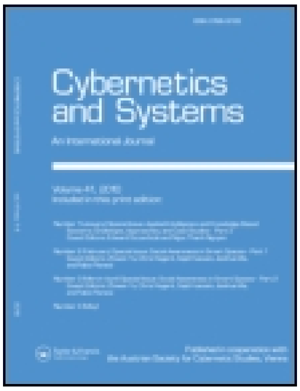

Cybernetics and Systems: An International Journal
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/ucbs20

TRANSLATIONS OF SERVICE LEVEL AGREEMENT IN SYSTEMS BASED ON SERVICE-ORIENTED ARCHITECTURES

Adam Grzech [a] , Piotr Rygielski [a] & Paweł Świątek [a]
[a] Institute of Computer Science, Wroclaw University of Technology, Wroclaw, Poland

Version of record first published: 25 Nov 2010

To cite this article: Adam Grzech, Piotr Rygielski & Paweł Świątek (2010): TRANSLATIONS OF SERVICE LEVEL AGREEMENT IN SYSTEMS BASED ON SERVICE-ORIENTED ARCHITECTURES, Cybernetics and Systems: An International Journal, 41:8, 610-627

To link to this article: http://dx.doi.org/10.1080/01969722.2010.520225

Cybernetics and Systems: An International Journal, 41:610–627
Copyright © 2010 Taylor & Francis Group, LLC
ISSN: 0196-9722 print/1087-6553 online
DOI: 10.1080/01969722.2010.520225

Translations of Service Level Agreement in Systems Based on Service-Oriented Architectures

ADAM GRZECH, PIOTR RYGIELSKI, and PAWEŁ ŚWIĄTEK
Institute of Computer Science, Wroclaw University of Technology, Wroclaw, Poland

The gain of the article is to introduce and to discuss a formal specification of a computer system's service level agreement (SLA) and its translation into the structure of complex services (composed of atomic services) delivering required functionalities and nonfunctionalities in a distributed environment. It is assumed that the SLA is composed of two parts specifying quantitative and qualitative requirements. The former requirements define the structure of the adequate complex services in form of a directed graph, where potential parallelism of atomic services performance may be taken into account. The latter—qualitative requirements—are applied to select the optimal complex service realization scenario; it is based on assumption that various atomic services distinguished in the complex services structures are available at the considered distributed environment in different versions and locations. Different versions of atomic services are atomic services delivering the required functionalities and satisfy nonfunctionalities at various levels. Different locations (installation place) of available atomic services means that the cost of atomic services delivery (communication and calculation) depends on parts of the distributed systems where the services are performed. The proposed model of SLA translation into complex services structures and variants may be applied—among others—to calculate upper and lower complex services' delivery times and to estimate the validity of possible parallelism in complex services.

KEYWORDS complex service optimization, graph parallelism, quality of service, service-oriented architecture, SLA

Address correspondence to Adam Grzech, Institute of Computer Science, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland. E-mail: adam.grzech@pwr.wroc.pl

INTRODUCTION

In systems based on a service-oriented architecture (SOA) paradigm, complex services are delivered by a distributed computer communication system as a composition of properly selected atomic services (Johnson et al. 1995; Milanovic and Malek 2004). Each atomic service provides certain and well-defined functionality and, moreover, is characterized by parameters describing quality of required and delivered service. Functionality of requested complex service is a sum of functionalities of ordered atomic services; the order is a consequence of complex service decomposition into simple tasks ruled by same time and data interdependencies. Besides requested functionality, the service requestor may specify certain nonfunctional properties of complex service formulated in terms of security, availability, cost, response time, quality measures, etc. Values of the nonfunctional parameters—given in the particular service level agreement (SLA; Keller and Ludwig 2003; Li et al. 2008)—of the requested complex service may be delivered without changes or should be negotiated before acceptance (Anderson et al. 2005). In order to deliver complex service with the requested functional and nonfunctional properties, appropriate atomic services must be chosen in the process of complex service composition.

One of the most important nonfunctional properties of services is service response time, which is a commonly used measure of the quality of service (QoS). Service response time is mainly influenced by three factors: execution time of atomic services, load of the system (number of requests present in the system), and communication delays on links connecting atomic services. In order to guarantee the service response time stated in the SLA all these factors must be taken into account in the process of service composition.

SLA TRANSLATION MODEL

It is assumed that new incoming *i*th complex service request is characterized by proper SLA description denoted by SLA_i. The SLA_i is composed of two parts describing the *i*th complex service request's functional and nonfunctional requirements, denoted respectively by SLA_{fi} and SLA_{nfi}. The first part characterizes functionalities that have to be performed, and the second contains values of parameters representing various required quality of service aspects (delivery times, security levels, guarantees, etc.).

The SLA_{fi} is a set of distinguished, separated, and ordered functionalities subsets:

$$SLA_{fi} = \{\Gamma_{i1}, \Gamma_{i2}, \ldots, \Gamma_{ij}, \ldots, \Gamma_{in_i}\} \tag{1}$$

where $\Gamma(i)$ ($\Gamma(i) = \Gamma_{i1} \cup \Gamma_{i2} \cup \cdots \cup \Gamma_{ij} \cup \cdots \cup \Gamma_{in_i}$) is a set of all functionalities necessary to complete the *i*th complex service request, $\Gamma_{i1} \prec \Gamma_{i2} \prec$

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

The proposed scheme covers all possible cases; n_i equal to 1 ($n_i = 1$) means that all required functionalities may be delivered in parallel manner, and m_j equal to 1 ($m_j = 1$ 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. It is assumed that the various versions of the distinguished atomic service offer exactly the same functionality and different values of nonfunctional parameters.

