2

Service based information systems analysis using task-level simulator

Tomasz Walkowiak
Wroclaw University of Technology
Poland

1. Introduction

Complex information systems (CIS) are nowadays the core of a large number of companies. And therefore, there is a large need to analyze various system configuration and chose the optimal solution during design and even operation of the information system.
In this paper we propose a common approach (Birta & Arbez, 2007) based on modelling and simulation. The aim of simulation is to calculate some performance metrics which should allow to compare different configuration taking into consideration technical (like performance) and economical (like price) aspects.
There is a large number of event driven computer network simulators, like OPNET, NS-2, QualNet, OMNeT++ or SSFNet/PRIME SSF(Liu, 2006; Nicol et al., 2003). However, they are mainly focused on a low level simulation (TCP/IP packets).
It is obvious that increasing the system details causes the simulation becoming useless due to the computational complexity and a large number of required parameter values to be given. On the other hand a high level of modelling could not allow to record required data for system measure calculation. Therefore, the level of system model details should be defined by requirements of the system measure calculation (Walkowiak, 2009).
Modelling and simulation based on TCP/IP packets level results in a large number of events during simulation and therefore in a long simulation time. It is a very good approach if one plans to analyze the influence of the traffic on the network performance. However in modern information systems high speed local networks are used. In a result for a large number of information systems (except media streaming ones) the local network traffic influence on the whole system performance is negligible.
Therefore, we want to propose a novel approach based on a higher level then TCP/IP packets. We will focus on a business service realized by an information system (Gold et al., 2007) and functional aspects of the system, i.e. performance aspects of business service realized by an information system (like buying a book in the internet bookstore). We assume that the main goal, taken into consideration during design and operation of the CIS, is to fulfil the user requirements, which could be seen as some requirements to perform a user tasks within a given time limit. Therefore, the presented in the chapter modelling and simulation will be focused on a process of execution of a user request, understand as a sequence of task realised on technical services provided by the system.

The structure of the chapter is as follows. In Section 2, a model of information system is given. In Section 3, information on simulator implementation is given, next exemplars information system is analysed and simulation results are presented. It is followed by information on graphical user interface. Finally, there are conclusions and plans for further work.

2. Computer information system modelling

As it was mentioned in the introduction we decided to analyze the CIS from the business service point of view. Generally speaking users of the system are generating tasks which are being realized by the CIS. 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 has to be allocated on a given host. Therefore, we can model CIS as a 4-tuple (Walkowiak, 2009):

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

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

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

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

where: H - set of hosts (computers); N – computer network.
We have assumed that the aspects of TCP/IP traffic are negligible therefore we will model the network communication as a random delay. 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). The time delay is modelled by a Gaussian distribution with a standard deviation equal to 10% of mean value.
The main technical infrastructure of the CIS are hosts. Each host is described by its 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.

We have distinguished a special kind of technical service witch models a load balancer (Aweya et al., 2002). A load balancer is described by its name and a limit of tasks (like all technical services) and additionally by a list of technical services, it sends requests to.

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 (Michalska & Walkowiak, 2008).
Therefore, *BS* is modelled as 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. We have distinguished two kinds of task: local and external. If request is send to service component and this component is able to respond without asking other service component than this tasks is assumed to be local. If request is send to service component and this component must ask another service component for response then than this tasks is assumed to be external. Each task is described by its name, task processing time parameter and in case of external task 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*) consist of 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).

