

Web Service Configuration under Multiple Quality-of-Service Attributes

PengCheng Xiong, YuShun Fan, and MengChu Zhou, *Fellow, IEEE*

Abstract—With the popularity of Internet technology, web services are becoming the most promising paradigm for distributed computing. However, when a single web service fails to meet service requestor's multiple function needs, web services need to be dynamically configured together to form a web service composition. Since there may be many configurations providing identical functionality with different Quality-of-Service (QoS), a choice needs to be made according to users' functional and non-functional requirements. In this paper, we formulate a web service functional configuration problem by using Petri nets. The graph structure and algebraic properties of the model are analyzed in detail to show that a basis solution of a state-shift equation of the Petri net model corresponds to a realizable configuration process. This result is later used to formulate the multiple attribute QoS optimization problem to a linear programming problem. Finally, a case study is performed to show that the proposed analysis result can be effectively applied in practice.

Index Terms—Web service, Petri nets, optimization, modeling and analysis, linear programming

Note to Practitioners

Web service framework has evolved to become an important paradigm for distributed computing. When any single web service fails to accomplish service requestor's multiple function requirements, multiple web services need to be dynamically configured together to form a web service composition. This work deals with automatic configuration of services under practical constraints. First, according to the customized or application-specific web service functional requirement, discover all the web services. Second, by analyzing function decomposition and function selection on the service interface information, build a complete service functional dependency configuration net based on Petri nets. Third, choose and compute the quality-of-service (QoS) attributes for the whole configuration. A transformation method is utilized to change non-linear aggregation functions to linear ones. Relative importance or value trade-offs of different attributes are represented through subjective preference or

Manuscript received February 2007. This work was supported in part by the National High Technology Research and Development (863) Program of China under Grant 2006AA04Z151, the China National Science Foundation under Grant 60674080, the National Basic Research Program of China (973 Program) under Grant 2006CB705407, and Chang Jiang Scholars Program of PRC Ministry of Education.

P.C. Xiong and Y.S. Fan are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: xpc03@mails.tsinghua.edu.cn, fanyus@tsinghua.edu.cn)

M.C. Zhou is with Department of ECE, New Jersey Institute of Technology Newark, NJ 07102-1982 USA and is also with School of Electro-Mechanical Engineering, Xidian University, Xi'an, Shanxi 710071, China (e-mail: zhou@njit.edu).

perception. Fourth, the QoS attribute value for each real web service is gained. An association algorithm translates and compiles QoS attributes. Finally, the linear programming problem is set and solved. The best configuration is found and sensitivity analysis is carried out. The concepts and developed algorithms can be readily put into industrial applications.

I. INTRODUCTION

A web service [1-3] framework has evolved to become a promising technology for the integration of disparate software components using Internet protocols. Web Services Description Language (WSDL) [4], Universal Description, Discovery, and Integration (UDDI) [5] and Simple Object Access Protocol (SOAP) [6] are three core standards for web service description, registration and binding respectively. Web service providers register web services through an UDDI registry. The web service that they intend to offer is defined by WSDL. Then, web service users/requestors discover the needed web services and send the requests via invocation interfaces. After the response from a web service provider, they invoke those services under SOAP. A web service is defined as a web-accessible function that is well-defined, self-contained and does not depend on the context of state of other web services. When any single web service fails to accomplish service requestor's multiple function requirements, multiple web services need to be dynamically configured together to form a web service composition to satisfy both the functional and non-functional requirements, such as Quality of Service (QoS). Menasce [7] presents an overview of the current QoS research. The recent state of the art in the QoS study mainly covers the following areas.

The first is QoS specification and description, i.e., QoS modeling. QoS serves as a key index for discriminating candidate web services and web service compositions with identical functionality. Ran [8] organizes a set of quantifiable QoS parameters and measurements, e.g., reliability, capacity, availability and cost, into multiple QoS categories. tModel existing in UDDI registries is used to formalize representation models for each QoS parameter. Perryea and Chung [9] group web services into several communities according to non-functional requirement. Xiong and Fan [10] propose a uniform QoS attribute definition framework in order to coordinate QoS properties from both web service provider's and user's perspectives. In addition to the generic QoS dimensions discussed by the above papers, Yang *et al.* [11]

mention a number of context-dependent and domain-dependent nonfunctional properties. Liu *et al.* [12] extend the QoS model by including both generic and domain specific criteria. The QoS information can be collected from service providers, e.g., cost; or from service requesters' feedback, e.g., execution time; or from a third party, e.g., service reputation. We assume that QoS information in this paper is collected in a fair manner and is represented as part of the tModel.

The second is the composition of web services according to QoS requirements. Web service composition covers three distinct but overlapping viewpoints, i.e., behavioral interface, choreography and orchestration [13]. Behavioral interface denotes the behavioral aspects of the interactions for a given individual service during the composition. Choreography deals with collaborative processes involving multiple services to achieve a common goal. Orchestration describes both control flow and data dependency of web services in a business environment which are often accomplished by the Business Process Execution Language for Web Services (BPEL for short) [14]. The QoS of the composed service is determined by the QoS of its underlying component services. The aggregation functions for the computation of the QoS through component services are described in detail in [15]. It is not hard to imagine that there may be many feasible web service compositions that can meet the functional requirement but carry different QoS attributes. Therefore the QoS of the resulting composite service is a determinant factor to satisfy different requirements and preferences by different users. For example, a user may demand to maximize the reliability while another may demand to minimize the total cost. Different users may also give different importance to QoS attributes. Thus it is necessary to take into account the preferences of users and optimize the web service composition consistent with the QoS requirements. There are two kinds of web service quality optimization methods, i.e., local and global optimizations. Ardagna and Pernici [16] discriminate global and local QoS constraints.

The local optimization is done at a task level, i.e., choosing for each task the web service with the best QoS. Casati and Shan [17] propose eFlow to suit adaptive and dynamic features of web services required by different individual users and to cope with a highly dynamic business environment. To prevent service providers deviating from the advertised QoS and causing losses to the users, Jurca *et al.* [18] propose a novel QoS monitoring mechanism to collect the ratings and compute the actual quality delivered to the users. Huang *et al.* [19] present a moderated fuzzy web service discovery approach to model users' subjective and fuzzy opinions and to assist service users and providers in reaching a consensus.

Compared with the local optimization approach, the global optimization is done at a process level where services are selected for each task to obtain the optimal global quality. Lamparter *et al.* [20] use utility function policies to model the multiple preferences of users. But the utility function is hard to be quantified for multi-attribute decision. Gu and Nahrstedt [21] propose a model for composing different application services

into a service path satisfying the user's quality requirements. Gao *et al.* [22] combine different service paths as a weighted multistage graph and transform the selection of the optimal execution plan into the selection of the longest path. Since the service path is defined as a sequential chain of service operations, there is limitation when the search methods are used in complex service execution. To overcome the disadvantages, Zeng *et al.* [15] use statechart to describe the workflow execution sequence in detail, i.e., the control-flow, and then arrange a set of candidate web services that can equally accomplish the function for each activity in the statechart. Their work is further improved by Gao *et al.* [23] by adding more quality dimensions, i.e., capacity and load, to their QoS model. It yet suffers from several drawbacks. First, they use 0-1 integer programming [24] to obtain the solution. It is NP-hard, and belongs to one of Karp's 21 NP-complete problems actually [25]. This paper proposes the use of a linear programming method that is solvable in polynomial time (complexity class P). Second, they assume that every candidate web service is associated with only one workflow task. However, a web service probably contains many invocation interfaces and operations that can support many workflow tasks in fact. Moreover, these web services can be reused as many times as possible. Third, the web service execution sequence is relatively rigid according to the statechart that defines the control-flow. Canfora *et al.* [26] propose to use genetic algorithms instead of integer programming [15] to deal with a large set of candidate web services.