The nonfunctional requirements may be decomposed into separate subsets of nonfunctionalities in a manner similar to that introduced for functional requirements; that is,

$$SLA_{nfi} = \left\{\mathrm{H}_{i1}, \mathrm{H}_{i2}, \ldots, \mathrm{H}_{ij}, \ldots, \mathrm{H}_{in_i}\right\} \tag{2}$$

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

According to the above assumption the SLA_{fi} of the *i*th complex service request may be translated into ordered subsets of atomic services:

$$SLA_{fi} = \left\{\Gamma_{i1}, \Gamma_{i2}, \ldots, \Gamma_{ij}, \ldots, \Gamma_{in_i}\right\} \Rightarrow \left\{AS_{i1}, AS_{i2}, \ldots, AS_{ij}, \ldots, AS_{in_i}\right\} \tag{3}$$

where $\{AS_{i1}, AS_{i2}, \ldots, AS_{ij}, \ldots, AS_{in_i}\}$ is a sequence of atomic services subsets satisfying an order ($AS_{i1} \prec AS_{i2} \prec \cdots \prec AS_{ij} \prec \cdots \prec AS_{in_i}$) predefined by the order in the functionalities subsets $\Gamma_{i1} \prec \Gamma_{i2} \prec \cdots \prec \Gamma_{ij} \prec \cdots \prec \Gamma_{in_i}$ (Figure 1).

The order in sequence of atomic services subsets 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 the subset $AS_{i,\ j+1}$ 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} ($l = 1, 2, \ldots, l(k)$); $AS_{ij} = \left\{a_{ij1}, a_{ij2}, \ldots, a_{ijn_i}\right\}$. Moreover, it is assumed that any version a_{ijkl} ($l = 1, 2, \ldots, l(k)$) of the particular a_{ijk} ($a_{ijk} = \{a_{ijk1}, a_{ijk2}, \ldots, a_{ijkl(k)}\}$) atomic services (for $k = 1, 2, \ldots, m_j$) assures

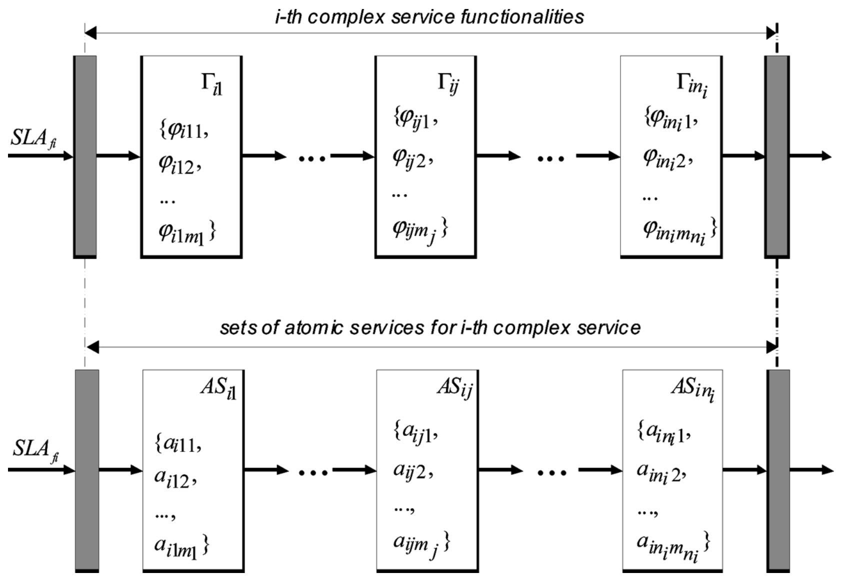
i-th complex service functionalities
SLA_{fi}
Γ_{i1} $\{\varphi_{i11}, \varphi_{i12}, \ldots \varphi_{i1m_1}\}$
Γ_{ij} $\{\varphi_{ij1}, \varphi_{ij2}, \ldots \varphi_{ijm_j}\}$
Γ_{in_i} $\{\varphi_{in_i1}, \varphi_{in_i2}, \ldots \varphi_{in_im_{n_i}}\}$
sets of atomic services for i-th complex service
SLA_{fi}
AS_{i1} $\{a_{i11}, a_{i12}, \ldots, a_{i1m_1}\}$
AS_{ij} $\{a_{ij1}, a_{ij2}, \ldots, a_{ijm_j}\}$
AS_{in_i} $\{a_{in_i1}, a_{in_i2}, \ldots, a_{in_im_{n_i}}\}$

FIGURE 1 Transformation of functionalities into sets of atomic services.

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 the lth version of the kth atomic service ($a_{ijkl} \in AS_{ij}$), the following conditions are satisfied:

$$fun(a_{ijkl}) = \varphi_{ijk} \quad \text{for } l = 1, 2, \ldots, m_k,$$

$$nfun(a_{ijkl}) \neq nfun(a_{ijkr}) \quad \text{for } l, r = 1, 2, \ldots, m_k \text{ and } l \neq r.$$

The ordered functionalities subsets SLA_{fi} determine the possible level of parallelism at the ith required complex service performance (in the particular computation and communication environment). The parallelism level $l_p(i)$ for the ith required complex service is uniquely defined by the maximal number of atomic services that may be performed in a parallel manner at distinguished subsets of functionalities (SLA_{fi}); that is,

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

The possible level of parallelism may be utilized or not in designing a performance scenario for the required ith complex service (Stefanovic and Martonosi 2000). Based on the above notations and definitions, two extreme required ith complex service compositions may be defined: the first extreme composition fully utilizes possible parallelism (available due to computation and communication resources parallelism), and the second extreme composition means that the required functionalities are delivered one by one (no computation and communication resources parallelism). The above presented discussion may be summarized as shown in Figure 2.

The known functional requirements SLA_{fi} may be presented as a sequence of subsets of functionalities, where the size of the mentioned latter subsets depends on the possible level of parallelism. The latter defines a set of possible performance scenarios (graphs) according to which the required complex service may be delivered. The space of possible solutions is limited—from one side—by the highest possible parallelism and—from the other side—by the simplest complex service realization scenario, according to which all the required atomic services are performed one by one in sequence.

