The problem of effective deployment architecture in SOA

A. WOŹNIAK, T. NOWICKI

adrian.wozniak@wat.edu.pl, tadeusz.nowicki@wat.edu.pl

Military University of Technology, Faculty of Cybernetics
Institute of Computer and Information Systems
Kaliskiego 2, 01-489 Warsaw, Poland

Service Oriented Architecture is popular in many organizations. In particular, it has already deeply rooted in large corporations that need to automate entire business processes and implement them in many systems. It has a unique feature that allows unambiguously indicate service that is to realise business process step. That indication is possible to show directly in BPMN diagram. Thus, it is possible to trace which server has used resources to implement the service and how much of those resources were needed. Therefore, it is possible to build an optimization task that, with limited and unreliable resources, will determine such allocation of components to servers and such an algorithm for assigning tasks to them, so that the processes will work as well as possible. The article presents a model of such an optimization task. This model consists of four layers. The Organization Layer describes the system environment – the types and frequency of initiating business process instances. The Integration Layer describes the business processes and indicates the services that should be performed at every step. The Component Layer describes component characteristics and what services they provide. In Server Layer both: server characteristics and runtime environments necessary for the component to run are described. Finally, the optimization task and evaluation criteria are formulated.

Keywords: SOA deployment, mathematical model, system architecture optimization.

DOI: 10.5604/01.3001.0013.6604

1. Introduction

The functioning of an organization, to some extent, can be reflected by the implementation of its business processes. Business processes [1], [2], [3] are understood as an ordered set of actions that are repetitive and aim at solving a certain problem. The more effectively they are implemented, the more effective the organisation as a whole is. Therefore, modelling, improvement and optimization of business processes is often the object of interest of the organization. One of the aspects of process optimization is shortening the time of their realization. One of the most effective methods of shortening the lead time is automation of processes, and in particular their computerization. Service Oriented Architecture (SOA) is a concept of software development, created for this purpose.

Service Oriented Architecture [4], [5], [6] is a way of designing IT systems, which is based on services. A service is an independent functional unit, which gives business value and is usually made available remotely using computer networks. From the SOA point of view, services are provided and called by the components. Components are programs that can be implemented in different technologies and on different servers. Each component can provide many services and can use other services. The diversity of technologies in SOA finds practical application, because often different technologies are used to meet the same business needs. In addition, companies often have systems developed before the introduction of SOA and produced in various technologies. Such older systems can also be regarded as components.

In order to implement a business process in SOA, it is necessary to cooperate with many components. Since they are manufactured in various technologies, an Enterprise Service Bus (ESB) bus is used as an intermediary. This bus mediates communication between components and, if necessary, adjusts the communication protocol to the needs of the component providing the service. In order to synchronize the operation of individual components and ensure the implementation of the business process, the so-called Business Process Management (BPM) engine is used in the SOA. Using this element, it is possible to implement business processes of the organization, which are realized in the system, and in particular to indicate the services, which are performed in order to realize the business process. In the literature, this issue was discussed mainly from three different points of view: scheduling of service calls, allocation of components to servers and selection of service instance (Service Composition).

The issue with the largest number of publications is the choice of service instance (called Service Composition). The problem that Service Composition solves is that the service is usually made available by different components or multiple instances of the same component on different servers and the best service instance should be selected according to some criteria (e.g. price, time, quality, etc.). Methods based on genetic algorithms are most often used to solve this problem [7], [8], [9], [10], [11]. Other known algorithms are also used, for example: Harmony Search in works [12], [13], The Artificial Bee Colony Algorithm in [14], Friut Fly Optimization Algorithm in [15], Bat Algorithm in [16], Binary Search Tree in [17] etc. You can also find a number of interesting original solutions in this area: [18], [19], [20], [21], [22]. The scheduling of service calls was dealt with in two articles. In [23] the method of scheduling tasks is presented, i.e. determining the order of their completion once they reach the queue of service calls to be completed by the Component. The article uses a method based on the critical path of the process. Article [24] presents a more sophisticated method, where global QoS (Quality of Service) requirements are first set from a process point of view, and then local prioritisation of each task, depending on how far it is from not meeting the QoS requirement.

The issue of allocating instances of components to servers is also discussed in the literature. In [25] is presented a method that reduces the problem to the task of capacity planning and presents the algorithm of its solution. At work [26] the authors focus on optimizing the availability of resources by analysing the weak points in the distribution of components for servers. Article [27] presents a method for optimising the allocation of servers to components using a genetic algorithm, but it is incomplete as it does not show how to evaluate the solution. There are also methods that search for optimal solutions during the operation of the system and adjust the form of these solutions on an ongoing basis. Such a method is set out in [28] and [29]. On the other hand, [30] shows a method that combines the choice of allocation and the choice of scheduling algorithm. Proposed scheduling algorithms are well developed there. However, part of the allocation has been significantly simplified, for example, optimisation is limited to the choice of the number of homogeneous servers for established assumptions. So far, a comprehensive method of optimizing the allocation of components to servers and the selection of the scheduling algorithm in SOA has not been presented. Therefore, the purpose of this article is to present the model of SOA system, and idea of optimization of such allocation.