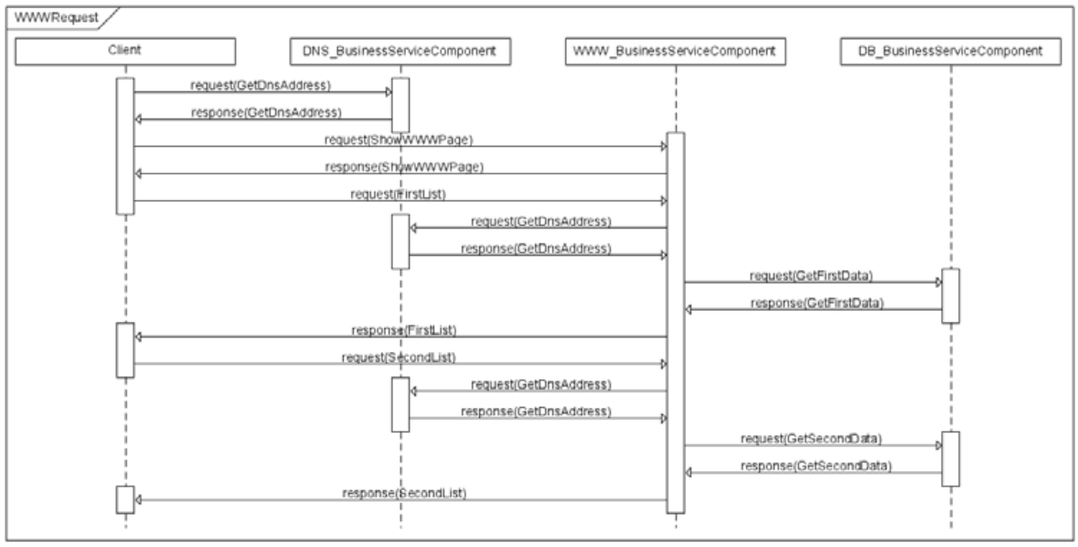


Fig. 1. Task and business services interaction

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 gives a sequence of a user task execution as presented on exemplar Fig. 1.
The user request execution time in the system is calculated as a sum of times required for TCP/IP communication and times of tasks processing on a given host.
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 (time-out 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.
Let $\tau_1, \tau_2, ..., \tau_e$ be a time moments when a task (t_j^i) with some execution time $(executiontime(t_j^i))$ is starting or finishing processing on a host $h = allocation(t_j^i)$. Let $number(h, \tau)$ 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 responses. Therefore, the time when task t_j^i finishes its execution τ_e has to fulfill a following rule:

$$\sum_{k=2}^{e} (\tau_k - \tau_{k-1}) \frac{performance(h)}{number(h, \tau_{k-1})} = executiontime(t_j^i) \quad . \tag{3}$$

Having above notation the task processing time is equal to:

$$pt(t_j^i) = \tau_e - \tau_1 \, . \tag{4}$$

3. Task-level simulator

Once a model has been developed, it is executed on a computer. It is done by a computer program which steps through time. One way of doing it is so called event-simulation. Which is based on a idea of event, which could is described by time of event occurring, type of event (in case of CIS it could be host failure) and element or set of elements of the system on which event has its influence. The simulation is done by analyzing a queue of event (sorted by time of event occurring) while updating the states of system elements according to rules related to a proper type of event.
As it was described in section 2, the network connections are modelled as a random delays. Therefore, we were not able to use mentioned in the introduction computer network simulators but we have to develop a new one (Walkowiak, 2009). The event-simulation

program could be written in general purpose programming language (like C++), in fast prototyping environment (like Matlab) or special purpose discrete-event simulation kernels. One of such kernels, is the Scalable Simulation Framework (SSF) (Nicol et al., 2003) which is a used for SSFNet (Nicol et al., 2003) computer network simulator. SSF is an object-oriented API - a collection of class interfaces with prototype implementations. It is available in C++ and Java. SSF API defines just five base classes: Entity, inChannel, outChannel, Process, and Event. The communication between entities and delivery of events is done by channels (channel mappings connects entities).

For the purpose of simulating CIS we have used Parallel Real-time Immersive Modeling Environment (PRIME) (Liu, 2006) implementation of SSF due to much better documentation then available for original SSF. We have developed a generic class (named BSObject) derived from SSF Entity which is a base of classes modeling CIS objects: host and client which models the behavior of CIS presented in section 2. Each object of BSObject class is connected with all other objects of that type by SFF channels what allows communication between them. In the first approach we have realized each client as a separated object. However, in case of increasing of the number of replicated clients the number of channels increases in power of two resulting in a large memory consumption and a long time for initialization simulation objects. Therefore, we have changed the implementation, and each replicated client is represented by one object.

The developed simulator is called SSF.BS (from SSF – the simulation framework and BS – business service).

4. Computer information system simulation analysis

4.1 First case study

For testing purposes of presented CIS system model (section 2) and developed extension of SSF (SSF.BS, section 3) we have analysed a case study information system. It consists of one type of client placed somewhere in internet, firewall, three hosts (Figure 2), three technical services and three business service components. An interaction between a client and tasks of each business service component is presented on UML diagram in Figure 1. The CIS structure as well as other functional parameters were described in a DML file (see example in Figure 3). The Domain Modeling Language (DML) (Nicol et al., 2003) is a SSF specific text-based language which includes a hierarchical list of attributes used to describe the topology of the model and model attributes values.