The above-mentioned extreme compositions determine some set of possible *i*th required complex service delivery scenarios. The possible

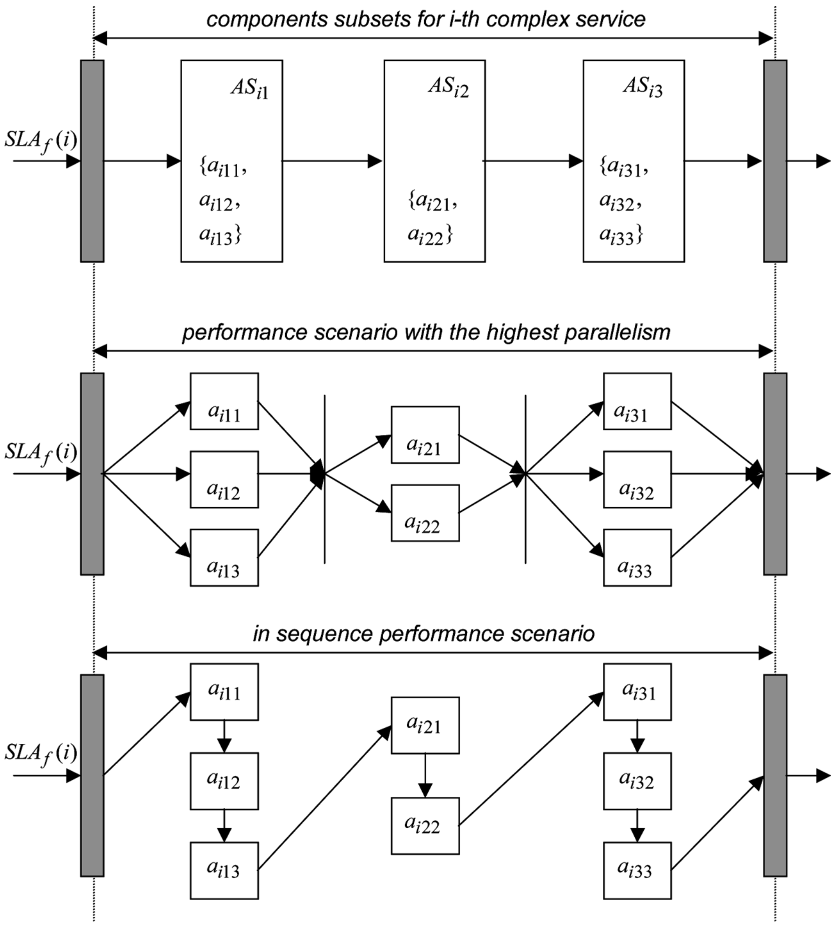

FIGURE 2 Two extreme performance scenarios for required complex service.

scenarios can be represented by a set of graphs GB_i—nodes of the graphs represent particular atomic services assuring the ith required complex service functionalities, and the graph edges represent an order according to which the atomic services functionalities have to be delivered.

The process of obtaining a complex service execution scenario graph is explained in detail in the next section.

PREREQUISITES FOR DETERMINING A COMPLEX SERVICE EXECUTION SCENARIO

The lth ($l = 1, 2, \ldots, L$) complex service $cs(l)$ is a response for a call for service fully described by the appropriate SLA denoted by the ith $SLA_i = (SLA_{fi}, SLA_{nfi})$, which contains information about functionality (SLA_{fi}) required by the user and nonfunctional requirements (SLA_{nfi}) determining the required level of the quality of service. The functional part of $SLA_{fi} = (\Phi_i, R_i)$ is a subject of the structure composition process, which delivers a set of atomic functionalities $\Phi_i = \{\varphi_{i1}, \ldots, \varphi_{in_i}\}$ present in system and defines the allowed order of execution of the required functionalities, with use of precedence relations $\prec$, given in matrix R_i.

The discussed precedence matrix describing a set of precedence relations may be—in the simplest case—described by a square binary order constraints matrix R_i with size $(n_i + 2) \times (n_i + 2)$. The R_i matrix defines which functionalities are bound with relation of precedence.

An exemplary order constraints (precedence) matrix is presented below:

$$R_i = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5}$$

Matrix R_i has a dimension 5×5, which corresponds to three atomic functionalities with the addition of abstract starting and ending functionality $\Phi_i = \{\varphi_{is}, \varphi_{i1}, \varphi_{i2}, \varphi_{i3}, \varphi_{ie}\}$. The matrix in this example should be understood as follows:

- Column 1—Functionality φ_{is} (abstract start functionality) should not be preceded by any other functionality.
- Column 2—Functionality φ_{i1} can be preceded by functionality φ_{is} or φ_{i3}.
- Column 3—Functionality φ_{i2} can be preceded by functionalities φ_{i1} or φ_{i3}.
- Column 4—Functionality φ_{i3} can be preceded by functionalities φ_{is}, φ_{i1}, or φ_{i2}.

- Column 5—Functionality φ_{ie} (abstract end functionality) can be preceded by functionalities φ_{i2} or φ_{i3}.

Order constraints given with matrix R_i can be transformed into description using the precedence relation $\prec$ as follows:

$$\begin{aligned} &\varphi_{is} \prec \{\varphi_{i1}, \varphi_{i3}\};\ \varphi_{i1} \prec \{\varphi_{i2}, \varphi_{i3}\};\ \varphi_{i2} \prec \{\varphi_{i3}, \varphi_{ie}\}; \\ &\quad \varphi_{l3} \prec \{\varphi_{l1}, \varphi_{l2}, \varphi_{le}\};\ \varphi_{ie} \prec \emptyset \end{aligned} \tag{6}$$

In general, binary 1 in the *k*th row and in the *j*th column ($r_{i(k,j)} = 1$) means that functionality φ_{ij} can be preceded by functionality φ_{ik}. Zero value in the *k*th row and in the *j*th column ($r_{i(k,j)} = 0$) means that functionality φ_{ij} cannot be preceded by functionality φ_{ik}. Having an abstract start and abstract end is guaranteed by zeros in the $j = 0$ column and $k = n_i + 1$ row.