The third is about the dynamic environment where composite services are invoked. The dynamic properties are two folds. First, a service composition is operating under a dynamic heterogeneous environment and its run-time performance is fluctuating with the varying quality of the services. Second, when a web service becomes unusable or unreliable, the other web services in the composition that interact with it may need to be altered in order to adapt to new environment. A number of projects, e.g., METEOR-S [28] and CrossFlow [29], and middleware, e.g., AgFlow [15], SwinDeW [3] and GlueQoS [30] deal with QoS aware web service selection. The quality dimensions in [28-30] are limited and the variability of those quality attributes is not considered.

The fourth is the web service configuration. Service configuration and service composition complement each other in nature. The former is function-oriented and offer the lower level business functions required by the orchestration. It is often done under Service Component Architecture (SCA) [31]. The latter is mainly process-oriented and provides the higher level orchestration. Compared with process-oriented service composition, function-oriented service configuration enjoys more agility and flexibility. For example, if a web service is configured by two services, these two services may be executed in sequence or parallel depending on their implementation. Moreover, service configuration under SCA can protect business logic and improve testability, e.g., web service configuration information can be used by service fault

management to track defects [32].

SCA builds on service encapsulation through the assembly of heterogeneous services. After the components that provide services and consume other services are implemented, they are assembled to build the business application through the wiring of service references to services [33]. An SCA module is assembled by configuring and wiring together components, entry points and external services. Entry points are the representation of interfaces that are offered for use by components outside a module. If a component in a module references the services provided outside it, they are represented as external services. Moldt *et al.* [34] propose a modeling technique based on high-level Petri nets for the modeling of processes by service references. SCA assembly operates at two levels, i.e., the assembly of loosely connected components within a system and the assembly of closely connected components within a module. A dynamic configuration can be modeled as a functional assembly of the overall requested function (In this paper, assembly and disassembly operations are assumed to be mutually invertible). For example, in Fig. 1, System A can be wired by subsystems B, C, and D, or wired by subsystems X and Y. Subsystem C can be wired by module components E and F while E has two candidate modules G and H for real implementation. The SCA configuration also becomes a service, which can be accessed and reused in a uniform manner. Multiple service components can be configured and assembled into groupings called composites, to provide specific business capabilities that can be a part of other services. For example, subsystem C in Fig. 1 can be a composite and itself can be referenced by other services. Since the assembly of an SCA system mirrors the assembly of a module, we do not distinguish among subsystems, modules and services in the rest of the paper.

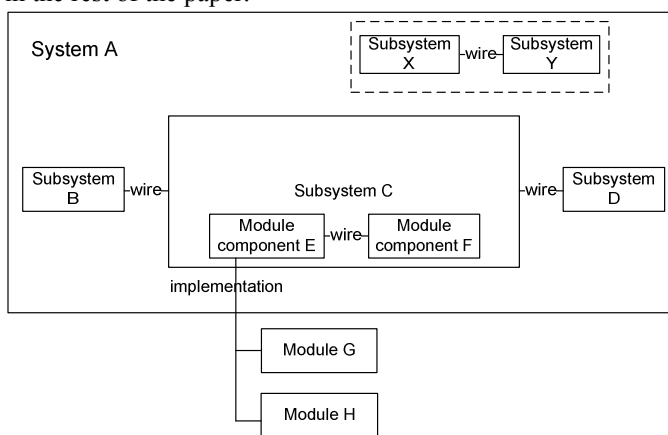


Fig. 1. Example of system disassembly

The main difficulty of deploying services under SCA is that the service disassembly methods are multiple and many configuration choices interweave to satisfy both the functional and non-functional requirements. We model the configuration problem as Service Functional Configuration (SFC). A web service may have multiple SFCs, e.g., in Fig. 1, System A can be configured by wiring subsystems B, C, and D, or subsystems X and Y. However, at one time, only one configuration can be

selected. The optimal choice should meet both the user's functional and QoS requirements.

As a dynamically formed digraph to present all of the references, Service Dependency Graph (SDG) is popularly used to depict functional dependency relationship among web services [35-36]. Liang and Su [35] propose an SDG based on an AND/OR graph to discover web services. Hasselmayer presents a dependency management architecture on web services and realizes the architecture with Jini connection technology [36]. Guo *et al.* [37] propose an optimized peer-to-peer overlay network for service discovery. However, although SDG provides a means for web service configuration descriptions in order to ensure functional interoperability among collaborating web services, it deals with only the functional aspect. It can hardly support dynamic configuration of web services to deal with non-functional requirement. Moreover, the SCA specification set being developed under the open service oriented architecture collaboration [38] and most of the current research on SCA does not incorporate QoS according to [39]. Thus, service requestors face a large number of choices of service configurations that can provide the similar function. In addition, an environment with changeable QoS of the component services makes the selection problem more challenging. Therefore, selecting an optimal configuration remains a difficult problem.

In this paper, we address the optimal configuration issues by concentrating on a) formal modeling and definition of the web service configuration and b) the optimal QoS search algorithm under varying configuration constraints. For the first issue, we introduce a formally defined configuration net named Service Configuration Net (SC-net) based on Petri nets [40-42] in order to help select candidate configurations. The main advantages of adopting Petri nets as a modeling language are two-fold. First, as a graphical and mathematical tool, Petri nets can provide a uniform environment for dealing with large and complex web service configurations. Second, through firing the transitions in the SC-net, we can generate a set of candidate configurations automatically and thus minimize the human efforts in searching. For the second issue, before we present our optimal QoS search algorithm, we carry out structural analysis to discover the essential properties of the SC-net we build. The analysis results clarify that the set of basis solutions of a state-shift equation of a configuration net is identical to set of the solutions that correspond to realizable configuration processes. Based on this conclusion, we formulate web service configuration under single and multiple QoS attributes as a linear programming problem. This significantly alleviates the solution complexity.

The rest of the paper is organized as follows: Section II models the configuration problem formally and proposes a service function configuration net based on Petri nets. Section III analyzes and compares the structural characteristics of the configuration net and the algebraic properties of solutions of the state-shift equation in detail. The web service quality attributes and aggregation function to compute the quality attributes for the configuration are provided in Section IV. Section V

specifies the problem as a linear programming problem while Section VI shows a real example to certify the validity of the analysis results and methodology. Section VII introduces the implementation of the methods. Finally, Section VIII concludes the paper.

II. MODELING WEB SERVICE CONFIGURATION WITH PETRI NETS

Assembly in SCA is the process of composing business applications by configuring components that provide service functions. We transform the assembly relationship in SCA into Petri nets, i.e., transform function combination and function selection to AND and OR structures in Petri nets in Fig. 2. We assume that a reader is familiar with basic Petri nets [40-42]. The following definition is used.