2. Operating model of the SOA system

Systems in SOA can be presented from two perspectives: organization and system, as shown in Figure 1. The area of organization represents the environment of the SOA system that stimulates it. This environment is responsible for initiating business processes in the system. In the system area you can specify three layers (from the lowest level): Integration, Component and Server Layer. Integration Layer represents the central element of SOA, which is an intermediary in calling all services. This layer stores information about all business processes carried out in the system and controls when to call which service to execute the process. The Component Layer represents the software that provides services. Each step in the business process is to launch a service in the system. The service may be provided by one or more components, so it is important to select the best service provider. Each component needs resources – the server and its computing power and memory – for its operation. An example of the presented concepts is shown in Figure 2. The depicted business process can be realized by an employee or can be automated with the use of SOA class system. The customer uses the user interface to complete the credit application. The application goes to the business process engine, which is connected to the ESB and starts the instance of the business process of handling the application. The engine runs subsequent services via a bus until the process is completed, which ends with a message being sent to the customer. In this example you can show that one “Send email to client” service is not a copy of the “Send contract” step, but you can still connect them.

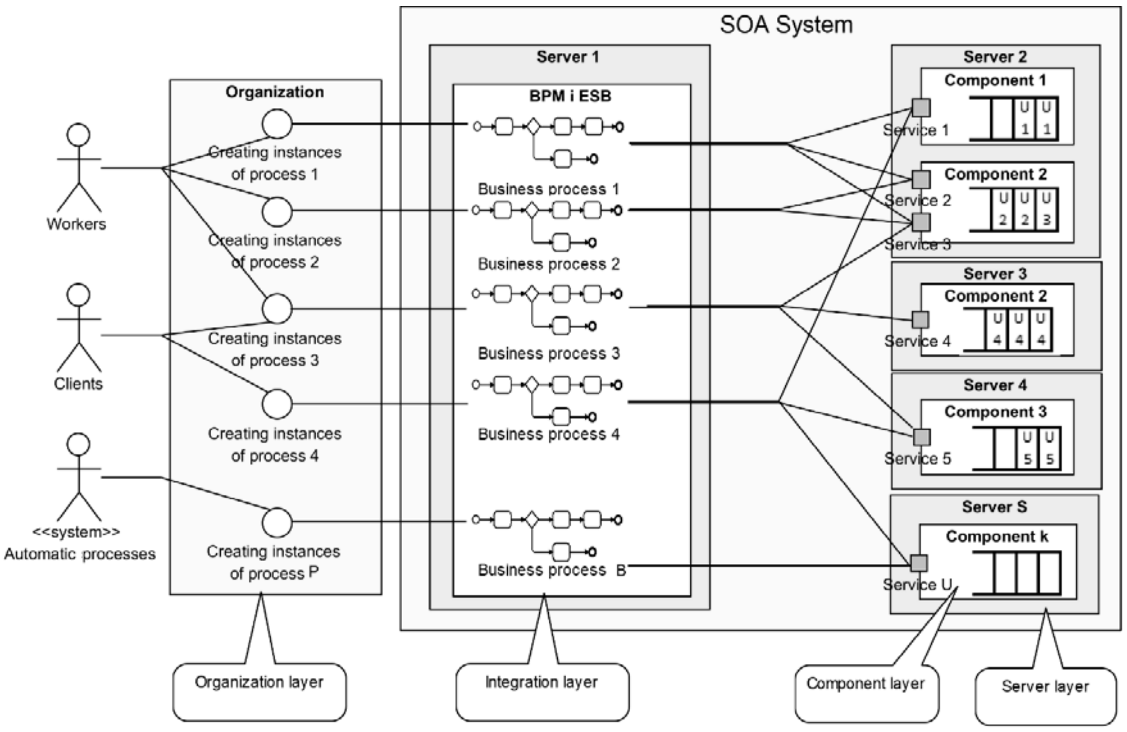

Fig. 1. Areas and layers of the SOA system model (Source: own study)

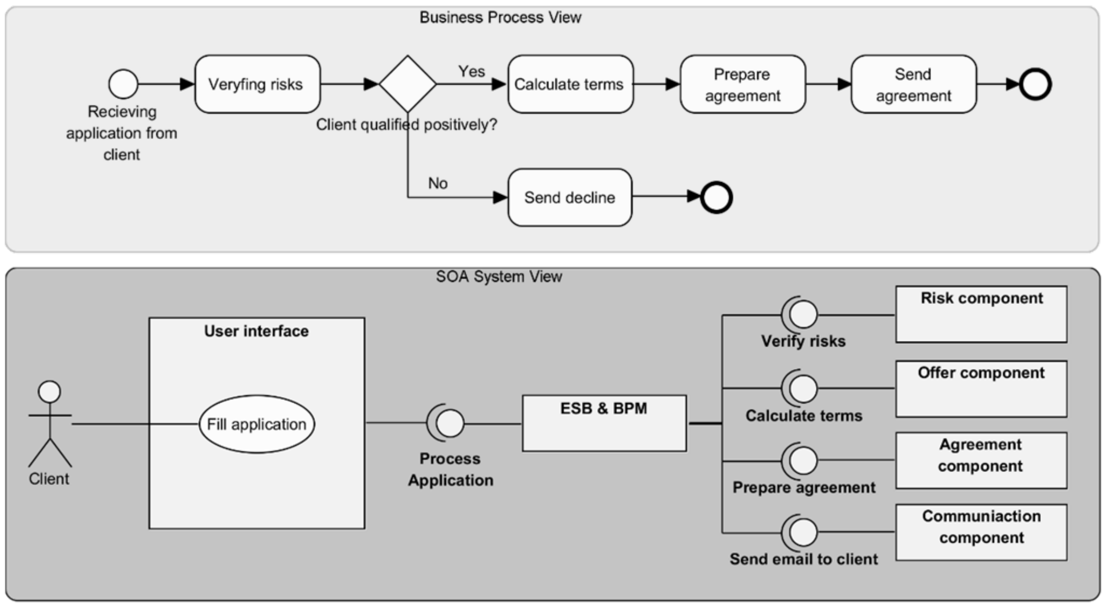

Fig. 2. Example of a business process implemented in SOA (Source: own study)

This is a frequently used procedure to maintain the universality and adaptability of services. In this case, the "Send email to client" service is generic and can be used in any business process in which you need to send an email message to the client. Different instances of the process would transfer different parameters to the service, such as title, content and attachments. In monolithic systems without SOA, optimization of their operation is focused on criteria concerning individual systems, such as: system response time or probability of system suitability. Optimization of the operation of each system individually rarely provides a solution that is optimal globally from the point of view of the entire business process. The advantage of system modelling in SOA is the possibility of unambiguous assignment of business process steps to services provided by the system. This makes it possible to adopt a different criterion for system optimisation. The described connection of processes with services enables the adoption of optimization criteria referring to business processes, such as: cost or time of business processes execution, probability of not executing the process in the required time, etc. The above mentioned connection enables the adoption of optimization criteria referring to business processes, such as: cost or time of business processes execution, probability of not executing the process in the required time, etc. With fixed hardware resources (processors, RAM on servers), different distribution of many components on servers will give different values of their performance parameters, and thus a different business value for the organization. Algorithms for selecting a service instance will also work better with different allocations of components to servers. It is therefore worth looking for the best possible allocation of components to servers and algorithms. Since SOA enables direct assignment of resources (used by the service) to the business process, it is possible to determine the optimal solution from this point of view.

3. Mathematical model of the problem

Further, three types of sets of numbers will be used to describe the model:

$\mathcal{R}$ – a set of real positive numbers,

$\mathcal{N}$ – a collection of natural numbers excluding zero,

$\mathcal{B} = \{0,1\}$ – binary collection.

3.1. Organisational Layer Model

The organization in which the SOA system operates is the initiator of business processes launched in the system. So, let:

$$\mathbb{B} = \{1, 2, 3, ..., b, ..., B\} \qquad (1)$$

denote a set of process type numbers in the organization. Each business process type can have many instances. E.g.: the sales process (type, class) can be executed many times in parallel (it can have many instances).

Intervals between business process initiations are different for instance. Due to different sources and the nature of processes, the distribution of a random variable defining the intervals between process starts is different for each process type. If the intervals between business process start-ups are random variables, then a sequence of these random variables is a stream of events that iniciate business process instance. Therefore, let

$$\overline{T} = \left[\overline{T}_b\right]_B \qquad (2)$$

where $\overline{T}_b$ mean a random variable being the length of time between successive instances of the b-type process. Then the function

$$H(t) = \left[H_b(t)\right]_B \qquad (3)$$

denote the vector of functions $H_b(t)$ being the expected number of running instances of this type of business process until t. In order to launch another instance of the business process, the previous instance does not have to be completed. In other words, one business process may have multiple instances at the same time.

3.2. Model of Integration Layer

The second area is the system. The first layer to be described will be the Integration Layer. It is the layer in which business process are realized using BPM engine. Due to the fact that engines control calls to services provided in the system, they are often implemented together with the ESB bus (e.g. SAP PO, MS BizTalk). Inquiries starting the process may come from different sources. For processes initiated by employees or customers, the most common sources are user interfaces. For processes triggered by events (such as daily at 9:00 a.m.), the most common source is the engine itself, which tracks the processes and events taking place in the system. Business processes in the engine are most often implemented using Business Process Modelling & Notation (BPMN) or Business Process Execution Language (BPEL). Most engines support both languages. In the BPMN language, business process tasks (subsequent steps) are implemented by means of a task. There are several types of tasks. A special case of a task is the “Service task”, which represents the launch of a service.

Let

$$\mathbb{U} = \{1, 2, 3, ..., u, ..., U\} \qquad (4)$$

mean a set of numbers of all services in the system. Services represent steps which are “Service Task” of business process – in this way the BPMN notation presents a direct relation between business processes in the organization and the functionalities activated in the system.

Relationships within the business process are always directional (there is a predecessor and a successor), therefore in the graphical representation of the process only directed graphs will be used, which means that the only edges will be arcs. These processes are defined by a vector:

$$PB = \left[PB_b \right]_B \tag{5}$$

where

$$PB_b = \left\langle G, \left\{ \xi_1, \xi_2 \right\}, \left\{ \psi \right\} \right\rangle \tag{6}$$

whereby

$G = \left\langle W, \underline{U}, P \right\rangle$ is a directed graph and

W – is a set of vertices of the graph, with each vertex representing a step in the business process (service task),

$\underline{U}$ – is a set of arcs of the graph, where each arc means the ability to move from one vertex (step in the process) to another,

P – a three-part relationship $\left(P \subset W \times \underline{U} \times P \right)$ that defines the way vertices and arcs (predecessor and successor) are linked,

$\xi_1 : W \to \mathbb{U}$ – is the function that assigns the number of the service called at the apex,

$\xi_2 : W \to \left\{ FLOW, OR, XOR, AND \right\}$,

$\psi : \underline{U} \to \left[1, 0 \right]$ – is the function that assigns to the arc the probability of transition between steps (probability of path selection).

The function ξ_2 assigns the goal type to the top of the graph. The relation referred to in the model is a reflection of the “Sequence flow” relation in the BPMN notation and means moving from step to step. AND, OR and XOR gateways are an important element of the flow. These gateways are reflected in the model by a parameter and probability assigned to the relation. Depending on the type of gateway, the next steps should be interpreted differently:

- if in the process step the gate type is set to “FLOW”, it is a reflection of the lack of a gate and treated as a simple transition between steps. Then only one relation (arc) to another step (apex) is allowed;
- If the gateway type is set to “OR” in the process step, the system selects one or more relations randomly according to the probability of path selection;
- If the gateway type is set to “XOR” in the process step, the system selects one of the branches according to the probability of each relation. The sum of probabilities may not be greater than 1, but it may be less than 1 (then the difference between the sum of probabilities and 1 is treated as the probability of ending of the process after the gate);
- If the gateway type is set to “AND” in the process step, all subsequent steps should be started.

The probability defined by the function ψ is only relevant if the business step is followed by an XOR or OR gateway. From the point of view of optimization of process execution time, it is not important what question is asked at the gate, but the probability of choosing each of the paths. The model presents only those steps of the business process that are carried out in the system. This is due to the fact that:

- the presented method does not affect the time of the process steps carried out outside the system;
- time of realization of off-system steps does not influence the results of optimization (it is important if the time of realization of processes has been shortened, and not how long the process lasted).

The ESB service bus is a central element of the SOA and is an intermediary in all service calls in the system and so it has been modeled. Each time a service is started by an ESB, it also generates a load on the ESB itself. This burden is different for each of the services. In the model we assume that there is only one central element (BPM and ESB), however it can be launched on several servers. It is due to a fact that vast majority of companies avoid implementing two or more ESB technologies because of high cost of maintaining it. Services are accepted and provided by components according to Time Sharing model rules or queue (usually FIFO). On the ESB, the next element of the process (service call) for processing is uploaded to the next component queue after processing the previous element. The model does not take into account network traffic, but only message delay times. The bandwidth between servers is fixed.

An element that significantly influences the time of business processes execution is the used algorithm of task scheduling. This algorithm is executed on an ESB and decides:

- selection of the component and server that is to provide the service (when many components provide the service or when there is a component performance redundancy);
- sequence of tasks.

Therefore,

$$\mathbb{A} = \left\{ 1, 2, 3, ..., a, ..., A \right\} \tag{7}$$

will be a set of numbers of considered scheduling algorithms. It is not possible to verify all algorithms. Only selected, most popular algorithms will be analyzed. After optimization, best algorithm is selected which is $WA \in \mathbb{A}$.

3.3. Component Layer Model

As shown above, business processes in SOA class systems are defined as sequences of service calls. The services are provided by components. Let

$$\mathbb{K} = \{1, 2, 3, ..., k, ..., K\} \tag{8}$$

denote the set of component numbers. Components are independent elements of the system, which are independent applications. They can be run on different servers and can be created using different technologies. Often SOA components are legacy monolithic systems, which have been modified so that they can share their functionality via services. Components perform tasks according to different algorithms, while the most popular ones will be considered further in their work. Therefore, let

$$K^Q = \left[K_k^Q \right]_K \tag{9}$$

where $K_k^Q \in \mathscr{B}$ – means a vector showing how tasks are handled by the component. Two ways of task handling are possible:

- Time Sharing (K_k^Q =1) – the most common query handling method (used for standard services, e.g. REST);
- Queue (K_k^Q =0) – a popular method of service execution, used in particular in asynchronous micro-services systems (where technologies such as Rabbit MQ, kafka are used). Usually it is implemented as FIFO queue.

Each Component provides services, so let

$$K^U = \left[K_{k,u}^U \right]_{KxU} \tag{10}$$

where $K_{k,u}^U \in \mathscr{B}$ means the binary array element showing the allocation of services to components. The number 1 at the intersection of the k-th row and the u-th column shows that the component with the k number provides the service number u. The same services can be provided by several components, and each component can be run multiple times on different servers for reasons of performance or reliability. Each component is assigned to the services it provides and launch environments in which it can operate. Regardless of whether a component provides services or not, it needs memory resources and computational power to work properly on the server.
Therefore, let

$$M^K = \left[M_k^K \right]_K \tag{11}$$

where M_k^K is the average percentage of the standard processor running time for the maintenance of the k-th component activity, and

$$P^K = \left[P_k^K \right]_K \tag{12}$$

where $P_k^K \in \mathscr{N}$ is the size of the RAM needed for correct functioning of the k-th component.

In the model it is not possible to run the same component several times on one server. This would require sharing server resources between two identical components, which eliminates the benefit of redundancy. Components that handle tasks in Time Sharing model already have functionality to invoke new threads for new service invocations. And if software architect decides to implement queue in component it is to avoid multiple threads of same service on the same server. Still it is possible to run same component on multiple servers. In the system model, only those services that are called within the business processes are considered.

Resources are used not only for the proper functioning of components and execution environments, but above all they are used to provide services. Each service is therefore described by the CPU and memory load it generates both for its realization (i.e. on the server where the component is running), but also for its handling in the BPM and Integration Layer. In order to be able to take into account delays at the network level, services are also described by the amount of data needed for transmission in the network. Therefore, let

- $M^U = \left[m_u^U \right]_U$ mean the vector of the average time needed to perform the u-th service on a standard processor;
- $P^U = \left[p_u^U \right]_U$ mean the vector of average size of memory needed to provide the u-th service;
- $D^U = \left[d_u^U \right]_U$ mean the vector of average size of data needed for the transfer in order to launch the u-th service;
- $M^Z = \left[m_u^Z \right]_U$ mean the vector of average time needed to process the query of u-th service on ESB bus on a standard processor (e.g.: for transformations between communication standards);
- $P^Z = \left[p_u^Z \right]_U$ mean the vector of average size of memory needed to process an u-th service request on a bus.

The bus is a standard SOA component, which also takes up resources on the server. Therefore, let:

- $M^E \in \mathscr{R}$ mean the average fraction of time a standard processor takes for the ESB to function properly,
- $P^E \in \mathscr{N}$ mean the average size of memory required for the correct functioning of the ESB.

3.4. Server Layer Model

In the Server Layer, we identify the resources participating in the allocation at the level of detail necessary to determine the optimal allocation. Therefore, we identify servers with their basic performance parameters (processor power and memory size) and reliability. Therefore, let

$$\mathbb{S} = \{1, 2, 3, ..., s, ..., S\} \tag{13}$$

mean a set of server numbers, and

- $M^S = \left[m_s^S \right]_S$ is the vector of computational power of the processors expressed as the quotient of its power and the power of a standard processor;
- $P^S = \left[p_s^S \right]_S$ is the vector of the size of the server operating memory expressed in megabytes;
- $T_1^S = \left[T_{s,1}^S \right]_S$ is the vector of random variables that represent properly working time of the s-th server;
- $T_2^S = \left[T_{s,2}^S \right]_S$ denotes the vector of random variables that represent each subsequent renewal time of the server.

In the Server Layer, from the point of view of the discussed optimization, the network structure is not important. Within scope of our interest is only effective bandwidth between servers, so let

$$P^s = \left[P_{s_1,s_2}^s \right]_{SxS} \text{ where } P_{s_1,s_2}^s \in \mathscr{R} \tag{14}$$

mean a matrix showing the data transfer rates between servers. The value in row s_1 and column s_2 shows the transmission throughput from the server s_1 to the server s_2 expressed in Megabits per second.

Multiple components and ESB buses can be run on each server, so let

$$K^S = \left[K_{k,s}^S \right]_{KxS} \text{ where } K_{k,s}^S \in \mathscr{B} \tag{15}$$

mean a binary matrix that shows the allocation of components to servers. The number 1 at the intersection of row k and column s shows that component k is assigned to the server s. Also let

$$S^E = \left[S_s^E \right]_S \text{ where } S_s^E \in \mathscr{B} \tag{16}$$

mean a binary vector showing whether the ESB bus is running on the server or not.

Another important element discussed in the Server Layer is the execution environments. Let us assume that

$$\mathbb{S}^U = \{1, 2, 3, ..., s^U, ..., S^U\} \tag{17}$$

is a collection of numbers of execution environment. Launch environments in the model represent the software running on the server that is necessary for the proper functioning of the component. E.g.: for a Java Spring component, the boot environment may be e.g. the JBoss application server and the Debian Linux operating system. Including runtime environments in the model is very important, because one of the assumptions and basics of SOA is independence from technology and the ability to use older systems by making their services available on the ESB. In practice, however, systems written in different technologies require different execution environments. Too many execution environments on a single server can significantly reduce its performance. Therefore, it is one of the important elements of optimization:

- on the one hand, we will achieve greater efficiency of servers, if there is one runtime environment on each server and the components are grouped technologically;
- on the other hand, running more components on one server reduces the time needed to transfer data.

Let's assume that:

$M^{SU} = \left[m_{s^U}^{SU} \right]_{S^U}$ – is the vector quotients of the standard processor's operating time, which is occupied by the s^U starting environment for proper operation;

$P^{SU} = \left[p_{s^U}^{SU} \right]_{S^U}$ – is the vector of sizes of the memory needed for correct operation of s^U execution environment.

Let us assume that

$$S^K = \left[S_{s^U,k}^K \right]_{S^U xK}, \; (S_{s^U,k}^k \in \mathscr{B}) \tag{18}$$

is a binary array showing that the execution environment is able to handle the correct operation of component. The number 1 at the intersection of the s^U row and the k column shows that the s^U execution environment is able to handle the k component. Let us assume that

$$S^S = \left[S_{s^U,s}^S \right]_{S^U xS}, \; (S_{s^U,s}^S \in \mathscr{B}) \tag{19}$$

is a binary matrix that shows the allocation of runtime environments to servers. The number 1 at the intersection of the s^U row and the s column shows that the execution environment with the s^U number is assigned to the server with the s number. The model uses a fraction of the standard processor power as the server load unit. Such a unit gives the flexibility of modeling (independence from the manufacturer and processor type) and makes the model independent of technological changes over time.

3.5. Data, decision variables and target function

To the data set belong:

$$D = \left\{ \begin{matrix} B, \overline{T}, H(t), \mathbb{U}, PB, \mathbb{A}, \mathbb{K}, K^Q, K^U, M^K, \\ P^K{}_k, M^U, P^U, D^U, M^Z, P^Z, M^E, \\ P^E, \mathbb{S}, M^S, P^S, T_1^S, T_2^S, P^S, \\ K^S, S^E, \mathbb{S}^U, M^{SU}, P^{SU}, S^K, S^S \end{matrix} \right\} \qquad (20)$$

Decision variables are:

$$X = \left\{ K^S, S^E, WA \right\} \qquad (21)$$

The task is to find such values of decision variables that at the decision criteria are optimally met. Decision criteria are as followes:

$\overline{k}_1(t,X) = E\left(k_1(t,X)\right)$ – average expected time of the business process in the system, weighted by the expected number of instances in a given time t of system operation, where:

$$k_1(t,X) = \sum_{b=1}^{B} \left(\frac{H_b(t)}{\sum_{i=1}^{B} H_i(t)} \cdot \frac{\sum_{i=1}^{LR_b} CR_{i,b}(t,X)}{LR_b(t)} \right) \qquad (22)$$