$$r_{i(k,j)} = \begin{cases} 1 & \text{if } \ \varphi_{ik} \prec \varphi_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Moreover, each row (except for the $k = n_i + 1$ row) has to have at least one 1 value, which guarantees the presence of exactly one end functionality. Additionally, a guarantee of having exactly one start functionality is determined by having at least one 1 in each column (except for the $j = 0$ column). The above assumptions can be summarized with the following formulas:

$$\underset{k \in \{0,\ldots,n_i\}}{\forall} \sum_{j=1}^{n_i+1} r_{i(k,j)} \geq 1 \tag{8}$$

$$\underset{j \in \{1,\ldots,n_i+1\}}{\forall} \sum_{k=0}^{n_i} r_{i(k,j)} \geq 1 \tag{9}$$

Having functionalities set $\Phi_i = \{\varphi_{i1}, \ldots, \varphi_{in_i}\}$ and order constraints matrix R_l gives one the ability to build a base graph denoted by GB_i. $GB_i = GB(SLA_{fi}) = GB(\{\Phi_i,\ R_i\}) = GB(VB_i,\ EB_i)$ is a graph defining the structure of complex service, where $VB_i = \{vb_{i1},\ vb_{i2}, \ldots, vb_{ik}, \ldots, vb_{in}\}$ is a set of vertices of a base graph (each vertex vb_{ik} corresponds to proper functionality φ_{ik}) and EB_i is a set of edges corresponding to the precedence relations defined by matrix R_i.

An exemplary graph for the R_i matrix (Eq. (5)) and functionalities set $\Phi_i = \{\varphi_{is},\ \varphi_{i1},\ \varphi_{i2},\ \varphi_{i3},\ \varphi_{ie}\}$ is presented in Figure 3. Each binary value 1 represents an edge between functionalities in graph GB_i.

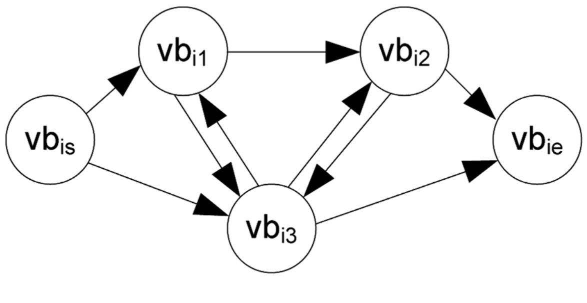

FIGURE 3 Graph GB_i representation for matrix R_i (Eq. (5)). Each binary 1 value in the matrix corresponds to an edge in structure graph GB_i. Redundant edges make the graph cyclic.

COMPLEX SERVICE SCENARIO COMPOSITION

The structure of complex service determines which atomic functionalities are delivered within it and what the order bounds are. Such service can be an object of an optimization process concerning the determination of the exact order of functionalities delivery and parallel execution manner. The result of processing complex service structure is called the *i*th complex service execution scenario.

The complex service execution scenario is a graph GC_i obtained in the scenario determination process. Scenario $GC_i = GC(SLA_{fi}) = GC(\{\Phi_i, R_i\}) = GC(VC_i, EC_i)$ is a graph containing vertices set $VC_i = VB_i = \{vb_{i1}, vb_{i2}, \ldots, vb_{ik}, \ldots, vb_{in}\}$, which are the same as in the service structure and edge set $EC_i = EB_i \backslash EA_i$, which is a subset of structure edge set EB_i:

$$EC_i \subseteq EB_i$$

$$EC_i \cup EA_i = EB_i$$

The problem of finding an optimal scenario can be formulated as follows:

Given:

- the *i*th complex service request given with SLA_i
- a graph GB_i with set of vertices VB_i and set of edges EB_i
- the order constraints matrix R_i
- an other parameters vector $\mathbf{a}_i$

Find:

Such an adjacency matrix R_{GC_i} (as a representation of graph GC_i) from a set of binary matrices of size $(n_i + 2) \times (n_i + 2)$ that minimizes the function f:

$$R^*_{GC_i} = \arg \min_{R_{GC_i}} f(R_{GC_i}; \mathbf{a}_i) \tag{10}$$

with respect to constraints:

- Graph GC_i represented by matrix R_{GC_i} should be acyclic.
- Graph GC_i represented by matrix R_{GC_i} should have exactly one abstract start and abstract end functionality:

$$\underset{k\in\{0,\ldots,n_i\}}{\forall} \sum_{j=1}^{n_i+1} r_{CG_{i(k,j)}} \geq 1 \tag{11}$$

$$\underset{j\in\{1,\ldots,n_l+1\}}{\forall} \sum_{k=0}^{n_i} r_{CG_{i(k,j)}} \geq 1. \tag{12}$$

- Matrix R_{GC_j} should comply with order constraints matrix R_i: $R_{GC_i} \otimes R_i = R_{GC_i}$.

Satisfaction of the order constraints can be determined via logical multiplication of each binary element of matrix. Operation $\otimes$ is defined as follows:

$$A \otimes B = C$$

$$\underset{k\in\{0,\ldots,n_i+1\}}{\forall}\ \underset{j\in\{0,\ldots,n_i+1\}}{\forall} : c_{kj} = \begin{cases} 1 & \text{if } a_{kj} = 1 \quad \text{and} \quad b_{kj} = 1 \\ 0 & \text{otherwise} \end{cases}$$

Equality $R_{GC_i} \otimes R_i = R_{GC_i}$ determines the satisfaction of order constraints. Function $f(R_{GC_i}; \mathbf{a}_i)$ can be any function determining the quality of the service level; for example, execution time, cost, security level, reliability, etc. Depending on the function relationship with quality (function value growth can mean a quality level increase or decrease), the optimization task might be maximization or minimization.

Determination of an optimal scenario consists in removing a subset of edges EA_i from the EB_l set in such way that each vertex in the resulting graph belongs to some path connecting the starting vertex and ending vertex in such way that input and output degrees of vertices $\{\varphi_{i1}, \varphi_{i2}, \ldots, \varphi_{in_i}\}$ are at least equal to 1. Moreover, the input degree of the starting vertex and the output degree of the ending vertex must be equal to 0. Additionally, the resulting graph GC_i representing a scenario must be acyclic.