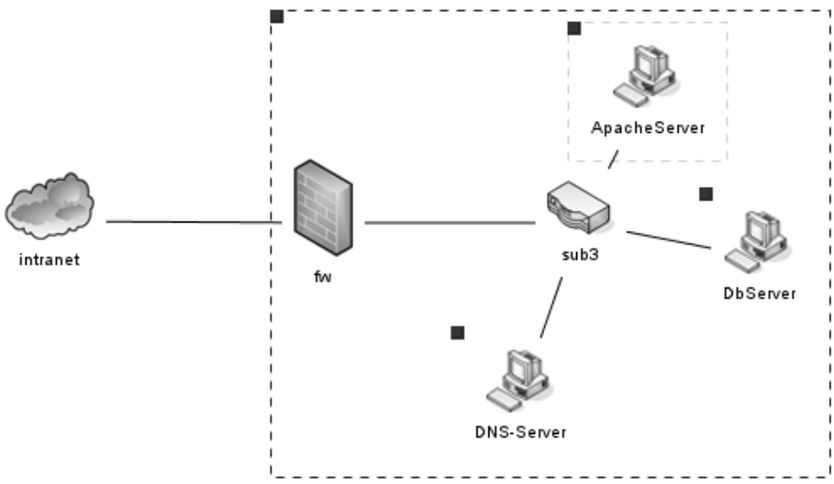


Fig. 2. Case study system overview

```
Net [
Host [
  Name DNS-Server
     Service [
        Name DNSService
        Limit 110
       LocalTask [
         Name GetDnsAddress
         Time 0.01]]]
     …
Client [
  Name Client
  Replicate 1000
  Sleep 10.0
  Activity [
     Name WWWRequest
     TaskCall [
       Host DNS-Server
       Service DNSService
       Task GetDnsAddress
          ]
  …
```

Fig. 3. Exemplar CIS description in DML file

In the presented information system we have observed the response time to a client request in a function of number of clients. The achieved results are presented in Figure 4.

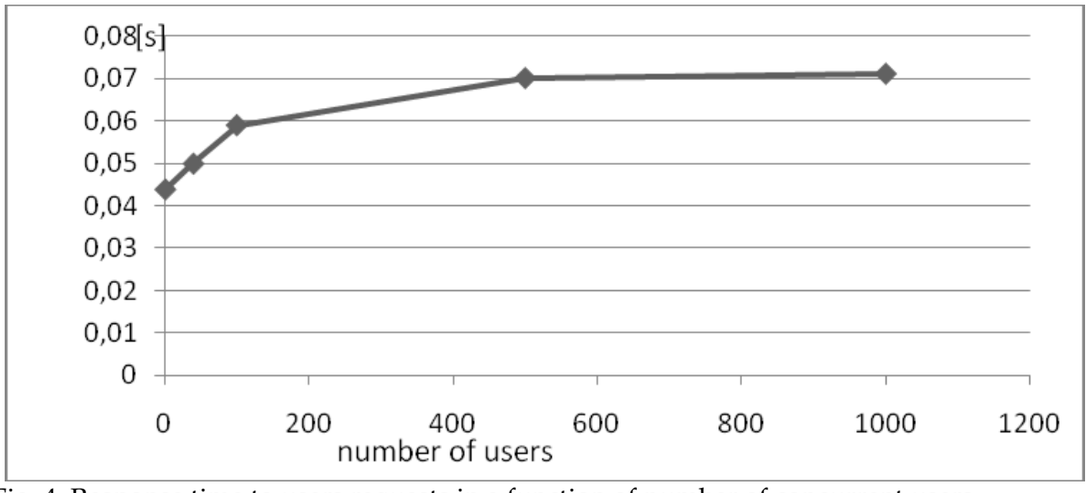


Fig. 4. Response time to users requests in a function of number of concurrent users

4.2. Simulator performance analysis

Next, we have tested the SSF.BS simulator performance and scalability. We calculated the time of running one batch of simulation of the exemplar IS described in previous chapter on a 2.80 GHz Intel Core Duo machine. We have compared the performance results with PWR.SSF.Net simulator (Zyla & Caban 2008) developed in Java. The CIS model used in PWR.SSF.Net differs from SSF.BS mainly in a method of calculation a task performance time and therefore the results of simulating cannot be compared. As it could be noticed on Figure 5 & 6 the presented in the paper simulator (SSF.BS) simulates the CIS in shorter time, and a difference with PWR.SSF.Net is increasing with an increase of number of users.