Definition 1: A Petri net is a 5-tuple, $PN = (P, T, I, O, M)$

where:

- i. $P = \{p_1, p_2, \dots, p_m\}$, $m > 0$, is a finite set of places pictured by circles;
- ii. $T = \{t_1, t_2, \dots, t_n\}$, $n > 0$, is a finite set of transitions pictured by bars, with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$;
- iii. $I: P \times T \rightarrow \{0, 1\}$ is an input function that defines the set of directed arcs from P to T ;
- iv. $O: P \times T \rightarrow \{0, 1\}$ is an output function that defines the set of directed arcs from T to P ;
- v. $M: P \rightarrow \mathbb{Z}$ is an $m \times 1$ column vector whose i th component represents the number of tokens in the i th place. An initial marking is denoted by M_0 , where $\mathbb{Z} = \{0, 1, 2, \dots\}$. Tokens are pictured by dots.

Postset of t is the set of output places of t , denoted by t^\bullet . Preset of t is the set of input places of t , denoted by *t . Post (Pre) set of p is the set of output (input) transitions of p , denoted by p^\bullet and *p respectively. In Fig. 2, $t_1^\bullet = \{p_{2-4}\}$, ${}^*t_1 = \{p_1\}$, $p_5^\bullet = \{t_{2-3}\}$ and $(p_5^\bullet)^\bullet = t_2^\bullet \cup t_3^\bullet = \{p_{6-7}\}$. A place is called a leaf place iff $p^\bullet = \emptyset$. In Fig. 2, p_{2-4} and p_{6-7} are leaf places. A marking in a Petri net is changed according to the following rules:

- i. A transition $t \in T$ is said to be enabled if and only if $M(p) \geq I(p, t)$, $\forall p \in P$, and denoted as $M[t > .$

- ii. Firing t at M leads to M' , denoted as $M[t > M'$, where $M'(p) = M(p) + O(p, t) - I(p, t)$.

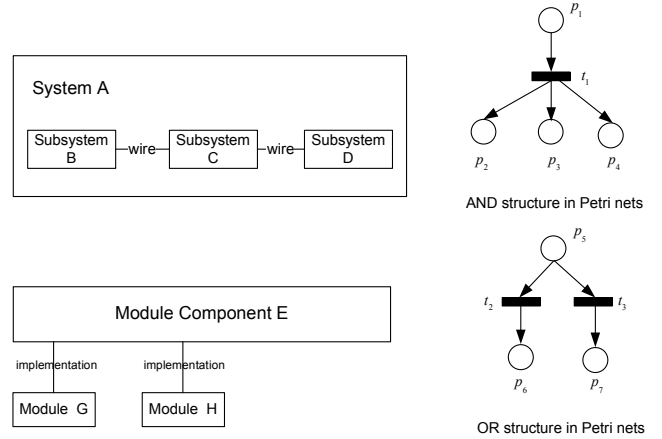


Fig. 2. Modeling SCA function assembly through Petri nets

Definition 2: Service set $S = WS \cup \{s_{dummy}\}$ where $WS = \{s_{1-2}\}$ is a finite set of real web services, and s_{dummy} is dummy service which denotes a composition or selection of real web services. We call a web service an atomic web service iff it does not reference other services.

Definition 3: Service Configuration Net (SC-net): an acyclic Petri net is an SC-net if:

- i. There is a place denoted by p' with no input arc, which corresponds to the service requested by a user. $M_0(p') = 1$ and $M_0(p) = 0$, $\forall p \neq p'$;
- ii. Every atomic web service in S is denoted by a leaf place;
- iii. Every other web service $s \in S$ excluding those mentioned in items 1 and 2 is mapped to a place set $P_s \subset P$. Every web service place $p \in P_s$ has the same function and non-functional attributes and can be treated as the duplication of s . The number of duplications is equal to the reusability frequency of s in the configuration;
- iv. Each non-leaf place has only one input arc and at least one output arc;
- v. Each transition has only one input arc and at least one output arc;
- vi. $\forall t \in T$, if $|t^\bullet| > 1$ and $\forall p_1, p_2 \in t^\bullet$, there is an AND relationship between p_1 and p_2 . $\forall t \in T$, if $|t^\bullet| = 1$,

$\bullet t = p$ and $|p^\bullet| > 1$, there is an OR relationship among $(p^\bullet)^\bullet$.

Note that the proposed SC-net falls into the class of disassembly Petri nets [43]. Based on the definition of the SC-net, we state SFC and a realizable configuration process.

Definition 4: An SFC at time ζ is denoted as $C(p', \zeta)$ such that (a) $p' \in C(p', \zeta)$, and (b) $\forall p \in C(p', \zeta)$, if $p^\bullet \neq \emptyset$, $\exists t \in p^\bullet$ and $\forall p'' \in t^\bullet$, $p'' \in C(p', \zeta)$.

Definition 5: A realizable configuration process is a sequence of transition firings from the initial marking M_0 to a current marking M . M is called a realizable configuration state.

If we denote system A, subsystems X, Y, B, C and D, module components E and F by p' and p_{3-9} respectively, we can obtain an SC-net in Fig. 3 for the configuration of system A in Fig. 1. There are three candidate configurations, i.e., $C_1(p', \zeta) = \{p', p_1, p_{3-4}\}$, $C_2(p', \zeta) = \{p', p_2, p_{5-8}\}$ and $C_3(p', \zeta) = \{p', p_2, p_{5-7}, p_9\}$. The firing transitions determine the configuration processes, e.g., $M_0[t_1 > M_1[t_3 > M_2$ corresponds to $C_1(p', \zeta)$ while $M_0[t_2 > M_3[t_4 > M_5[t_5 > M_6$ corresponds to $C_2(p', \zeta)$. Note that since the subgraph in the dotted trapezoid denotes the configuration information for subsystem C, if C can be referenced by other services, the subgraph can also be reused for configuration description according to the property 3 of the Definition 3.

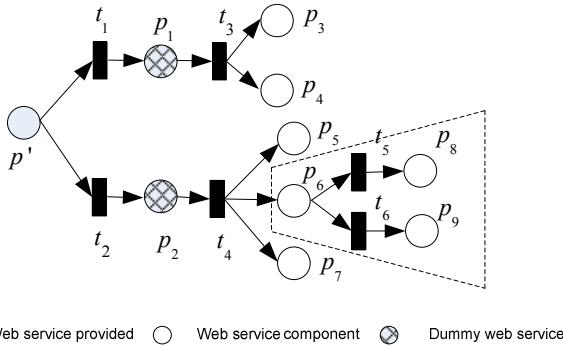


Fig. 3. SC-net representation of the configuration of system A

III. STRUCTURAL ANALYSIS OF SC-NET

In order to derive the candidate web service configurations automatically and obtain a more efficient QoS search algorithm, in this section, we analyze the graph and the algebraic structural properties of the SC-net.

A. Definition of incidence matrix and state-shift equation

Definition 6: For an SC-net with m places and n transitions, following [40], the incidence matrix $A = [a_{ij}] = O - I$ is an

$m \times n$ matrix of integers.