The set of edges that are subject to be removed is uniquely defined by a result of subtraction of adjacency matrices $R^*_{A_i} = R_i - R^*_{GC_i}$. The remaining binary 1 values in the adjacency matrix $R^*_{A_i}$ define edges that are subject to be removed from graph GB_i to obtain an optimal complex service execution scenario GC^*_i.

For an exemplary matrix R_i (Eq. (5)) there are six possible scenario graphs GC_l. All are presented in Figure 4.

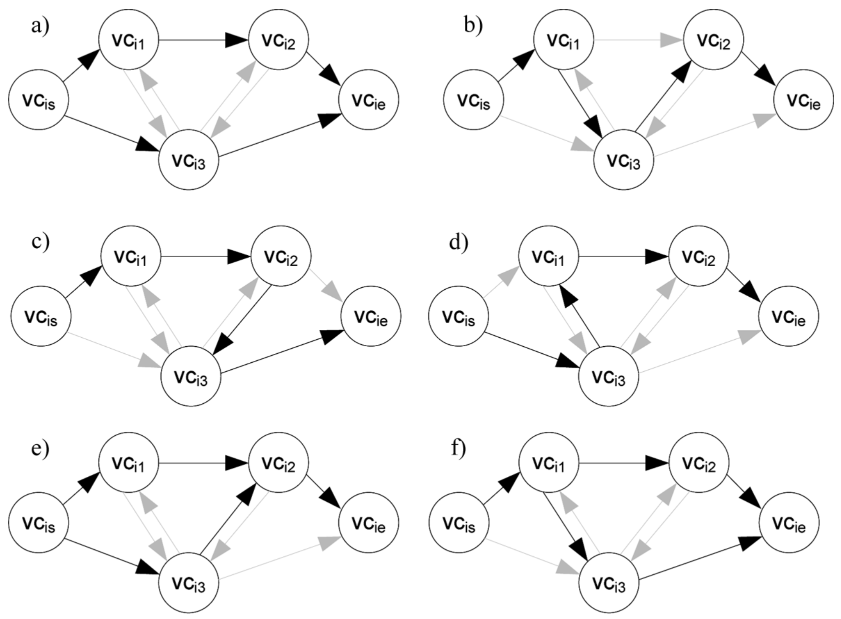

FIGURE 4 Six possible scenario graphs GC_i obtained for structure graph GB_i with order constraints given with matrix R_i (Eq. (5)). Black edges are edges from set EC_i, grey ones are from set EA_i.

The scenario presented in Figure 4(a) was obtained after substraction $R_{GC_{ia}} = R_i - R_{GA_{ia}}$:

$$R_{GC_{ia}} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

One can notice that the scenarios presented in Figure 4 differ in case of serial and parallel execution possibilities. All functionalities from scenarios (b), (c), and (d) are executed in series (one by one) so one may suspect that the quality expressed, for example, as execution time may be worse than in scenarios (a), (e), and (f), which use parallel execution. Parallel execution of functionalities shortens complex service execution time but utilizes more of the system resources than a serial execution scenario at a particular moment.

In the case of a serial scenario, we can estimate the time of execution of complex service using the following equation:

$$t(GC_{iserial}) = \sum_{k=1}^{n_i+2} t(vb_{ik}) + t(EC_i), \qquad (13)$$

where $t(EC_i)$ is an overall time of transferring requests between functionalities in a complex service scenario. If a transfer delay request between the kth and $(k+1)$th functionality is denoted as $t(ec_{lk,k+1})$, calculation of transfer times in the whole complex service with a serial scenario is as follows:

$$EC_i = \left\{ec_{ikj} : r_{CG_{i(k,j)}} = 1\right\} \tag{14}$$

$$t(EC_i) = \sum_{k=1}^{n_i+1} t\left(ec_{lk,k+1}\right). \tag{15}$$

In an extreme parallel scenario the above calculations are slightly different. Time of execution of a complex service is calculated as follows:

$$t\left(GC_{iparallel}\right) = \max_{k \in \{1,2,\ldots,n_i+2\}} \{t(vb_{ik}) + t(ec_{i1k}) + t(ec_{ikn+2})\} \tag{16}$$

Two extreme scenarios—with best and worst execution times—are presented in Figure 5.

The situation where the maximum the number of parallel functionalities delivery is limited should be considered. A need for the introduction of such a limitation may arise in the case where utilization of a larger number of parallel functionalities delivery can result in a decrease of end-user quality; for example, increases the cost of service to such a level that it violates the user's requirement.

In order to determine parallelism in an execution scenario there is a need to introduce a measure that will determine the parallelism level.

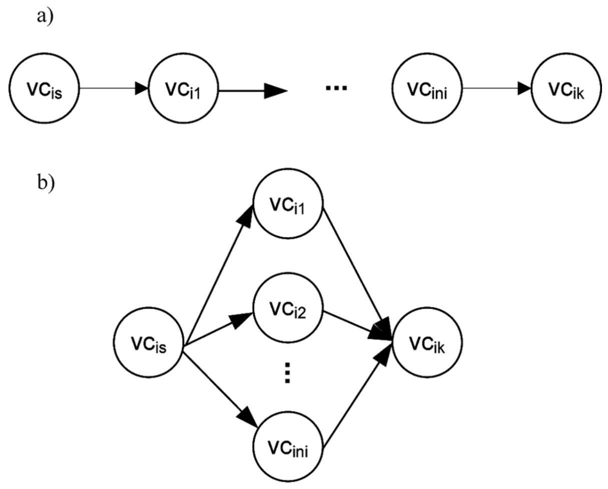

FIGURE 5 Two extreme scenario graphs. Serial scenario (a) gives the worst execution time and extreme parallel scenario (b) gives the best execution time.

In the literature, (Rau and Fisher 2000) the most popular parallelism level is an instruction-level parallelism (ILP), which measures the number of operations that can be done simultaneously divided by the number of all operations that should be performed (Eq. (17)):

$$l_{p_{ILP}} = \frac{n_{ipar}}{n_i} \tag{17}$$

where $l_{p_{ILP}}$ is an ILP measure and n_{ipar} is a number of operations that can be executed in a parallel manner.