For a number of concurrent users less than 300 (Figure 5) the SSF.BS is 10 times faster than PWR.SSF.Net. The main reason of this difference is the level of modelling details. In both cases simulators perform similar number of events per second. However, PWR.SSF.Net simulates the transmission of TCP/IP packets whereas SSF.BS works on higher level the tasks and therefore in case of presented here approach the number of events is smaller.

Not, only computational complexity of SSF.BS is lower than PWR.SSF.Net but also the usage of memory for SSF.BS is much smaller. For a case study example the SSF.BS requires 1.8 Mbytes for 0.1 client requests per second upto 4.8 Mbytes for a 1000 concurrent users. In case of PWR.SSF.Net it is hard to state the memory usage due to the memory management techniques in Java. This is the problem of enlarging the difference of speed between analysed simulators. For number of clients more then 300 (Figure 6) Java based PWR.SSF.Net starts to have problems with memory management and large number of processing time is used by JVM garbage collector (even Java based simulator was started 1 Gbyte memory limit). It results in 1000 faster simulation of SSF.BS in case of 1000 concurrent users.

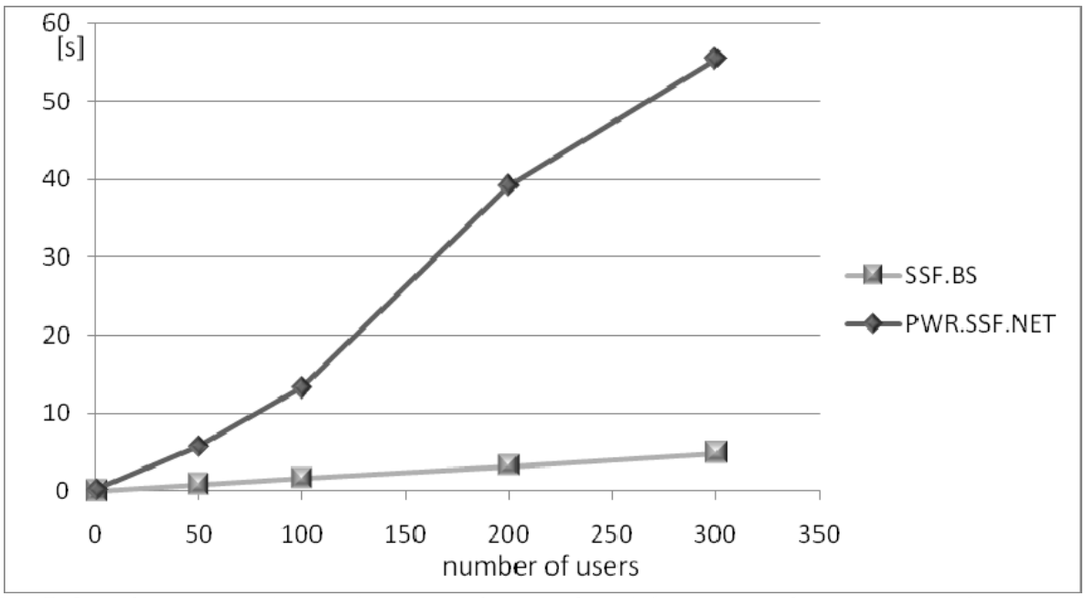


Fig. 5. Simulation time (time of running the simulator) for case study system in a function of number of users (till 300 concurrent users)