$\overline{k}_2(t,X) = E\left(k_2(t,X)\right)$ – the expected variance of the time of execution of business processes weighted by the expected number of instances in the given time t of the system's operation, where:

$$k_2(t,X) = \sum_{b=1}^{B} \left(\frac{H_b(t)}{\sum_{i=1}^{B} H_i(t)} \cdot \frac{\sum_{i=1}^{LR_b} \left(\frac{\sum_{i=1}^{LR_b} CR_{i,b}}{LR_b} - CR_{i,b}(t,X) \right)^2}{LR_b(t)} \right) \qquad (23)$$

where:

$CR_{i,b}(t,X)$ – random variable denoting the execution time of the i-th instance of the b-th business process in time t;

$LR_b(t)$ – random variable denoting the number of instances of the b-th business process in time t;

$\overline{k}_3(t,X) = E\left(k_3(t,X)\right)$ – means the expected degree of utilization of allocated processor resources for the implementation of services at a given time t, where:

$$k_3(t,X) = \frac{\sum_{s=1}^{S} M_s tu_s(t,X)}{\sum_{s=1}^{S} \left(M_s \cdot Y(s)\right) \cdot t} \qquad (24)$$

and

$$Y(s) = \begin{cases} 1 \; where \; \sum_{k=1}^{K} KS_{k,s} + SE_s > 0 \\ 0 \qquad\qquad \text{otherwise} \end{cases} \qquad (25)$$

$tu_s(t,X)$ – random variable denoting the time spent on services by processors.

Only servers that have an assigned component or bus are taken into account. So if you don't use any server for the system, it will increase the value of this criterion function.

$\overline{k}_4(t,X) = E\left(k_4(t,X)\right)$ – means the expected rate of utilisation of allocated memory resources for the provision of services at time t, where:

$$k_4(t,X) = \frac{\sum_{s=1}^{S} \int_{i=0}^{t} pu_s(i,X)di}{\sum_{s=1}^{S} \left(M_s \cdot Y(s)\right) \cdot t} \qquad (26)$$

where $pu_s(t,X)$ – is a random variable denoting the size of memory occupied by services at time t. As before, only servers that have an assigned component or bus are taken into account.

Based on the above criteria, the target function can be defined:

$$\overline{k}(t,X) = \left(\overline{k}_1(t,X), \overline{k}_2(t,X), \overline{k}_3(t,X), \overline{k}_4(t,X)\right) \qquad (27)$$

It is worth noting that variables $CR_{i,b}(t,X), LR_b(t,X), tu_s(t,X), pu_s(t,X)$ cannot be determined analytically due to the algorithmic nature of the description of business processes and random characteristics of the environment (system), the status of which changes over time, and which affects the values of these variables.

4. The concept of SOA system optimization

Bearing in mind the mathematical model presented in the previous chapter, it is possible to formulate the task of optimizing the allocation of components to servers in the SOA and selecting algorithms indicating the component instance for the next service instance. Unfortunately, it is not possible to use classic optimization methods to find an optimal solution for several reasons:

- due to the random nature of SOA system operation, e.g.: business process instances appear at different intervals, calculations required for service provision;
- random nature of server reliability – server operation and repair times are random variables;
- due to the algorithmic nature of the course of the SOA functioning – an algorithm tracking the course of the process is required.

Therefore, a heuristic method of finding an optimal solution is proposed, consisting of three complementary parts: a review algorithm, a genetic algorithm and a simulator.

Figure 3 presents a proposed procedure aimed at facilitating the management of the SOA deployment architecture, in particular to ensure optimal: allocation of servers to components, scheduling algorithm and service selection algorithms. Before starting the procedure, it is necessary to collect the information described in the model. Most of them are obtained at different stages of the life cycle of the system regardless of the method presented. Before implementation, the load generated by services and components on different hardware configurations is tested during performance tests. After deployment, monitoring systems often collect information about the processing time of service calls. Less frequently, information is obtained about the intensity of business process launches before they are implemented in SOA. This could happen if there was no data in other systems, from which one could deduce that information. In the absence of this information, these parameters are usually estimated using the expert method to estimate the computing power and memory needed for the equipment and then adjusted if necessary. With this data you can run an algorithm that will find the best solution. The full overview algorithm verifies all combinations of scheduling and service selection algorithms. This is possible because their number is usually small. However, it is not possible to have a complete overview of the allocation of components to servers due to the incomparably higher number of possible combinations. However, a genetic algorithm is a perfect fit here because of the ease with which it is possible to save the allocation of components to servers as a genotype. Genotype is a binary vector, which, in the case of the presented solution, was created by transforming the binary matrix into a vector. The binary matrix from which the vector (genotype) was created represents the allocation of components to servers. Value 1 in the i-th row and j-th column means that the i-th component has been allocated to the j-th server, and 0 means no allocation. Each value contained in the matrix is called a gene. A set of genotypes in a genetic algorithm is called a population. The first step of a genetic algorithm is to generate an initial population of solutions by introducing genotypes with random values into the population. The most difficult element is the next step – evaluation of the solution. For this purpose, a simulator is used, which allows you to estimate the value of the criterion function, for each allocation (genotype). Once the assessment has been carried out, it will be known whether or not a better solution has been achieved. Depending on the end criterion chosen, this may result in the termination of the genetic algorithm. Such a stop condition could be for example: 1000 iterations without improving the result. If the stop condition is not met, then selection is carried out, which means that the solutions are ranked according to the function of the criterion, and the worst ones are rejected. In place of the rejected, new ones are introduced. They are generated by crossing genotypes of the best solutions. The higher the place on the list of best solutions, the greater is the chance to participate in the generation of a child.