For a scenario given by a graph, other parallelism measures can be used; for example, a measure that uses a mean weighted path length in the graph, where weights correspond to time needed to deliver proper functionality φ_{ik}. The proposed measure l_{pMP} can be defined as follows:

$$l_{pMP}(GC_i) = \left(\frac{1}{n_p} \sum_{k=1}^{n_p} p_l(k) \right)^{-1} \tag{18}$$

where n_p is a number of paths in graph GC_i and $p_i(k)$ is a weighted kth path length. Other parallelism measures can be found, for example, in Jain and Rajaraman (1994).

In order to optimize end-user quality there is need to introduce constraints concerning the parallelism level of execution scenario graph GC_i. When there are no constraints in SLA_{nfi} concerning QoS parameters—for example, cost—the formerly formulated problem can be used without change; otherwise, one should add a constraint concerning the parallelism level of scenario:

$$l_p(GC_i) = l_p(R_{GC_i}) \le l_{p\max},$$

where l_p function returns a parallelism level (with respect to chosen measure) for a graph or for an adjacency matrix that is a representation of a graph. This function should be designed in such a way that higher values mean delivery of a larger number of functionalities simultaneously, and smaller values mean that the scenario graph is mainly executed as an extreme serial scenario.

QUANTITATIVE ANALYSIS OF COMPLEX SERVICES REALIZATIONS

In general, the optimization task may be formulated (under the assumption that all scenarios originating from the GB_i graph deliver at least nonfunctional requirements) as follows:

$$SLA^*_{nfi} \leftarrow \max_{\{GC_{i1}, GC_{i2}, \ldots, GC_{is}\}} \left\{ \max_{a_{ijkl} \in a_{ijk} \in AS_{ij}} \{\mathrm{H}_{i1}, \mathrm{H}_{i2}, \ldots, \mathrm{H}_{in_i}\} \right\}.$$

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

$$SLA^*_{nfi}(GC_i) \leftarrow \max_{a_{ijkl} \in a_{ijk} \in AS_{ij}} \{\mathrm{H}_{i1}, \mathrm{H}_{i2}, \ldots, \mathrm{H}_{in_i}\},$$

where GC_i represents the selected *i*th required complex service scenario (composition).

The above formulated task means that the optimal versions of atomic services, determined by the selected *i*th required complex service performance scenario, should be selected. Such formulations of optimization problems can be found in Grzech et al. (2010) and Grzech and Swiatek (2009).

Any graph GC_{ip} ($GC_{ip} \in GB_i$ and $p = 1, 2, \ldots, s$) (the *p*th possible performance scenario for the required *i*th complex service) may be interpreted as a root graph for a set of realization graphs $\{G_{ipr}\}$ for $r = 1, 2, \ldots, R$, where R is a number of all different combinations of atomic services versions. For example, assuming that the are three distinguished groups of atomic services ($n_i = 3$) containing, respectively, 3, 2, and 3 atomic services ($m_1 = 3$, $m_2 = 2$, and $m_3 = 3$) with two versions of each, the example of root graph is presented in Figure 6; in the considered example the required complex service

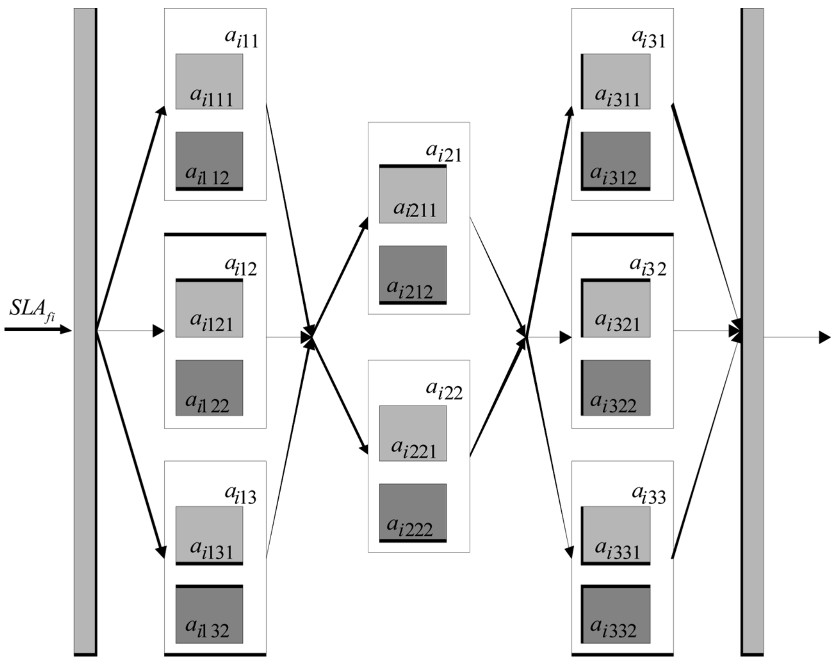

FIGURE 6 Root graph for the required complex service realization graphs.