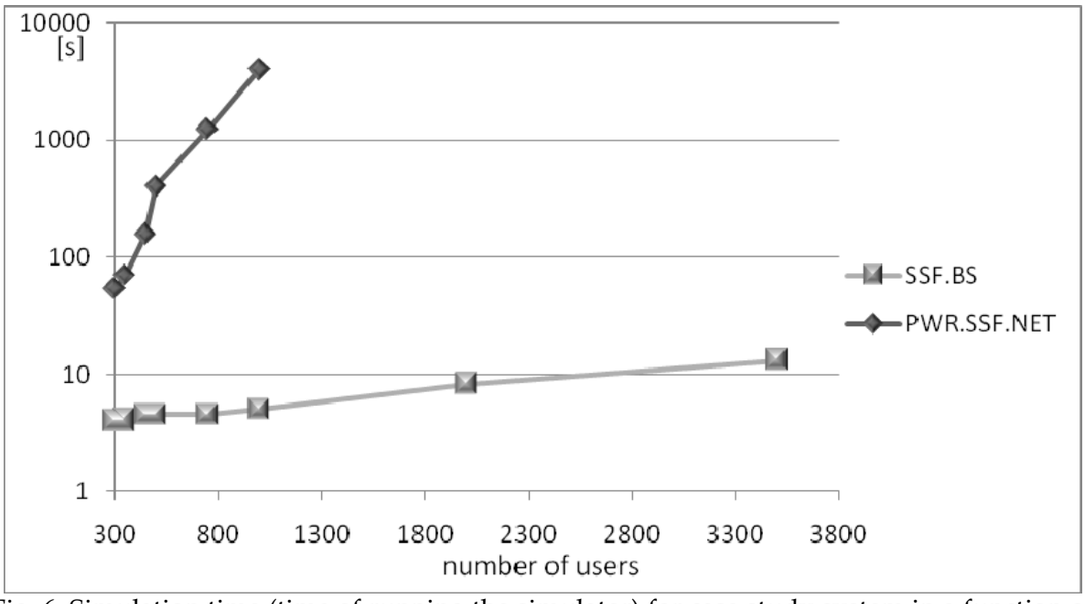


Fig. 6. Simulation time (time of running the simulator) for case study system in a function of number of users (for more than 300 users)

4.3. Second case study – load balancer

A very common technique of achieving height availability of their services in CIS is using a load balancer. Load balancer allows a traffic distribution among replicated services on a server farm. Therefore, the most common load balancing algorithm – *round robin* (Aweya, et al. 2002) - has been implemented in the SSF.BS.

