that increasing user satisfaction.

In the future, we plan to extend our solution with other types of failures, which could model viruses, malwares or attack influence on the system performance.

Acknowledgment

The presented work was funded by the Polish National Science Centre under contract no. 4759/B/TO2/2011/40.

References

1. Barlow, R., Proschan, F . *Mathematical Theory of Reliability*. Philadelphia: Society for Industrial and Applied Mathematics, 1996. 345 p.
2. Birta, L., Arbez, G. *Modelling and Simulation: Exploring Dynamic System Behaviour*. London: Springer, 2007. 246 p.
3. Fishman, G. *Monte Carlo: Concepts, Algorithms, and Applications*. Springer-Verlag, 1996. 129 p.
4. Gold, N., Knight, C., Mohan, A. and M. Munro. Understanding service-oriented software, *IEEE Software*, Vol. 21, 2004, pp. 71–77.
5. Lipinski, Z. State Model of Service Reliability. *International Conference on Dependability of Computer Systems (DEPCOS-RELCOMEX'06), IEEE Computer Society, 2006,* pp. 35–42.
6. Liu, H., Chu, L. and W. Recker. Performance Evaluation of ITS Strategies Using Microscopic Simulation. *Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems, 2004,* pp. 255–270.
7. Liu, J. *Parallel Real-time Immersive Modelling Environment (PRIME), Scalable Simulation Framework (SSF), User's manual*. Colorado School of Mines Department of Mathematical and Computer Sciences, 2006. [Available online: http://prime.mines.edu/]
8. Michalska, K. and T. Walkowiak. Simulation approach to performance analysis information systems with load balancer. In: *Information systems architecture and technology: advances in Web-Age Information Systems*. 2009, pp. 269–278.
9. Walkowiak, T. *Information systems performance analysis using task-level simulator. DepCoS – RELCOMEX, 2009*. IEEE Computer Society Press, pp. 218−225.
10. Walkowiak, T. Simulation approach to Web system dependability analysis. In: *Summer Safety and Reliability Seminars, SSARS 2011, Gdańsk-Sopot, Poland, 03–09 July 2011 / Ed. by Krzysztof Kołowrocki, Joanna Soszyńska-Budny*. Gdynia: Polish Safety and Reliability Association. 2011. Vol. 1, pp. 197–204.
11. Walkowiak, T. and K. Michalska. Functional based reliability analysis of Web based information systems. *Dependable computer systems / Wojciech Zamojski, et al. (Eds.).* Berlin; Heidelberg: Springer, 2011, pp. 257–269.