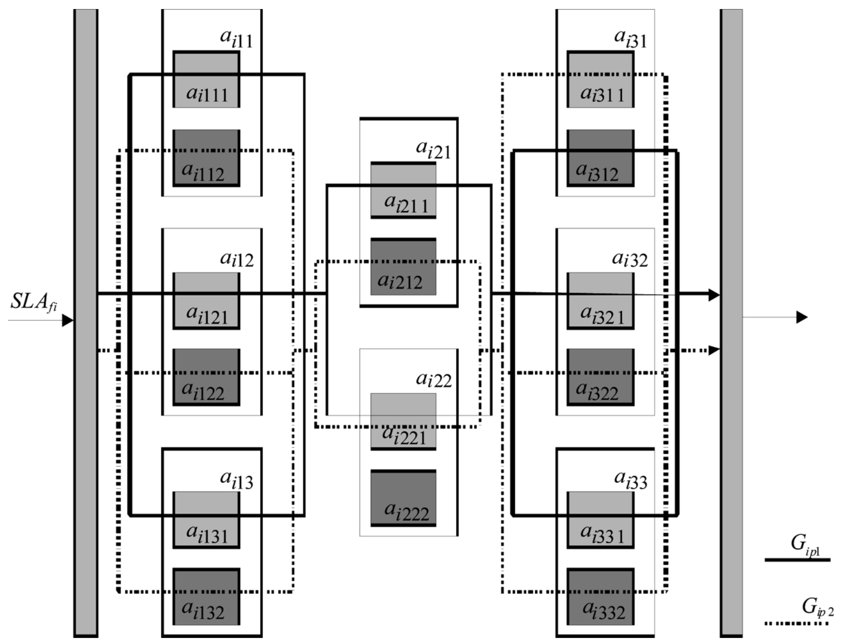

FIGURE 7 Two possible complex service realization graphs for root graph (Figure 6).

root graph is a source of 256 ($R = 256$) different ith complex service realization graphs with possible level of parallelism equal to 3 ($l_p(i) = 3$). The illustrative calculations simply show how the size of the complex services (measured by the number of atomic services necessary to complete the required complex service) influences the increase of the number of possible solutions.

Any two realization graphs (determined by a particular root graph defined by a particular complex service functionalities) in the set $\{G_{ipr}\}$ differ at least by one version of the set of atomic services a_{ijk} standing for the graph GC_{ip} nodes. Two possible realization graphs (G_{ip1} and G_{ip2}) for the above required complex service root graph are presented in Figure 7.

COMPLEX SERVICE DELIVERY TIMES AND RESOURCES UTILIZATION

The two distinguished extreme compositions of the ith required complex service—that is, based on the highest possible parallelism of atomic services (represented by the G_{ipar} graph) and based on the sequence of atomic services (represented by the G_{iseq} set of graphs)—can be used to estimate the limits of the ith required complex service completing times.

Let us denote by $d(G_{ipar})$ and $d(G_{iseq})$ the completion times for the two extreme compositions for the ith required complex service execution time. The introduced times (delays) satisfy the inequalities given below.

For the highest possible parallel composition of the *i*th required complex service (under the assumption that the computation and communication resources parallelism, available at the distributed system, does not introduce any complex service performance limits):

$$d(G_{ipar}) \leq \max_{k=1,2,\dots,m_1} com_{in}(a_{i1k}) + \max_{k=1,2,\dots,m_1} cal(a_{i1k}) + \sum_{j=1}^{n_i-1} \left\{ \max_{k=1,2,\dots,m_j;l=1,2,\dots,m_{j+1}} com(a_{ijk}, a_{i,j+1,l}) + \max_{l=1,2,\dots,m_{j+1}} cal(a_{i,j+1,l}) \right\} + \max_{l=1,2,\dots,m_{n_i}} com_{out}(a_{in_il}), \tag{19}$$

where $com_{in}(a_{i1k})$ is a communication delay between the *i*th complex service input interface and the *k*th ($k = 1, 2, \dots, m_1$) atomic service from the A_{i1} set; $cal(a_{i1k})$ is a calculation time of the *k*th ($k = 1, 2, \dots, m_1$) atomic service from the A_{i1} set; $com(a_{ijk}, a_{i,j+1,l})$ is a communication delay between the *k*th ($k = 1, 2, \dots, m_j$) and *l*th ($k = 1, 2, \dots, m_{j+1}$) atomic services belonging to two separate, neighboring, and ordered sets of atomic services A_{ij} and $A_{i,j+1}$, respectively; $cal(a_{ijk})$ is a calculation time of the *k*th ($k = 1, 2, \dots, m_j$) atomic service from the A_{ij} set; and $com_{out}(a_{in_ik})$ is a communication delay between the *k*th ($k = 1, 2, \dots, m_{n_i}$) atomic service from the A_{in_i} set and the *i*th complex service output interface.

For sequence composition of the *i*th required complex service (under the assumption that the computation and communication resources parallelism does not exist and the complex service is composed of atomic services performed one by one):

$$d(G_{iseq}) = \max_{k=1,2,\dots,m_1} com_{in}(a_{i1k}) + \sum_{j=1}^{n_i} \left(\sum_{k=1}^{m_j} cal(a_{ijk}) + \left((m_j - 1) \max_{l,u \in \{1,2,\dots,m_j\}, l \neq u} com(a_{ijl}, a_{iju}) \right) \right) + \max_{k=1,2,\dots,m_{n_i}} com_{out}(a_{in_ik}), \tag{20}$$

where $com(a_{ijl}, a_{iju})$ is a communication delay between successive atomic services ordered according to the given G_{iseq}.

It is rather obvious that the above given expressions for $d(G_{ipar})$ and $d(G_{iseq})$ present, respectively, lower and upper *i*th complex service completion (delivery) times and, moreover, that the compared times always satisfy the following inequality: $d(G_{ipar}) \leq d(G_{iseq})$.

The value of $d(G_{iseq}) - d(G_{ipar})$, which is always nonnegative, may be considered as a simple measure of parallelism attractiveness at the *i*th complex service's completing procedures. If the distance between the two considered completion times is relatively high, the parallel processing of the selected atomic services is worth organizing and deploying. If the distance between the two compared values is small, then all efforts necessary to organize and perform atomic services in a parallel manner are not compensated by noticeable reduction of the ith complex service completion time.

The value $d(G_{iseq}) - d(G_{ipar})$, that is, parallelism attractiveness, strongly depends on possible parallelism in the ith complex service completion procedures; that is, level of resources parallelism (the latter is measured by the highest number of atomic services that can be computed in parallel). Assurance of the highest possible level in computing atomic services in a parallel manner is possible if, and only if, a required level of resources distribution is obtainable. This means that the possible parallelism attractiveness should be compared with the efficiency of the required resources utilization.