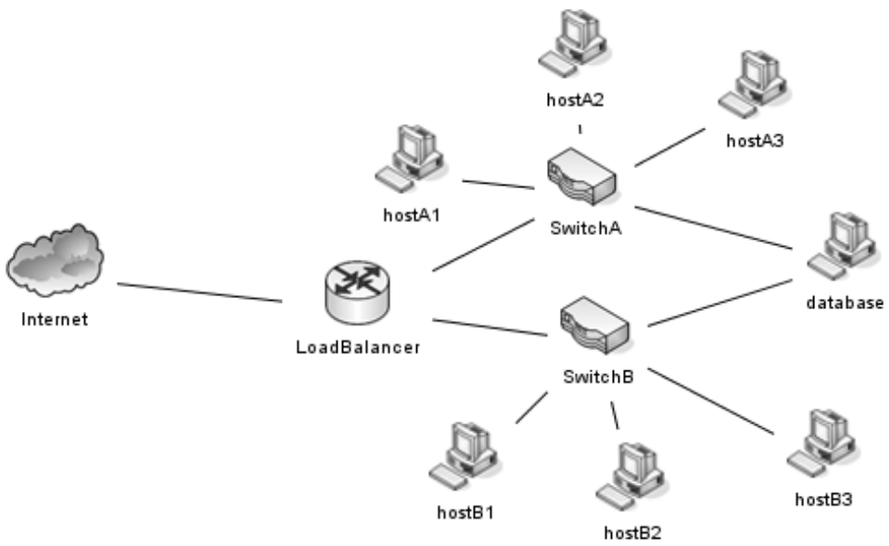


Fig. 7. Load balancer case study system overview

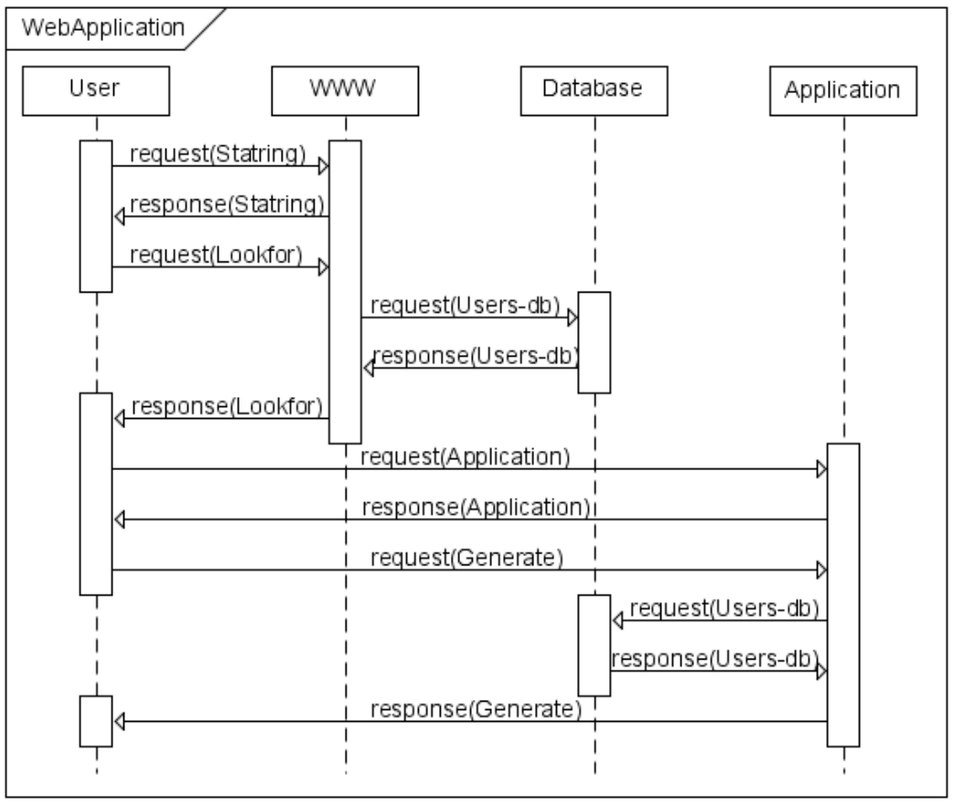


Fig. 8. Task and business services interaction for case study

For the case study analysis of CIS with load balancing we propose an exemplar service system illustrated in Fig.7 . Essentially the test-bed system consists of two server farms A (included host „hostA1"-„hostA3") and B (included host „hostB1"-„hostB3") and a database server. Both farms are connected with LoadBalancer as a gate to internet users. For the case study, let us imagine, that this system is responsible for some Web Application that allows searching the database and executes a Tomcat based application. Fig. 8 shows choreography of this service, based on three service components. WWW service component has been replicated on hosts: A1-A3, Application of on hosts: B1- B3 and Database is not replicated is placed on one host. For this scenario two configuration has been proposed: first (I) standard and second (II) with all hosts with doubled performance parameter.
The achieved simulation results, the response time to user requests in a function of number of concurrent users is presented in Figure 9. The simulation time was set to 1000 seconds. The limit of concurrent tasks for all technical services was equal to 1000, whereas the inter-activity delay time equal to 1 s. As it could be expected the response time for configurations II is almost twice shorter than for configuration I. However, if we slightly change configuration II, setting the performance of database host equal to the value used in configurations I the resulting response time will be very similar to results of configuration I. These small experiment shows the ability of simulator to compare performance of different system configurations.

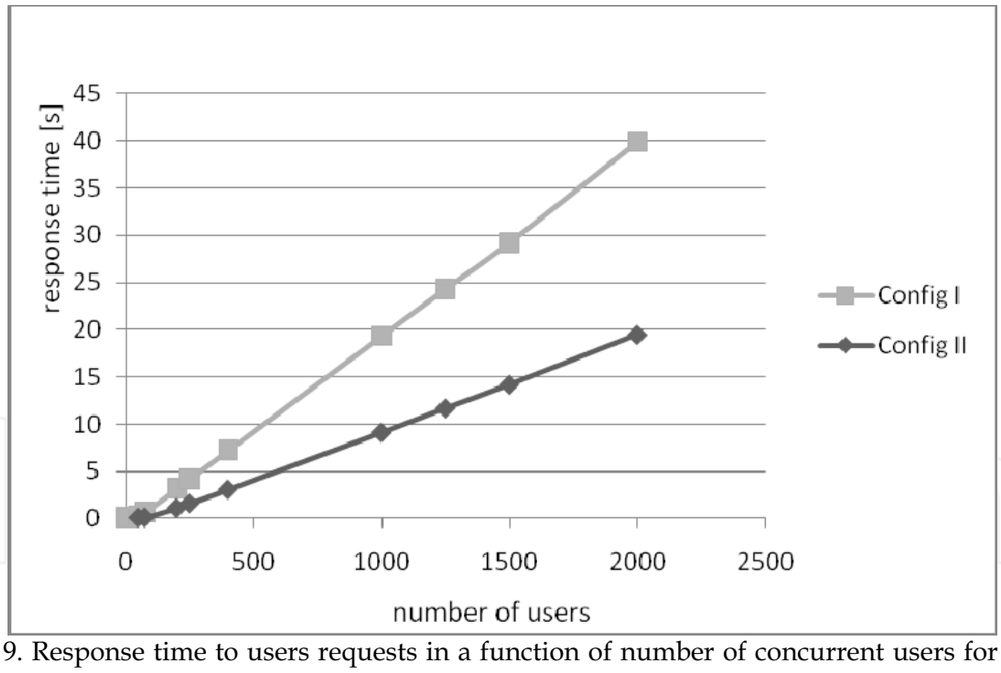


Fig. 9. Response time to users requests in a function of number of concurrent users for two configurations of load balancer case study