For example, for the SC-net in Fig. 3, the incidence matrix

$$A = \begin{bmatrix} -1, 1, 0, 0, 0, 0, 0, 0, 0, 0 \\ -1, 0, 1, 0, 0, 0, 0, 0, 0, 0 \\ 0, -1, 0, 1, 1, 0, 0, 0, 0, 0 \\ 0, 0, -1, 0, 0, 1, 1, 1, 0, 0 \\ 0, 0, 0, 0, 0, 0, 0, -1, 0, 1, 0 \\ 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 1 \end{bmatrix}^T.$$

Definition 7: The state-shift equation is $[E \ -A] \alpha = M_0$,

where A is the incidence matrix, $\alpha = \begin{bmatrix} M \\ x \end{bmatrix}$ is an $(m+n) \times 1$ column vector and E is an $m \times m$ unit matrix. x is an $n \times 1$ column vector of nonnegative integers. M and M_0 denote the current and initial marking, respectively.

x is called the firing count vector and the j th element of x denotes the number of times that t_j must fire to transform M_0 to M . The state-shift equation can be easily transformed to another type denoted by $Ax = M - M_0$ [40]. If we use $(\bullet p)^k$ and $(p^\bullet)^l$ to denote the k th fired input transition and the l th fired output transition for p , we can rewrite the state-shift equation as $M_0(p) = M(p) - \sum (\bullet p)^k + \sum (p^\bullet)^l$.

B. Definition of subgraphs and solutions

In this subsection, we define subgraphs in an SC-net and solutions for the state-shift equation and clarify the relationship between the structural properties of subgraphs and the algebraic characteristics of solutions.

Definition 8: In an SC-net, a subgraph that corresponds to a realizable Configuration Process (CP) consists of the fired transitions, their input places and the non-zero element of current marking M .

For example, for the SC-net in Fig. 3, a realizable CP can be $CP_1 = \{(P, T) \mid P = \{p'\}, T = \emptyset\}$, $CP_2 = \{(P, T) \mid P = \{p', p_1\}, T = \{t_1\}\}$ or $CP_3 = \{(P, T) \mid P = \{p', p_1, p_3, p_4\}, T = \{t_1, t_3\}\}$. In CP, p' has single output arc and the other non-leaf place has single input arc and single output arc.

Theorem 1: The set of places of a CP in which no transition can be enabled under M is a candidate SFC.

Proof:

1) Since p' does not have any input arc and $M_0(p') = 1$, we have $M(p') + \sum (p^\bullet)^l = 1$. Then $M(p') = 1$ or $\sum (p^\bullet)^l = 1$, which means the set of places of CP contains p' . This satisfies property (a) of SFC.

2) Without loss of generality, suppose that $M_0[t_1 > M_1[t_2 > M_2 \dots [t_n > M$. Since each transition has only one input arc, suppose that the input places of the fired

transitions are p', p_1, p_2, \dots , denoted as a place set P_F . Since no transition under M can be enabled, $\forall p_i \in P_F$ in CP, $\exists t_{i+1} \in p_i^\bullet$, $\forall p \in t_{i+1}^\bullet$, p is in P_F or $M(p) > 0$. This satisfies property (b) of SFC. \square

For example, since t_1 and t_2 are enabled in CP_1 , t_3 is enabled in CP_2 and no transition can be enabled in CP_3 , only the set of places of CP_3 is a candidate SFC.

Definition 9: In an SC-net, a subgraph that corresponds to a Solution of the State-shift Equation (SSE) consists of the transitions corresponding to non-zero elements of x , their input and output places, and places corresponding to non-zero elements of M .

For example, for the SC-net in Fig. 3, a solution of the state-shift equation can be

$$\alpha_1 = [0, 0.2, 0.8, 0, 0, 0, 0, 0, 0, 0, 0.2, 0.8, 0, 0, 0, 0]^T \quad \text{or}$$

$$\alpha_2 = [0, 0.1, 0, 0.9, 0.9, 0, 0, 0, 0, 0, 1, 0, 0.9, 0, 0, 0]^T, \text{ then SSE can be}$$

$$SSE_1 = \{(P, T) \mid P = \{p', p_1, p_2\}, T = \{t_1, t_2\}\} \quad \text{and}$$

$$SSE_2 = \{(P, T) \mid P = \{p', p_1, p_3, p_4\}, T = \{t_1, t_3\}\}$$

respectively.

Definition 10: If an SSE corresponds to a CP, and the final marking corresponds to a realizable configuration state, then the solution of the state-shift equation is a realizable solution.

Definition 11: If the column vectors of $[E \ -A]$ corresponding to non-zero elements of the solution are linearly independent of one another, the solution is a basis solution.

Definition 12: If all elements of a solution are non-negative integer, this solution is called non-negative integer solution.

Theorem 2[44]: Column vectors corresponding to the non-zero elements of the realizable solution in $[E \ -A]$ are linearly independent.

Theorem 3: A solution is a non-negative integer solution if and only if it is a realizable solution.

Proof: From Definition 8, it is obvious that if a solution is a realizable one, it is a non-negative integer one.

Consider the web service place p' . Following the equation $M_0(p') = M(p') + \sum (p^\bullet)^l = 1$, we have $M(p') = 1$ and $\sum (p^\bullet)^l = 0$ or $M(p') = 0$ and $\sum (p^\bullet)^l = 1$. When $M(p') = 1$ and $\sum (p^\bullet)^l = 0$, the other elements of the solution have to be 0. This means that the corresponding SSE consists only of place p' . In this case, the configuration process is in the initial state and this solution is obviously a realizable solution. When $M(p') = 0$ and $\sum (p^\bullet)^l = 1$, by the non-negative integer property of the solution, p' has a single output arc. This satisfies the structural property in Definition 8 that p' has a single output arc.

Consider a non-leaf place p . We have $M_0(p) = M(p) -$

$\sum (p^\bullet)^k + \sum (p^\bullet)^l = 0$. From property 4 in Definition 3, we have $M(p) + \sum (p^\bullet)^l = 1$. Since p is a non-leaf place, we have $M(p) = 0$, then $\sum (p^\bullet)^l = 1$ and p has only one output transition. This satisfies the structural property in Definition 8 that a non-leaf place has a single input arc and single output arc. \square

Theorem 4: A solution is a basis solution if and only if it is a non-negative integer solution.

Proof: From Theorems 2 and 3, it is obvious that if a solution is a non-negative integer solution, it is a basis solution. We prove the necessity by contradiction that we show that the solution whose element is not a non-negative integer is not a basis solution.

When the solution whose element is not non-negative integer, its SSE can be regarded as the composition of some CPs. There are two cases:

(1) Besides the leaf place of SSE, there does not exist a place with $M(p) > 0$.

(2) Besides the leaf place of SSE, there exists a place with $M(p) > 0$.

In case (1), from the leaf places to the output transition of the root place p' , we add the column vectors successively. We can generate two or more non-zero column vectors of which only the element corresponding to the web service place p' is not 0, and the other elements are all 0. This means that the column vectors are not linearly independent and the solution is not a basis solution.

In case (2), we can extract the subgraph whose root place satisfies $M(p) > 0$ and its successive places and transitions from SSE. From the leaf places to the output transition of the root place p , we add the column vectors successively. We can then generate two or more non-zero column vectors of which only the element corresponding to the web service place p is not 0, and the other elements are all 0. This means that the column vectors are not linearly independent, and the solution is not a basis solution. \square

For example, SSE_1 is in case (1). We can generate two non-zero column vectors of which only the element corresponding to the web service place p' is 1, and the other elements are all 0. SSE_2 is in case (2), we can extract the subgraph $\{(P, T) \mid P = \{p_1, p_{3-4}\}, T = \{t_3\}\}$, and we can then generate two column vectors of which only the element corresponding to the web service place p_1 is 1, and the other elements are all 0. In both cases, the solution is not a basis solution.