The above discussed $d(G_{iseq}) - d(G_{ipar})$ value may be applied both as a simple measure of resource utilization in a distributed environment as well as for distributed environment design purposes. The gain of the former application is to evaluate resource utilization factors reflecting the validity of the distributed environment (especially available parallelism) for required complex services realization purposes. The aim of the latter is to predict the required level of resources parallelism for a known set of complex services (given by a known set of complex services SLA).

EXPERIMENTAL STUDY

In the former analysis a lower and upper bound for service completion time (represented as a graph) were introduced (Eq. (19) for parallel and Eq. (20) for serial). Moreover, there has been a parallelism measure for graph G_i introduced (Eq. (4)) having its upper and lower bound dependent on number of vertices used in graph, respectively, $lp(G_{i\,\mathrm{sec}}) = \sum_{k=1}^{n_i} m_k$ and $lp(G_{ipar}) = 1$. Of course, one can use other measures to describe the possible graph parallelism level.

Having such a parallelism measure allows estimating differences in service completion times with respect to various values of the mentioned measure. This can be used to estimate the benefit (expressed as service completion time in this example) of using scenario graphs with various parallelism level values. The experimental example shows lower and upper bounds of service completion time (shortest and longest obtainable completion time at a given level of parallelism) for an exemplary graph with seven vertices (two of the vertices represent service input and output interfaces). In the example another parallelism measure (PM1) was used for comparison and is expressed as total number of paths connecting input and output interfaces divided by the vertices count. PM2 is a measure mentioned previously (Eq. (4)), expressed as the maximum number of vertices that can be used simultaneously in processing one request.

The experiment consisted of generating all possible directed acyclic graphs with seven vertices and with a distinguished starting and ending vertex. Each vertex of such a generated graph has been represented as a computational node with the proper atomic service version installed in it. Edges of the graph were modeled as communication channels linking computational nodes. To model the system and execute the experiment an OMNeT++ simulation environment has been used.

For each of the generated scenario graphs there was a single complex service request executed and completion time of such a composed complex service with graph parallelism measure values has been stored. From the collected data lower and upper bounds have been chosen (lowest and highest possible completion times for the given parallelism measure value) and are presented in Figure 8. Proper delays have been modeled as follows: single atomic service completion time was set to 1 s; transport delay for each edge was set to 0.1 s. It is noteworthy that the parallelism measures used in the experiment have different domains, and best and worst possible completion times in this example $d(G_{ipar}) = 1.2$ and $d(G_{iseq}) = 5.6$ are reached for various parallelism level values. This example shows that the monotonicity of the parallelism measures may vary. The completion time curve obtained for PM2 is nonincreasing but the curve obtained for PM1 is nonmonotonic.

The approach discussed in this article is useful in various optimization tasks concerning the shape of a service execution graph with respect to quality of service constraints. It can be implemented in a real SOA system for use

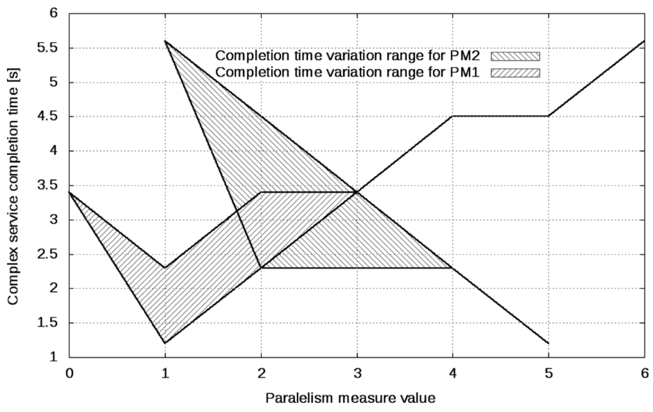

FIGURE 8 Lower and upper bounds (determined with completion time variation range) of complex service completion time with respect to constraints put on parallelism-level value using two different parallelism measures proposed.

in the composition process in order to optimize the quality of services delivered with the parallelism-level restrictions.

REFERENCES

Anderson, S., Grau, A., and Hughes, C. 2005. *Specification and Satisfaction of SLAs in Service Oriented Architectures*. Paper presented at the 5th Annual DIRC Research Conference, NESC – Edinburgh, Scotland..

Grzech, A. and Swiatek, P. 2009. Modeling and optimization of complex services in service-based systems. *Cybernetics and Systems* 40: 706–723.

Grzech, A., Rygielski, P., and Swiatek, P. 2010. *QoS-Aware Infrastructure Resources Allocation in Systems Based on Service-Oriented Architecture Paradigm*. 6th Working International Conference HET-NETs, Zakopane, Poland.

Jain, K. K. and Rajaraman, V. 1994. Parallelism measures of task graphs for multiprocessors. *Microprocessing & Microprogramming* 40(4): 249–259.

Johnson, R., Gamma, E., Helm, R., and Vlisides, J. 1995. *Design patterns; elements of reusable object-oriented software*. Addison-Wesley, New York, USA.

Keller, A. and Ludwig, H. 2003. The WSLA framework: Specifying and monitoring service level agreements for Web services. *Journal of Network and Systems Management* 11(1): 5781.

Li, X., Turner, S. J., Tony, K. H., Chan, H. M., and Hung, T. 2008. *Design of an SLA-Driven QoS Management Platform for Provisioning Multimedia Personalized Services*. Paper presented at the 22nd AINA Workshop, Okinawa, Japan.

Milanovic, N. and Malek, M. 2004. Current solutions for Web service composition. *IEEE Internet Computing* 8(6): 51–59.

Rau, B. R. and Fisher, J. A. 2000. Instruction-level parallel processing: History, overview, and perspective. In *Readings in Computer Architecture*, edited by M. D. Hill, N. P. Jouppi, and G. S. Sohi. San Francisco, CA: Morgan Kaufmann, pp. 288–308.

Stefanovic, D. and Martonosi, M. 2000. Limits and graph structure of available instruction-level parallelism (Research Note). In Proceedings from the 6th Euro-Par Conference, edited by A. Dode, T. Ludwig, W. Karl, and R. Wismller. Springer-Verlag, London UK, pp. 1018–1022.