5. Graphical interface

The previous section showed the possibilities of using SSF.BS simulator and its good computational performance capabilities. However, nowadays the practical usage of any computer tool requires a good graphical interface. As it was mentioned in the section 3, all

input information of modelled CIS is described in DML text file. Even the DML file format is simple (Figure 3), it is difficult for a human being to describe a CIS with large number of host and sophisticated service interaction without any error in text file.
Within the framework of DESEREC EU grant (http://www.deserec.eu) a Java based graphical tool called "Integrated Analysis Environment" (IAE) was developed (Michalska & Walkowiak, 2008b) for a usage of PWR.SSF.Net simulator. After a few changes in IAE it was adopted to SSF.BS simulator.
In IAE we took into consideration an inconvenient format of Domain Modelling Language and we proposed its XML representation with all supplements attributes of proposed extended simulation framework - called XDML. Creation of XDML language gave many processing possibilities. IAE framework using JAXB techniques and implemented translation methods creates one model (XDML) from other modelling languages: system infrastructure from SDL (System Description Language, http://www.positif.org/) and task interaction from WS-CDL (WebServices Choreography Description Language, http://www.w3.org/). This XDML model is visualized showing the structure of the network and it's element (Figure 10). Each network element has several functional parameters and user can graphically edit this information. 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 simplified its work.
After setting up all parameters of network elements and service components the user is able to perform simulation. It is done by transforming XDML into DML. The resulting DML file is then simulated. Simulation is integrated into IAE since both tools are developed in Java therefore user can see on the screen text output from the simulator on-line. The results from simulation (output file from simulator) are caught by IAE and response time to user requests is calculated and displayed.