Theorem 5: A solution corresponds to a CP if and only if it is a basis solution.

Proof: From Theorems 3 and 4, it is obvious. \square

From Theorem 1, the set of places of a CP in which no transition can be enabled under M is a candidate SFC while

Theorem 5 represents that the set of the CPs is correspondent to the set of the basis solutions of $[E \ -A] \alpha = M_0$. Relationship among the set of solutions correspondent to SSEs, CPs and SFCs is shown in Fig. 4. We can conclude that the set of basis solutions of a state-shift equation of the SC-net is identical to the set of solutions which corresponds to a realizable configuration process. If we formulate non-functional objectives in a way that can restrict the search space of the set of solutions correspondent to CPs with optimal QoS to the set of solutions correspondent to the SFCs, we can then search the optimal QoS configuration by a linear programming technique.

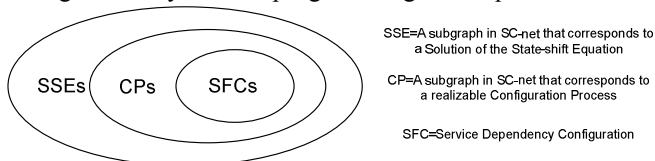


Fig. 4. Relationship among the set of solutions correspondent to SSEs, CPs and SFCs

IV. WEB SERVICE QUALITY MODELING

A. QoS attributes

Each functionality of a service can be evaluated by several QoS properties and parameters. The QoS parameters are sorted into runtime, transaction support, configuration management and cost, and security-related ones [8]. Each is made up of several metrics and sub-metrics. In this paper, we focus on configuration related QoS attributes. Generally, there are two kinds of QoS attributes, i.e., cost and benefit. For cost ones, i.e., the higher the value, the less optimal the solution. Other attributes are benefit, i.e., the higher the value, the better the solution. These include availability and reliability. Some of the QoS attributes can be measured quantitatively as shown in Tab.1.

TABLE I
QOS ATTRIBUTES AND MEASURES USED FOR EVALUATING CANDIDATE CONFIGURATIONS.

QoS attributes	Measures
Cost	The fee to be paid by a service requestor for invoking the web service each time.
Availability	The attribute for evaluating an immediate availability of a web service. It can be computed as the ratio of the service accessible time to the total time of observation.
Reliability	The probability of receiving the processing result within the expected duration time after the web service is

successfully invoked.

Successful execution rate

The probability that a request is correctly responded within the anticipated time indicated in the web service invocation context, e.g., the operation is successfully completed by the specific business activity execution. The successful execution rate is a measure related to component and/or hardware resource configuration of web services and the network connection status.

B. Aggregation

Suppose that we have q QoS attributes to evaluate, i.e., ψ_{1-q} .

The SFC QoS attributes are the cumulative effect of the QoS attributes of the non-dummy web service places that instantiate the modules of the SFC. The dummy web service places included in the SFC do not have QoS values. For calculation convenience, here we assume that a QoS attribute for a candidate SFC is a function of the QoS attribute of all the component services. We explain each attribute's aggregation function as follows:

1. Cost. The cost of an SFC is the sum of the costs of all the web services involved. Suppose that the reusability frequency for web service denoted by p is p_f , i.e., $p_f = 1$ if the web service is used once, $p_f > 1$ if the web service is reused.

Let $\psi_1(C(p', \zeta))$ and $\psi_1(p, \zeta)$ denote the cost of SFC and the web service p at time ζ , respectively. Then we have
$$\psi_1(C(p', \zeta)) = \sum_{p \in C(p', \zeta)} \psi_1(p, \zeta) * p_f.$$

2. Availability. The availability of an SFC is given by the product of the availability of all the web services involved.

Let $\psi_2(C(p', \zeta))$ and $\psi_2(p, \zeta)$ denote the results after applying the logarithm function to the availability of an SFC and the availability of the web service p at time ζ , respectively. Then the original product relations among all web services are converted into addition relations, i.e.,
$$\psi_2(C(p', \zeta)) = \sum_{p \in C(p', \zeta)} \psi_2(p, \zeta) * p_f.$$
 Then we transform the non-linear aggregation function to a linear aggregation function.

C. Computation

Suppose that we have N_c SFC candidates. The j th QoS attribute for the i th SFC is denoted as $\psi_{i,j}$, $1 \leq i \leq N_c$, $1 \leq j \leq q$. Different SFC candidate corresponds to different solution vector α_i , $1 \leq i \leq N_c$.

In order to compute $\psi_{i,j}$ by linear formulation of α_i , according to the definition of SFC and the linear aggregation

function, for the j th QoS attribute we associate a $1 \times (m+n)$ QoS attribute vectors V_j . Suppose that $V_j = [V_m \ V_n]$, where V_m and V_n are $1 \times m$ and $1 \times n$ vectors respectively. Elements in V_m are associated to the m places while elements in V_n are associated to the n transitions accordingly. The values of the elements in V_j are determined in the following algorithm:

Algorithm 1 (Association)

Step 1. If the j th QoS attribute is cost, we associate a positive large enough number L^+ to non-leaf web service places. If the j th QoS attribute is benefit, we associate a negative large enough number L^- .

Step 2. Every leaf web service place p is associated with QoS attribute $\psi_j(p, \zeta)$.

Step 3. For every non-leaf and non-dummy web service place p , we associate $\psi_j(p, \zeta)$ to all of its output transitions $t \in p^\bullet$.

Step 4. For every non-leaf and dummy web service place p , we associate a number 0 to all of its output transitions $t \in p^\bullet$.

Theorem 6: The j th QoS attribute for the i th candidate SFC, i.e., $\psi_{i,j}$, can be computed as $V_j^* \alpha_i$.

Proof: Since $\alpha = \begin{bmatrix} M \\ x \end{bmatrix}$, $V_j^* \alpha$ is equal to $V_m^* M + V_n^* x$.

For the web service s denoted by the non-leaf web service place, the reusability frequency of the web service in the configuration is equal to the firing number of the transitions $t \in p^\bullet$ where $p \in P_s$. Then the QoS attribute of non-leaf web service place can be aggregated through $V_n^* x$.

For the web service denoted by the leaf web service place p , the reusability frequency of the web service in the configuration is equal to the number of tokens in $M(p)$. Then the QoS attribute of leaf web service places can be aggregated through $V_m^* M$. Dummy web services chosen in a configuration do not affect the final result. Since the aggregation functions can be treated as linear ones, the conclusion holds. \square

V. PROBLEM SPECIFICATION

A linear programming problem is a kind of problem that has three inputs, i.e., a set of variables, an objective function, and a set of constraints [24]. Linear programming attempts to maximize or minimize the value of the objective function by adjusting the values of the variables under constraints. The results are the maximum (or minimum) value of the objective function and the values of variables at this maximum (or minimum). The objective function and the constraints are linear.

The $(m+n) \times 1$ column vector $\alpha = \begin{bmatrix} M \\ x \end{bmatrix}$ can be treated as the variable vector for the linear programming problem. The state-shift equation, i.e., $[E \ -A] \alpha = M_0$, can be taken as the constraints.