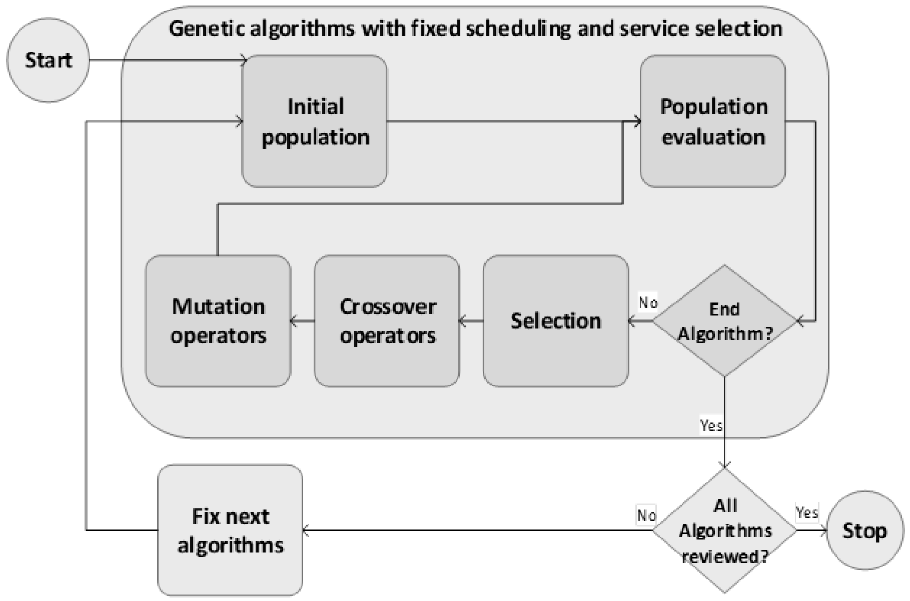

Fig. 3. Architecture management procedure (Source: own study)

In crossbreeding, two new genotypes are always created, each of which is a copy of a randomly selected genomes of a parent. The last part of the genetic algorithm is a mutation, i.e. a change in the value of randomly selected genes in order to broaden the range of reviewed solutions. These steps are carried out in the loop until the stop condition is met. The use of 4 evaluation criteria makes it necessary to apply multi-criteria optimisation methods. Depending on the chosen method, the result will consist of one solution (e.g. weighted criteria, hierarchical, mini-max method) or multiple (pareto-optimal set) after the algorithm is finished.

The method is complemented by the deployment of a solution and setting the criteria by which the genetic algorithm will be restarted. This is necessary because the environment in SOA is constantly changing. Both the intensity of running individual processes may change and the characteristics of the services themselves, e.g.: larger customer orders may cause the average load generated by the service instances to increase. Usually standard monitoring tools (e.g. Logstash in combination with Elasticsearch) are able to obtain this information. If a sufficiently large (in excess of pre-determined criteria) change in the model is detected, genetic algorithm optimisation of the allocation should be carried out again. Such a method allows to keep the solution close to the optimal one in terms of the proposed criteria.

5. Conclusions

SOA enables the introduction of new criteria in the optimization of the system deployment architecture. This criterion is the operation of the business process. It is possible thanks to unambiguous mapping from a step in the business process to a service made available by the component, which further allows to estimate the load to be transferred to the server. However, the mathematical model of the problem is complex, multi-layered, contains random variables, and business processes are described in an algorithmic way. For these reasons it is not possible to prepare a deterministic algorithm to solve the problem. The characteristics of this model, however, allow to build a simulator and determine the value of criteria through simulation. Having a method of evaluation defined in this way, it is possible to prepare an optimization algorithm, e.g. evolutionary algorithm and prepare a method of its solution.

6. Bibliography

[1] Dumas M., Rosa M., Mendling J., Reijers H., *Fundamentals of Business Process Management*, Springer London, 1988.
[2] Hammer M., Champy J., *Reengineering the Corporation*, HarperCollins Publishers Inc., 1993.
[3] Gawin B., Marcinkowski B., *Symulacja Procesów Biznesowych*, Helion, 2013.

[4] Erl T., *Service Oriented Architecture: Concepts*: *Concepts, Technology and Design*, HarperCollins Publishers, 2005.

[5] Erl T., *SOA Principles of Service Design*, Prentice Hall, 2007.

[6] Roshen W., *SOA-Based enterprise integration*, Mc Graw Hill, 2009.

[7] Czarnul P., “Modelling, optimization and execution of workflow applications with data distribution, service selection and budget constraints in BeesyCluster”, *Computer Science and Information Technology* (IMCSIT), 2010.

[8] Xiang C., Zhao W., Tian C., Nie J., Zhang J., “QoS-aware, Optimal and Automated Service Composition with Users’ Constraints”, IEEE 8th International Conference, 2011.

[9] Liu Y., Wu L., Liu S., “A Novel QoS--Aware Service Composition Approach Based on Path Decomposition”, Services Computing Conference (APSCC), IEEE Asia-Pacific, 2012.

[10] Syu Y., FanJiang Y., Kuo J., Ma S., “Towards a Genetic Algorithm Approach to Automating Workflow Composition for Web Services with Transactional and QoS--Awareness”, IEEE World Congress, 2011.

[11] Ludwig S., “Clonal selection based genetic algorithm for workflow service selection”, Evolutionary Computation (CEC), IEEE Congress, 2012.

[12] Mohammed M., Chikh M., Fethallah H., “QoS-aware web service selection based on harmony search”, ISKO-Maghreb: Concepts and Tools for knowledge Management, 4th International Symposium, 2014.

[13] Esfahani P., Habibi J., Varaee T., “Application of Social Harmony Search Algorithm on Composite Web Service Selection Based on Quality Attributes”, Genetic and Evolutionary Computing (ICGEC), Sixth International Conference, 2012.

[14] Liu Z., Xu X., “S-ABC – A Service--Oriented Artificial Bee Colony Algorithm for Global Optimal Services Selection in Concurrent Requests Environment”, Web Services (ICWS), IEEE International Conference, 2014.

[15] Zhang Y., Cui G., Wang Y., Guo X., Zhao S., “An optimization algorithm for service composition based on an improved FOA”, *Tsinghua Science and Technology*, 20(1):90–99 (2015).

[16] Hashmi K., AlJafar H., Malik Z., Alhosban A., “A bat algorithm based approach of QoS optimization for long term business pattern”, 7th International Conference on Information and Communication Systems (ICICS), 2016.

[17] Oh M., Baik J., Kang S., Choi H., “An Efficient Approach for QoS-Aware Service Selection Based on a Tree-Based Algorithm”, Computer and Information Science, Seventh IEEE/ACIS International Conference, 2008.

[18] Wang Z., Xu F., Xu X., “A Cost-Effective Service Composition Method for Mass Customized QoS Requirements”, IEEE Ninth International Conference, 2012.

[19] Zhang G., Wang Z., “A QoS-Aware Service Composition Optimization Based on Logical Structure”, CiSE International Conference, 2009.

[20] Pan S., Mao Q., “Semantic Web Service Composition Planner Agent with a QoS--Aware Selection Model”, Web Information Systems and Mining, 2009.

[21] Luo Y., Qi Y., Shen L., Hou, D., Sapa C., Chen Y., “An Improved Heuristic for QoS-Aware Service Composition Framework”, High Performance Computing and Communications, 10th IEEE International Conference, 2008.

[22] Wang C., Wang S., Chen H., Huang C., “A Reliability-Aware Approach for Web Services Execution Planning”, Services, IEEE Congress, 2007.

[23] Dyachuk D., Deters R., “Service Level Agreement Aware Workflow Scheduling”, Services Computing, IEEE International Conference, 2007.

[24] Dyachuk D., Deters R., “Ensuring Service Level Agreements for Service Workflows”, Services Computing, IEEE International Conference, 2008.

[25] Zhang C., Chang R., Perng C., So E., Tang C., Tao T., “An Optimal Capacity Planning Algorithm for Provisioning Cluster-Based Failure-Resilient Composite Services”, Services Computing, IEEE International Conference, 2009.

[26] Xie L., Luo J., Qiu J., Pershing J., Li Y., Chen Y., “Availability weak point analysis over an SOA deployment framework”, Network Operations and Management Symposium, 2008.

[27] Mennes R., Spinnewyn B., Latré S., Botero J., “GRECO: A Distributed Genetic Algorithm for Reliable Application Placement in Hybrid Clouds”, 5th IEEE

International Conference on Cloud Networking, 2016.
[28] Schmid M., “An approach for autonomic performance management in SOA workflows”, Integrated Network Management, IFIP/IEEE International Symposium, 2011.
[29] Almeida J., Almeida V., Ardagna D., Francalanci C., Trubian M., “Resource Management in the Autonomic Service-Oriented Architecture”, Autonomic Computing, IEEE International Conference, 2006.
[30] Huang K., Lu Y., Tsai M., Wu Y., Chang H., “Performance-Efficient Service Deployment and Scheduling Methods for Composite Cloud Services”, IEEE/ACM 9th International Conference on Utility and Cloud Computing, 2016.

Problem wyznaczania efektywnej architektury wdrożeniowej SOA

A. WOŹNIAK, T. NOWICKI

Architektura SOA jest popularna w wielu organizacjach. Została głęboko zakorzeniona szczególnie w dużych organizacjach, które muszą zautomatyzować procesy biznesowe i wdrożyć je w wielu systemach. Posiada ona unikalną cechę, która pozwalającą jednoznacznie wskazać usługę w systemie, która ma realizować krok procesu biznesowego. Wskazanie to można pokazać bezpośrednio na diagramie BPMN. W ten sposób możliwe jest śledzenie, który serwer wykorzystał zasoby do realizacji usługi i ile tych zasobów było potrzebnych. Zatem możliwe jest zbudowanie zadania optymalizacyjnego, które przy ograniczonych i zawodnych zasobach określi taki przydział komponentów do serwerów oraz taki algorytm przypisywania im zadań, aby procesy działały jak najlepiej. W artykule przedstawiono model takiego zadania optymalizacyjnego. Składa się on z czterech warstw. Warstwa organizacyjna opisuje środowisko systemowe – typy i częstotliwość inicjowania instancji procesów biznesowych. Warstwa silnika procesów biznesowych i ESB opisuje procesy biznesowe i wskazuje usługi, które powinny być wykonywane na każdym kroku procesu. Warstwa komponentów opisuje charakterystyki komponentów i przypisane do nich usługi. W warstwie serwerów opisano zarówno właściwości serwerów, jak i środowisk uruchomieniowych niezbędnych do poprawnego działania komponentów. Model zakończony jest sformułowaniem zadania optymalizacji i kryteriów oceny.

Słowa kluczowe: wdrożenie systemów typu SOA, model matematyczny, optymalizacja architektury systemów.