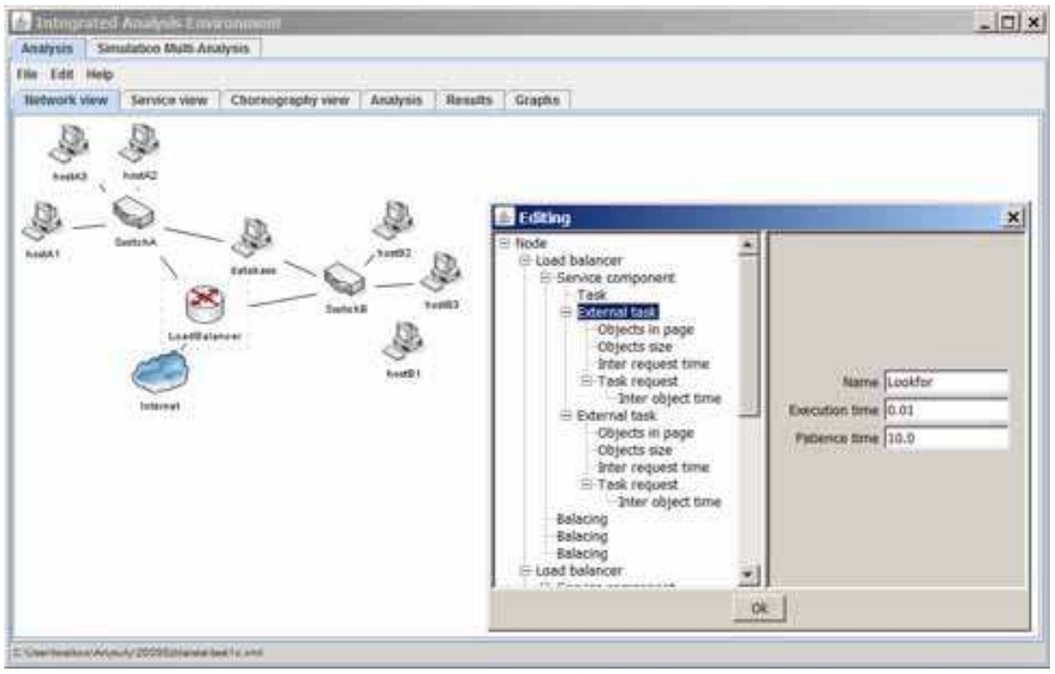


Fig. 10. Integrated Analysis Environment - screenshot

6. Conclusion

We have presented a simulation approach to functional analysis of complex information systems. Developed simulation software allows to analyze the effectiveness (understood in given exemplar as a the response time to a client request) of a given configuration of computer system. Changes in a host performance or in a number of clients can be easily verified. Also, some economic analysis could be done following the idea presented in (Walkowiak & Mazurkiewicz, 2005). The implementation of CIS simulator done based on SSF allows to apply in a simple and fast way changes in the CIS model. Also the time performance of SSF kernel results in a very effective simulator of CIS.
We are now working on implementing other load balancing algorithms what should allow to analyze a wider range of enterprise information systems and compare different load balancing algorithms.
We also plan to extend the model and simulator to include the reliability model of technical infrastructure components. It should allow to measure the availability of a business service in a function of functional and reliability parameters of information systems components.

7. References

Avižienis, A. ; Laprie, J. ; Randell, B. (2000). Fundamental Concepts of Dependability. *Proceedinggs of 3rd Information Survivability Workshop* (ISW-2000), Boston, Massachusetts

Aweya, J.; Ouellette, M.; Montuno, D.; Doray, B.; Felske, K. (2002). An adaptive load balancing scheme for web servers. *International Journal of Network Management*, Vol. 12

Birta, L. ; Arbez, G. (2007). Modelling and Simulation: Exploring Dynamic System Behaviour. Springer, London

Gold, N.; Knight, C. ; Mohan, A.; Munro, M. (2004). Understanding service-oriented software. *IEEE Software*, Vol. 21, 71– 77

Liu, J. (2006). Parallel Real-time Immersive Modeling Environment (PRIME), Scalable Simulation Framework (SSF), User's maual. Colorado School of Mines Department of Mathematical and Computer Sciences, 2006, [Online]. Available: http://prime.mines.edu/

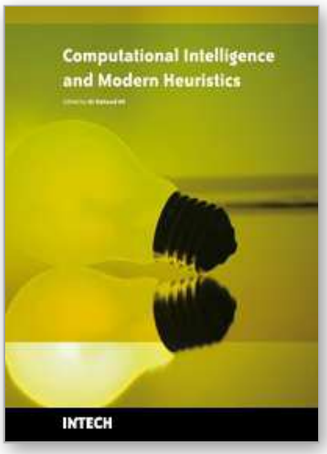
Nicol, D. ; Liu, J., Liljenstam, M. ; Guanhua, Y. (2003). Simulation of large scale networks using SSF. *Proceedings of the 2003 Winter Simulation Conference*, Vol. 1, pp. 650–657, New Orleans,

Michalska, K. ; Walkowiak, T. (2008). Hierarchical Approach to Dependability Analysis of Information Systems by Modeling and Simulation. *Proceedings of the 2008 Second international Conference on Emerging Security information, Systems and Technologies,* , pp. 356-361 Cap Esterel, IEEE Computer Society, Washington

Walkowiak, T. ; Mazurkiewicz, J. (2005) Reliability and Functional Analysis of Discrete Transport System with Dispatcher. *Advances in Safety and Reliability, European Safety and Reliability Conference – ESREL 2005*, Gdynia, pp. 2017-2023, Taylor & Francis Group, London

Walkowiak, T. (2009). Information systems performance analysis using task-level simulator, *Proceedings of International Conference on Dependability of Computer Systems*, pp. 218-225, Brunow, IEEE Computer Society Press, Los Alamitos

Zyla, M.; Caban, D. (2008). Dependability Analysis of SOA systems. *Proceedings of International Conference on Dependability of Computer Systems*, pp. 301–306, Szklarska Poreba, IEEE Computer Society Press, Los Alamitos



Computational Intelligence and Modern Heuristics

Edited by Al-Dahoud Ali

ISBN 978-953-7619-28-2
Hard cover, 348 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

The chapters of this book are collected mainly from the best selected papers that have been published in the 4th International conference on Information Technology ICIT 2009, that has been held in Al-Zaytoonah University, Jordan in the period 3-5/6/2009. The other chapters have been collected as related works to the topics of the book.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen