HET-NETs 2010
ISBN XXX–XXX
pp. xx–xx

Simulation environment for delivering quality of service in systems based on service-oriented architecture paradigm

ADAM GRZECH [a] PIOTR RYGIELSKI [a] PAWEŁ ŚWIĄTEK [a]

[a]Institute of Computer Science
Wroclaw University of Technology, Poland
{adam.grzech, piotr.rygielski}@pwr.wroc.pl

Abstract: In this paper a model of complex service in service-oriented architecture (*SOA*) system is presented. A complex service is composed with a set of atomic services. Each atomic service is characterized with its own non-functional parameters what allows to formulate quality of servie optimization tasks. A simulation environment has been developed to allow experiments execution to determine quality of service (*QoS*) of composed service.

Keywords: : quality of service, service-oriented architecture, simulation

1. Introduction

Recent popularity of system based on service-oriented architecture (*SOA*) paradigm has lead to growth of interest concerning quality of service level provisioning in such systems. Efficient management of system resources can lead to delay decrease, cost optimization and security level increase [6].

In this paper a model of serially connected atomic services is considered. Atomic services are organized into layers where each atomic service in one layer has equal functionality and differs in non-functional parameters values such as processing speed, security level, cost of execution etc. Incoming requests should be distributed in such way that user requirements concerning quality of service are satisfied and are profitable for service provider. For testing purposes for such service distribution algorithms a simulation environment has been developed.

The paper is organised as follows. In section 2 a serial-parallel complex service model is presented. Problem of resource distribution for incoming service request has been formulated in section 3. In section 4 a simulation environment proposed as a testbed for quality of service provisioning algorithms is described. Usage of developed simulation software is presented by example in section 5. Section 6 is dedicated for final remarks and future work outline.

2. Complex service model

It is assumed that new incoming i-th complex service request is characterized by proper Service Level Agreement description denoted by $SLA(i)$. The $SLA(i)$ is composed of two parts describing functional and nonfunctional requirements, respectively $SLA_f(i)$ and $SLA_{nf}(i)$. The first part characterizes functionalities that have to be performed, while the second contains values of parameters representing various quality of service aspects. The $SLA_f(i)$ is a set of functionalities subsets:

$$SLA_f(i) = \{\Gamma_{i1}, \Gamma_{i2}, \ldots, \Gamma_{ij}, \ldots, \Gamma_{in_i}\} \tag{1}$$

where:

- $\Gamma_{i1} \prec \Gamma_{i2} \prec \ldots \prec \Gamma_{ij} \prec \ldots \prec \Gamma_{in_i}$ ordered subset of distinguished functionalities subsets required by i-th complex service request; $\Gamma_{ij} \prec \Gamma_{ij+1}$ (for $j = 1, 2, \ldots, n_i - 1$) denotes that delivery of functionalities from the subset Γ_{ij+1} cannot start before completing functionalities from the Γ_{ij} subset.
- $\Gamma_{ij} = \{\varphi_{ij1}, \varphi_{ij2}, \ldots, \varphi_{ijm_j}\}$ (for $i = 1, 2, \ldots, n_i$) is a subset of functionalities φ_{ijk} ($k = 1, 2, \ldots, m_j$) that may be delivered in parallel manner (within Γ_{ij} subset); the formerly mentioned feature of particular functionalities are denoted by $\varphi_{ijk} \parallel \varphi_{ijl}$ ($\varphi_{ijk}, \varphi_{ijl} \in \Gamma_{ij}$ for $k, l = 1, 2, \ldots, m + j$ and $k \neq l$).

The proposed scheme covers all possible cases; $n_i = 1$ means that all required functionalities may be delivered in parallel manner, while $m_j = 1$ (for $j = 1, 2, \ldots, n_i$) means that all required functionalities have to be delivered in sequence.

It is also assumed that the φ_{ijk} ($j = 1, 2, \ldots, n_i$ and $k = 1, 2, \ldots, m_j$) functionalities are delivered by atomic services available at the computer system in several versions.

The nonfunctional requirements may be decomposed in a similar manner, i.e.:

$$SLA_nf(i) = \{H_{i1}, H_{i2}, \ldots, H_{ij}, \ldots, H_{in_i}\} \tag{2}$$

where $H_{ij} = \{\gamma_{ij1}, \gamma_{ij2}, \ldots, \gamma_{ijm_j}\}$ is a subset of nonfunctional requirements related respectively to the $\Gamma_{ij} = \{\varphi_{ij1}, \varphi_{ij2}, \ldots, \varphi_{ijm_j}\}$ subset of functionalities.

According to the above assumption the $SLA_f(i)$ of the i-th complex service request may be translated into ordered subsets of atomic services:

$$SLA_f(i) = \{\Gamma_{i1}, \Gamma_{i2}, \ldots, \Gamma_{ij}, \ldots, \Gamma_{in_i}\} \Rightarrow \{AS_{i1}, AS_{i2}, \ldots, AS_{ij}, \ldots, AS_{in_i}\}, \tag{3}$$

where $\{AS_{i1}, AS_{i2}, \ldots, AS_{ij}, \ldots AS_{in_i}\}$ is a sequence of atomic services subsets satisfying an order ($AS_{i1} \prec AS_{i2} \prec \ldots \prec AS_{ij} \prec \ldots \prec AS_{in_i}$) predefined by order in the functionalities subsets. The order in sequence of atomic services is interpreted as the order in functionalities subsets: $AS_{ij} \prec AS_{i,j+1}$ (for $j = 1, 2, \ldots, n_i - 1$) states that atomic services from subsets $AS_{i,j+1}$ cannot be started before all services from the AS_{ij} subset are completed.

Each subset of atomic services AS_{ij} (for $j = 1, 2, \ldots, n_i$) contains a_{ijk} atomic services (for $k = 1, 2, \ldots, m_j$) available at the computer system in several versions a_{ijkl} ($l = 1, 2, \ldots, m_k$). Moreover, it is assumed that any version a_{ijkl} ($l = 1, 2, \ldots, m_k$) of the particular a_{ijk} atomic services (for $k = 1, 2, \ldots, m_j$) assures the same required functionality φ_{ijk} and satisfies nonfunctional requirements at various levels.

The above assumption means that – if $fun(a_{ijkl})$ and $nfun(a_{ijkl})$ denote, respectively, functionality and level of nonfunctional requirements satisfaction delivered by l-th version of k-th atomic service ($a_{ijkl} \in AS_{ij}$) – the following conditions are satisfied:

- $fun(a_{ijkl}) = \varphi_{ijk}$ for $l = 1, 2, \ldots, m_k$,
- $nfun(a_{ijkl}) \neq nfun(a_{ijkr})$ for $l, r = 1, 2, \ldots, m_k$ and $l \neq r$.

The ordered functionalities subsets $SLA_f(i)$ determines possible level of parallelism at the i-th requested complex service performance (in the particular environment). The parallelism level $l_p(i)$ for i-th requested complex service is uniquely defined by the maximal number of atomic services that may be performed in parallel manner at distinguished subsets of functionalities ($SLA_f(i)$), i.e.,

$$l_p(i) = \max\{m_1, m_2, \ldots, m_j, \ldots, m_{n_i}\}. \tag{4}$$

The possible level of parallelism may be utilized or not in processing of the i-th requested complex service. Based on the above notations and definitions two extreme compositions exist. The first one utilizes possible parallelism (available due to computation resources parallelism), while the second extreme composition means that the requested functionalities are delivered one-by-one (no computation and communication resources parallelism).

The above presented discussion may be summarized as follows. The known functional requirements $SLA_f(i)$ may be presented as a sequence of subsets of functionalities, where the size of the mentioned latter subsets depends on possible level of parallelism. The available level of parallelism defines a set of possible performance scenarios according to which the requested complex service may be delivered. The space of possible solutions is limited – from one side – by the highest possible level parallelism and – from the another side – by natural scenario, where all requested atomic services are performed in sequence.

The mentioned above extreme compositions determines some set of possible i-th requested complex service delivery scenarios. The possible scenario can be represented by a set of graphs $G(i)$ – nodes of graph represent particular atomic services assuring i-th requested complex service functionalities, while graph edges represent an order according to which atomic services functionalities have to be delivered.

The set of all possible graphs $G(i)$ $\Big(G(i) = \{G_{i1}, G_{i2}, \ldots, G_{is}\}\Big)$ assures the requested functionality, but offers various level of nonfunctional requirement satisfaction. The latter may be obtained (and optimized) assuming that at least one node of the particular graph contains at least two versions of the requested atomic service.

3. Problem formulation

In general, the optimization task of maximizing delivered quality may be formulated as follows:

For given:

- Subsets of atomic services $a_{ijkl} \in a_{ijk} \in AS_{ij}$
- Set of all possible graphs $G(i)$ for i-th requested complex service
- Subsets of nonfunctional requirements H_{ij}
- Processing scheme given by order $AS_{i1} \prec AS_{i2} \prec \ldots \prec AS_{ij} \prec \ldots \prec AS_{in_i}$

Find: such subset of atomic services versions a_{ijkl} that executed with respect to processing scheme maximizes quality of service $SLA^*_{nf}(i)$.

$$SLA^*_{nf}(i) \leftarrow \max_{G_{i1},G_{i2},\ldots,G_{is}} \left\{ \max_{a_{ijkl} \in a_{ijk} \in AS_{ij}} \{H_{i1}, H_{i2}, \ldots, H_{in_i}\} \right\}. \tag{5}$$

The latter task may be reduced where the particular i-th requested complex service composition (i.e., an graph equivalent to particular i-th requested complex service processing scheme) is assumed. In such a case the optimization task can be formulated as:

$$SLA^*_{nf}(i, G_i) \leftarrow \max_{a_{ijkl} \in a_{ijk} \in AS_{ij}} \{H_{i1}, H_{i2}, \ldots, H_{in_i}\}. \tag{6}$$

The above formulated task means that the optimal versions of atomic services, determined by the selected i-th requested complex service performance scenario, should be selected.

4. Simulation environment

Testing new methods and approaches for improving system quality on real system is time-consuming and can be difficult, expensive and cause damage to system [1]. Therefore there is a need to develop a testbed for testing such mechanisms. There are several simulation environments dedicated for communication networks and in this work OMNeT++ was used. OMNeT++ is an open source, component-based, modular and open-architecture simulation framework written in C++ with good documentation and on-line support. As an *IDE* (Integrated Development Environment) Eclipse CDT is used, so OMNeT++ can be launched at the most popular operation systems – Linux, Mac OS X and Windows.

Developed environment contains four distinguishable modules: generator module, request distribution unit (*RDU*), set of layers with atomic services and a sink module where global statistics are collected and requests are removed from system. Generator module is a complex module and consists of sub-generators each generating requests from different

classes. Each class of requests is characterized with parameters describing e.g. service method (based on Integrated services (IntServ) model, Differentiated services (DiffServ) model or best effort [5]), request data size, requests interarrival time, non-functional requirements and others. The last developed component was an adapter which allows to connect simulation environment to a real network [4]. Generated requests are sent to request distribution unit which is responsible for choosing an execution path in system. There is also implemented a request flow shaping mechanism like token bucket and admission control algorithm. Structure of considered system is presented on figure 1.

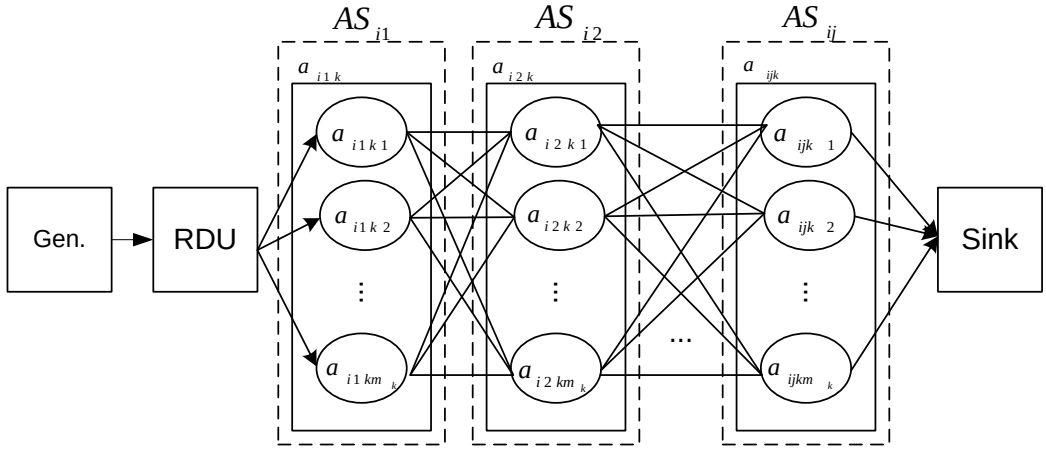

Fig. 1: Structure of considered system with distinguished modules: generator module (*Gen.*), request distribution unit (*RDU*), atomic services structure and sink module.

Assignment of resources is performed to a system which structure is composed by an outer mechanism of composition [2]. One can consider a situation when some atomic services subsets AS_{ij} are executed in parallel or in loop, so different layers can be interpreted as presented on figure 2.

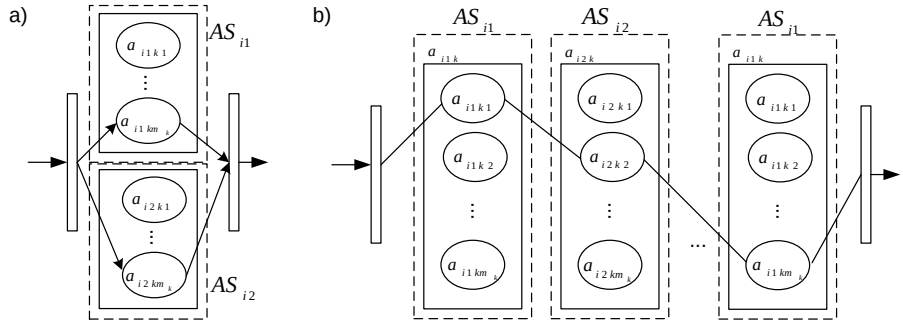

Fig. 2. Exemplary interpretation of atomic services structure: a) parallel, b) single loop

Each atomic service version a_{ijkl} is modelled as a single-queue single processor node

but in general it can be multi-queue single processor. Distinction of more than one queue can differentiate requests e.g. coming from different classes or being serviced in different way. Noteworthy is also a fact that communication channels can be modelled in flexible way, user can tune such parameters as transfer delay, channel capacity or packet loss ratio.

Last module of presented environment is responsible for removing serviced request from the system and collecting data about the progress of service. Collected data is useful for determination of quality of service level. Configuration of data being collected is easy and one is able to collect additional data if needed.

Simulation environment was designed as a testbed for system resource allocation algorithms (choosing best subset of atomic service versions) so possibility of algorithm replacement without changing simulator structure is a strong advantage.

5. Simulation environment usage example

In order to evaluate the quality of service delivered by the considered service-oriented system there were following algorithms of service requests distribution implemented: *IS Greedy*, *BE Greedy*, *BE DAG-SP*, *BE random* and *BE round-robin*. Prefix *BE* in algorithms names means that algorithm deliver quality of service based on *best effort* approach (without any warranty) and *IS* algorithm was based on integrated services model. *IS* algorithm uses reservation mechanism of computational and communication resources so the quality of service can be guaranteed. Repository of reference algorithms based on *Best-effort* approach includes: *BE Greedy* which checks every possibility to choose atomic services versions subset to optimize quality; *BE DAG-SP* (directed acyclic graph – shortest path) which finds shortest path in weighted graph taking into consideration communication delays between atomic services; *BE random* and *BE round-robin* which chooses respectively random and sequential atomic service versions.

To present potential of described tool a comparison of reference algorithms was performed. One can formulate problem as follows: for given current atomic services queue lengths, processing speeds, communication channel delay and request size, find such subset of atomic services that minimizes complex service execution delay.

To examine presented algorithm efficiency the simulation environment was configured as follows. There were three subsets of atomic services each delivering same functionality $n_i = 3$ and each containing three versions of atomic services $\forall a_{ijkl}, j = 1, 2, 3$: $k = 1, l = 3$. Communication delay of traffic channels was proportional to the request size which was random with exponential distribution with mean 200 bytes. Atomic services processing speed was also random value with uniform distribution from range 3 to 10 $kbytes/s$. Average complex service execution delays for each considered algorithm are presented on figure 3.

The *IS* algorithm delivered lowest delay because of resource reservation and higher service priority of requests coming from that class than from *best-effort* one. During resource reservation period no *best-effort* service request can be serviced except of situation, when request with resource availability guarantee will leave atomic service earlier than end of

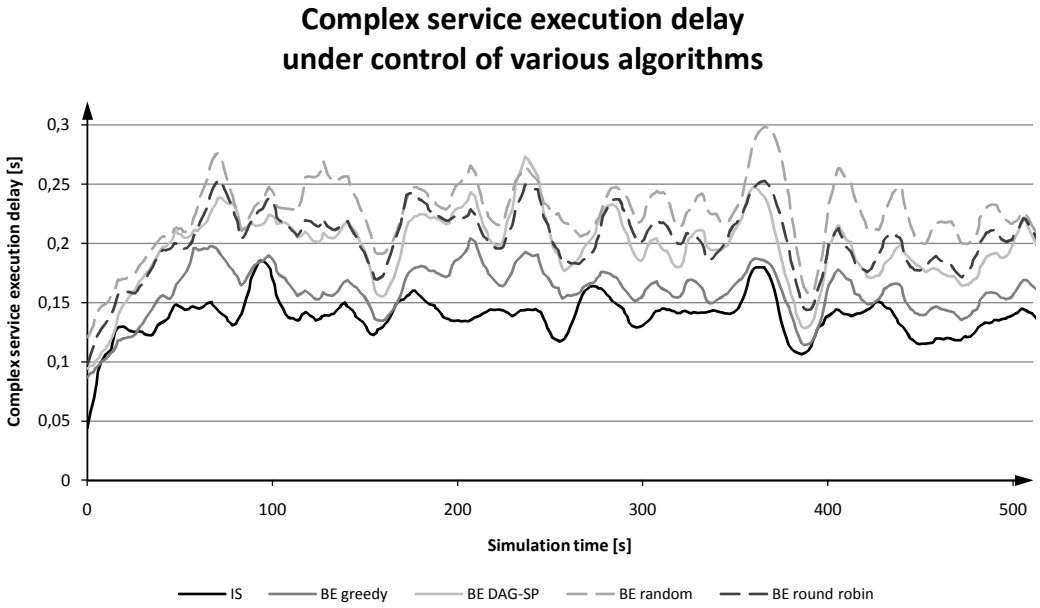

Fig. 3: Results of exemplary simulation run. Comparison of complex service execution delay under control of various algorithms. Requests served with *IS* algorithm were present in system simultaneously with requests serviced by reference *BE* algorithms.

reservation period. The best from *best-effort* algorithms group was greedy version of that algorithm. In situation when requests serviced by *IS* algorithms do not share resources with those being serviced by *BE* algorithm both delivers the same delay [3]. Delay delivered by *BE DAG-SP* algorithm was a little higher and quality for *BE round-robin* and *BE random* was much worse than other reference algorithms.

6. Final remarks

In this paper model of complex service processing scheme was presented. Distinction of functional and nonfunctional requirements of SLA request was introduced and problem of delivering quality of service was formulated. For the evaluation of performance of the presented system a simulation environment in OMNeT++ was implemented. The simulation environment enables to implement various algorithms of request distribution in the presented system. Moreover the environment allows to use any source of request stream; presented generator can be substituted by component translating real web service server logs or even use stream of requests from real network.

Future plans for development of simulation environment include implementation of new requests distribution algorithms to enlarge algorithms repository and reorganize structure of modelled system to general graph structure with more possible inputs and outputs of a simulated system.

Acknowledgements

The research presented in this paper has been partially supported by the European Union within the European Regional Development Fund program no. POIG.01.03.01-00-008/08.

References

[1] L. Borzemski , A. Zatwarnicka, K. Zatwarnicki, The framework for distributed web systems simulation, *Proc. of ISAT 2007, in: Information Technology and Web Engineering: Models, Concepts, and Challenges*, ed. L. Borzemski, Wrocław, 2007, pp. 17–24

[2] K. J. Brzostowski, J. P. Drapała, P. R. Świątek, J. M. Tomczak: Tools for automatic processing of users requirements in soa architecture, *Information Systems Architecture and Technology (ISAT'2009)* "Service oriented distributed systems: concepts and infrastructure", Szklarska Poręba, Poland, September 2009, pp. 137-146

[3] A. Grzech, P. Świątek: Modeling and optimization of complex services in service-based systems, *Cybernetics and Systems* , 40(08), pp. 706–723, 2009.

[4] M. Tüxen, I. Rüngeler, E. P. Rathgeb: Interface connecting the INET simulation framework with the real world, *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops* ICST, Brussels, Belgium, 2008, pp. 1–6.

[5] Z. Wang: Internet QoS: architecture and mechanisms for Quality of Service, *Academic Press*, 2001.

[6] Wang, G.; Chen, A.; Wang, C.; Fung, C.; Uczekaj, S.: Integrated quality of service (QoS) management in service-oriented enterprise architectures *Enterprise Distributed Object Computing Conference*, 2004. EDOC 2004. Proceedings. Eighth IEEE International Volume , Issue , 20-24 Sept. 2004 pp. 21 - 32