The web service user may have single or multiple QoS objectives. For example, the user may want the total cost minimized with the configuration availability maximized. From Theorem 6, for the j th QoS attribute, we have

$$\psi_j = V_j^* \alpha \quad (1)$$

A. Web Service Configuration under Single QoS Objective

If the j th QoS attribute is cost, the search problem of optimal configuration is formulated as (2). If the j th QoS attribute is benefit, it is formulated as (3).

$$\text{Minimize } \psi_j = V_j^* \alpha \quad (2)$$

$$\text{Or Maximize } \psi_j = V_j^* \alpha \quad (3)$$

$$\text{s.t. } [E \ -A] \alpha = M_0 \text{ and } \alpha \geq 0, \text{ where } \alpha = \begin{bmatrix} M \\ x \end{bmatrix} \text{ and}$$

$$M_0 = [1, 0, 0, 0, \dots, 0]^T$$

Lemma 1: The SSE with the optimal QoS is a CP.

Proof: From Theorem 5, the set of basis solutions of a state-shift equation of the SC-net is identical to the set of solutions which corresponds to a realizable configuration process. Since the optimal value of the objective function of the linear programming occurs with a basis solution, the SSE with the optimal QoS is a CP. \square

Theorem 7: The association algorithm can limit the SSE with the optimal QoS to be achieved only when an SSE corresponds to an SFC.

Proof: From Lemma 1, the SSE with the optimal QoS is a CP. Then we finish our proof by contradiction. We suppose that there exists a CP with the optimal QoS that does not correspond to an SFC.

By Theorem 1, there exists a transition that can be enabled under M . According to the structural characteristics of the SC-net, no transition in CP under M can be enabled if every non-leaf place $p \in P$ satisfies $M(p) = 0$. Then there exists at least one non-leaf place $p \in P$ satisfying $M(p) > 0$. According to Step 1 in the association algorithm, there are two cases.

1) If the j th QoS attribute is cost, after aggregation through $V_m^* M$, $V_j^* \alpha$ will be too large.

2) If the j th QoS attribute is benefit, after aggregation through $V_m^* M$, $V_j^* \alpha$ will be too small.

Considering the objective function stated in (2) and (3), $V_j^* \alpha$ can achieve the optimal value neither in case a) nor b).

Therefore, the conclusion holds. \square

Note that, the approach we apply in Step 1 in the association

$$V_2 = [L^-, L^-, L^-, L^-, L^-, L^-, -0.11, L^-, L^-, L^-, L^-, L^-, L^-, -0.36, -0.04, -0.11, -0.08, L^-, -0.05, -0.02, -0.06, -0.19, -0.13, -0.01, -0.17, -0.08, -0.11, -0.03, -0.05, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

Then the problem can be formulated as:

$$\text{Maximize Value} = \sum_{j=1}^2 \Gamma_j * w_j$$

$$\text{s.t. } [E \quad -A] \alpha = M_0$$

$$\alpha = \begin{bmatrix} M \\ x \end{bmatrix} \text{ and } \alpha \geq 0, M_0 = [1, 0, 0, 0, 0, \dots, 0]^T \text{ and } A \text{ is the}$$

incidence matrix of the net.

TABLE II

THE WEB SERVICES p' AND p_{1-22} DENOTE AND THEIR QOS ATTRIBUTES.

Places	Web service name	Cost	Availability(after taking logarithm)
p'	Sales Management Service	10	0.99 (-0.01)
p_1	Logistics Service	50	0.84 (-0.17)
p_2	Order Management Service	20	0.92 (-0.08)
p_3	Customer Payment Service	50	0.89 (-0.11)
p_4	Transportation Management Service	35	0.97 (-0.03)
p_5	Warehouse Management Service	60	0.95 (-0.05)
p_6	Invoice Management Service	40	0.89 (-0.11)
p_{13}	Railage Registry Service	10	0.70 (-0.36)
p_{14}	Airfreight Registry Service	100	0.96 (-0.04)
p_{15}	Product Inquiry Service (Application area Indexed)	25	0.89 (-0.11)
p_{16}	Product Inquiry Service (Provider Indexed)	20	0.92 (-0.08)
p_{18}	Credit Validation Service	60	0.95 (-0.05)
p_{19}	Remittance Service	30	0.98 (-0.02)
p_{20}	Bank 3 Account Service	28	0.94 (-0.06)
p_{21}	Bank 2 Account Service	22	0.83 (-0.19)
p_{22}	Bank 1 Account Service	25	0.88 (-0.13)
$p_{7-12,17}$	Dummy Service	L^+	L^-

Suppose that $L^+ = 1000$, $L^- = -8$ and the weight for cost and availability are 0.2 and 0.8, respectively. The problem can be solved by a simplex algorithm or interior-point algorithm. The result is

$$\alpha = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0]^T$$

The corresponding SFC contains $p', p_{1-6}, p_{14}, p_{16}$ (reuse once), p_{18} and p_{19} .

In order to make our configuration adapted to the dynamic environment, we can perform sensitivity analysis [47] after the optimal configuration is found. Generally, in the standard linear programming problem, if there is any change in the values of coefficient matrix (i.e., $[E \quad -A]$), right hand side vector (i.e., M_0) or objective function coefficients (i.e., V_i and w_i), the optimal solution is likely to change. However, when the perturbations are within a certain range, the current optimal solution may remain unchanged. This invariance of the optimal solution is a desirable property which helps reduce significantly the computational complexity when perturbations occur and/or we are uncertain about the exact values of coefficients. In the SFC problem, the coefficient matrix and right hand side vector often stay unchanged, while the objective function coefficients V_i often fluctuate according to the variable environment (suppose that web service user's preference on w_i stays unaffected). Since we treat the SFC problem as two-attribute MADM in the above case, when more than one attribute and/or more than one web service change, the problem becomes more complicated. We take an example that the attributes of Remittance Service (p_{19}) can change while the attributes of other web services in the optimal configuration remain unchanged. Through sensitivity computation, we deduce that the maximum range of cost perturbation is 18.14 and the maximum range of reliability perturbation is 3.45% for Remittance Service when the optimal configuration stays unchanged.

VII. IMPLEMENTATION FRAMEWORK

We present our implementation framework in Fig. 7. There are mainly four roles in the framework, web service user, service configuration manager, UDDI registry and web services. The service configuration manager contains two major components, i.e., service configuration planner and service configuration optimizer.

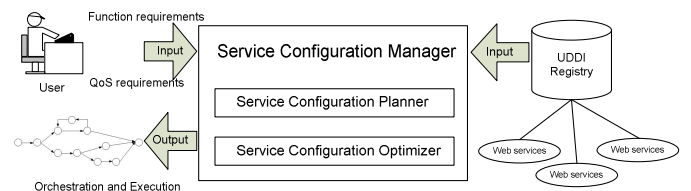


Fig. 7. Implementation framework for SC-net

We describe the interaction between roles in our framework according to the life cycle of a web service [48-49].

Registration stage. Web service providers publish their web service and invocation interfaces they intend to offer in WSDL

and register the web services to a common registration table located at UDDI. Schema API in UDDI [5] defines four data types mainly in functional aspects, i.e., *businessEntity*, *businessService*, *bindingTemplate* and *tModel*. We extend the UDDI data model with web service quality information and encapsulate QoS information following a uniform template formalized by XML Schema Definition in our previous work [10]. Since QoS metadata in XML format can be sent to UDDI through SOAP, the new UDDI registry can provide service configuration manager with both the function and QoS attributes of the web services.

Configuration stage. The web service user inputs the function and QoS requirement to the service configuration manager. In the case study of sales management service configuration, the salesman may require the function of recording details of the order, checking the credit of the customers, delivering the goods and so on. The relative importance of different QoS attributes represented for subjective preference or perception is also provided, e.g., the weight for cost and availability are 0.2 and 0.8, respectively. First, according to the functional requirement, the service configuration planner automatically selects services based on whether they meet the configuration's structural constraints and match interfaces. An SC-net corresponding to the configuration is built. The service configuration planner may check the potential problems due to missing services and unsatisfied interfaces. Second, the service configuration optimizer obtains the QoS requirement from the user and accesses to QoS metadata about services from the extended UDDI data model. The QoS aggregation function is computed. Through the association algorithm mentioned above, a linear programming problem is specified. Finally, the service configuration optimizer finds the optimal configuration. Sensitivity analysis can also be performed afterwards.

We adopt the Spring Framework [50] for the implementation of a service configuration planner. The components in a module are created and wired using dependency injection capabilities from Spring Framework. Access methods and protocols, e.g., SOAP, can be adopted to specify the binding information that describes how services can be accessed and referenced. The service configuration optimizer is implemented by a linear programming solver based on the simplex algorithm.

Orchestration stage. The optimal configuration is deployed in the execution environment. The web service orchestration and deployment are done with the CIMFlow-System [51], which is a formerly developed central workflow management system. In CIMFlow-System, we use a process modeler module to establish a workflow model and task allocator module to receive the task allocation request and assign tasks to specific web service interfaces.

Execution stage. After deployment, the business process can be executed and supported by the workflow engine in CIMFlow-System. When a transaction instance is generated, the service configuration manager determines which services should be used in the configuration by interacting with the CIMFlow-System. The optimal configuration information is

sent back to CIMFlow-System for dynamic binding. Runtime performance analysis and optimization are also conducted with the CIMFlow-System, e.g., the turnaround time calculation and optimal execution path selection [52].

VIII. CONCLUSIONS

Web service framework brings in a new revolution in traditional computing. By assembling the service components, service component architecture provides a programming model for the creation and assembly of business systems using a service-oriented architecture. However, many web service-related problems still remain open, including web service modeling, service discovery, service selection, service configuration, service deployment and execution. This work deals with functional and non-functional constraint modeling and configuration of services under constraints. This work can be treated as a complement to the SCA policy framework to support the specification of functional requirement and QoS expectations.

The paper presents a service functional configuration net based on Petri nets for the web service presentation and automatic assembly. The configuration specifications for the module and component services are described through the structure of disassembly Petri nets. The candidate configurations are generated automatically through firing the transitions in such Petri nets.

Next, by carefully analyzing on the structure of the configuration net and the algebraic property of the state-shift equation of Petri nets, we discover that the set of all candidate configuration processes is identical to the set of all basis solutions of the state-shift equation, which leads to the useful conclusion that the functional constraint of a configuration can be replaced by the state-shift equation.

Then, after compiling the QoS attributes and aggregation function to compute the QoS for the whole configuration, we formalize the optimal web service configuration problem as a linear programming problem. Hence, more efficient algorithms can be applied.

Finally, we implement the proposed functions and algorithms as the service configuration manager and incorporate the web service registration, orchestration and invocation functions into the framework architecture. The implementation framework is a platform from component design through to concrete deployment.

The proposed approach in this paper lacks such features as concurrency and synchronization. Actually, a service configuration net models a set of components that are used to make a particular business function. It models the way in which they are configured. It does not model the time sequences involved in executing particular service operations. The techniques proposed in this paper need to be extended to deal with business process sequencing and runtime related QoS. This will be left for future exploration.

REFERENCES

- [1] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI", *IEEE Internet Computing*, Vol. 6, No. 2, Mar./Apr. 2002, pp. 86-93.
- [2] F. Kraemer and P. Herrmann, "Service Specification by Composition of Collaborations--An Example", *Proc. of 2006 IEEE/WIC/ACM International Conference on Web Intelligence and International Agent Technology Workshops*, Hong Kong, Dec. 2006, pp. 129-133.
- [3] J. Yan, Y. Yang and G.K. Raikundalia, "SwinDeW-a p2p-based decentralized workflow management system", *IEEE Trans. on Systems, Man and Cybernetics, Part A*, Vol. 36, Iss. 5, Sept. 2006, pp. 922-935.
- [4] R. Chinnici, M. Gudgin, J.J. Moreau and S. Weerawarana, "Web Services Description Language (WSDL) Ver. 1.2", World Wide Web Consortium, 2002, www.w3.org/TR/wsdl12/.
- [5] "Universal Description, Discovery and Integration. Organization for the Advancement of Structured Information Systems", 2002, www.uddi.org/specification.html.
- [6] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Nielsen, S. Thatte and D. Winer, "Simple Object Access Protocol (SOAP) 1.1", <http://www.w3.org/TR/soap/>.
- [7] D.A. Menasce, "QoS issues in Web services", *IEEE Internet Computing*, Vol. 6, Iss. 6, Nov./Dec. 2006, pp. 72-75.
- [8] S. Ran, "A model for web services discovery with QoS", *SIGecom Exchanges*, Vol. 4, Iss. 1, Mar. 2003, pp. 1-10.
- [9] C.A. Perryea and S. Chung, "Community-Based Service Discovery", *Proc. of 2006 IEEE International Conference on Web Services*, Chicago, USA, Sept. 2006, pp. 903-906.
- [10] P.C. Xiong and Y.S. Fan, "A QoS-aware UDDI model for web services discovery", *Proc. of 2006 Asia Pacific Symposium on SSME*, Beijing, China, Nov. 2006, pp. 121-123.
- [11] W.J. Yang, J.Z. Li and K.H. Wang, "Domain-Adaptive Service Evaluation Model", *Chinese Journal of Computers*, Vol. 28, No. 4, Apr. 2005, pp. 514-523.
- [12] Y. Liu, A.H.H. Ngu, and L. Zeng, "QoS Computation and Policing in Dynamic Web Service Selection", *Proc. of the 13th International World Wide Web Conference*, New York, USA, May 2004, pp. 66-73.
- [13] A. Barros, M. Dumas and P. Oaks, "Standards for Web Service Choreography and Orchestration: Status and Perspectives", *Proc. of 2006 Business Process Management Workshops*, Nancy, France. Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 2006, Vol. 3812, pp. 61-74.
- [14] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, "Business Process Execution Language for Web Services", version 1.1, May 2003, <http://www-106.ibm.com/developerworks/webservices/library/ws-bpel>
- [15] L.Z. Zeng, B. Benattallah, A.H.H. Ngu, M. Dumas, J. Kalaganam and H. Chang, "QoS-aware middleware for Web services composition", *IEEE Trans. on Software Engineering*, Vol. 30, Iss. 5, May 2004, pp. 311-327.
- [16] D. Ardagna and B. Pernici, "Global and local QoS constraints guarantee in Web service selection", *Proc. of 2005 IEEE International Conference on Web Services*, Orlando, USA, July 2005, pp. 805-806.
- [17] F. Casati and M.C. Shan, "Dynamic and Adaptive Composition of E-Services", *Information Systems*, Vol. 26, Iss. 3, May 2001, pp. 143-163.
- [18] R. Jurca, B. Faltings and W. Binder, "Reliable QoS Monitoring Based on Client Feedback", *Proc. of the 16th International World Wide Web Conference*, Banff, Canada, May 2007, pp. 1003-1011.
- [19] C.L. Huang, C.C. Lo, K.M. Chao and M. Younas, "Reaching consensus: A moderated fuzzy web services discovery method", *Information and Software Technology*, Vol. 48, Iss. 6, June 2006, pp. 410-423.
- [20] S. Lamparter, A. Ankolekar, S. Grimm, "Preference-based Selection of Highly Configurable Web Services", *Proc. of the 16th International World Wide Web Conference*, Banff, Canada, May 2007, pp. 1013-1022.
- [21] X. Gu and K. Nahrstedt, "A Scalable QoS-Aware Service Aggregation Model for Peer-to-Peer Computing Grids", *Proc. of the 11th IEEE International Symposium on High Performance Distributed Computing*, Edinburgh, UK, July 2002, pp. 73-82.
- [22] Y. Gao, B. Zhang, J. Na, L. Yang, Y. Dai and Q. Gong, "Optimal Selection of Web Services for Composition Based on Interface-Matching and Weighted Multistage Graph", *Proc. of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Dalian, China, Dec. 2005, pp. 336-338.
- [23] A.Q. Gao, D.Q. Yang, S.W. Tang and M. Zhang, "Web Service Composition Using Integer Programming-based Models", *Proc. of the IEEE International Conference on e-Business Engineering*, Beijing, China, Oct. 2005, pp. 603-606.
- [24] R.S. Garfinkel, G.L. Nemhouse, *Integer Programming*, John Wiley and Sons, New York, 1972.
- [25] R.M. Karp, "Reducibility Among Combinatorial Problems", in R.E. Miller and J.W. Thatcher (Eds.): *Complexity of Computer Computations*. Plenum Press, New York, 1972, pp. 85-103.
- [26] G. Canfora, M.D. Penta, R. Esposito and M.L. Villani, "An approach for QoS-aware service composition based on genetic algorithms", *Proc. of the 2005 conference on Genetic and evolutionary computation*, Washington, D.C., USA, June 2005, pp. 1069-1075.
- [27] G. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.
- [28] R. Aggarwal, K. Verma, J. Miller and W. Milnor, "Constraint Driven Web Service Composition in METEOR-S", *Proc. of the 2004 IEEE International Conference on Services Computing*, Shanghai, China, Sept. 2004, pp. 23-30.
- [29] P. Grefen, and Y. Hoffner, "CrossFlow-cross-organizational workflow support for virtual organizations", *Proc. of the Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises*, Sydney, Australia, Mar. 1999, pp. 90-91.
- [30] E. Wohlstader, S. Tai, T. Mikalsen, I. Rouvellou and P.Devanbu, "GlueQoS: middleware to sweeten quality-of-service policy interactions", *Proc. of the 26th International Conference on Software Engineering*, Edinburgh, United Kingdom, May 2004, pp. 189-199.
- [31] IBM, SAP, Oracle, BEA, Sybase, Siebel Systems, etc.: SCA Assembly Model Specification, <http://www-128.ibm.com/developerworks/library/specification/ws-sca/>
- [32] Y. Li, K.W. Sun, J. Qiu and Y. Chen, "Self-reconfiguration of service-based systems: a case study for service level agreements and resource optimization", *Proc. of the 2005 IEEE International Conference on Web Services*, Orlando, USA, Vol. 1, July 2005, pp. 266-273.
- [33] J.L. Fiadeiro, A. Lopes, L. Bocchi, "A Formal Approach to Service Component Architecture", *Proc. of the 3rd International Workshop on Web Services and Formal Methods*, Vienna, Austria, Sept. 2006, pp. 193-213.
- [34] D. Moldt, S. Offermann and J. Ortmann, "A Proposal for Petri Net Based Web Service Application Modeling", *Proc. of the 4th International Conference on Web Engineering*, Munich, Germany, July 2004. Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg, 2004, Vol. 3140, pp. 93-97.
- [35] Q.A. Liang and S.Y.W. Su, "AND/OR Graph and Search Algorithm for Discovering Composite Web Services", *International Journal of Web Services Research*, Vol. 2, No. 4, Oct.-Dec. 2005, pp. 48-67.
- [36] P. Hasselmayer, "Managing Dynamic Service Dependencies", *Proc. of the 12th International Workshop on Distributed Systems: Operations and Management*, Nancy, France, Oct. 2001, pp. 141-150.
- [37] H.Q. Guo, D.Q. Zhang, L.H. Ngoh, W.C. Wong, S. Zheng and Y.K. Koh, "An Optimized Peer-to-Peer Overlay Network for Service Discovery", *Proc. of the 11th IEEE Symposium on Computers and Communications*, Sardinia, Italy, June 2006, pp. 82-87.
- [38] Open Service Oriented Architecture Collaboration. <http://www.osoa.org/display/Main/Home>
- [39] Z.L. Zou and Z.H. Duan, "Building Business Processes or Assembling Service Components: Reuse Services with BPEL4WS and SCA", *Proc. of the 4th IEEE European Conference on Web Services*, Zurich, Switzerland, Dec. 2006, pp. 138-147.
- [40] M.C. Zhou and K. Venkatesh, *Modeling, Simulation and Control of Flexible Manufacturing Systems: A Petri Net Approach*, World Scientific, Singapore, 1998.
- [41] B. Hruz and M.C. Zhou, *Modeling and Control of Discrete Event Dynamic Systems*, Springer, London, UK, 2007.
- [42] P.C. Xiong, Y.S. Fan and M.C. Zhou, "QoS-aware Web Service Configuration," accepted by *IEEE Trans. on Systems, Man, and Cybernetics: Part A*, Aug. 2007.

- [43] Y. Tang, M.C. Zhou, and R.J. Caudill, "An Integrated Approach to Disassembly Planning and Demanufacturing Operation," *IEEE Trans. on Robotics and Automation*, Vol. 17, Iss. 6, Dec. 2001, pp. 773-784.
- [44] T. Suzuki, T. Kanehara, A. Inaba and S. Okuma, "On algebraic and graph structural properties of assembly Petri net", *Proc. of the 1993 IEEE International Conference on Robotics and Automation*, Yokohama, Japan, Vol. 2, May 1993, pp. 507-514.
- [45] C.L. Hwang and K.L. Yoon, *Multiple Attribute Decision Making: Methods and Applications*, Springer, New York, 1981.
- [46] J. Ma, Z.P. Fan and L.H. Huang, "A subjective and objective integrated approach to determine attribute weights", *European Journal of Operational Research*, Vol. 112, Iss. 2, Jan. 1999, pp. 397-404.
- [47] A. Deif, *Sensitivity Analysis in Linear Systems*, Springer, New York, 1986.
- [48] C. Pautasso and G. Alonso, "Flexible Binding for Reusable Composition of Web Services", *Proc. of the Workshop on Software Composition*, Edinburgh, Scotland, Apr. 2005, pp. 151-166.
- [49] E. Karakoc, K. Kardas, and P. Senkul, "A Workflow-Based Web Service Composition System", *Proc. of 2006 IEEE/WIC/ACM International Conference on Web Intelligence and International Agent Technology Workshops*, Hong Kong, Dec. 2006, pp. 113-116.
- [50] Spring Framework. <http://www.springframework.org/>
- [51] Y.S. Fan, *Fundamentals of Workflow Management Technology*, Springer, New York, 2001.
- [52] J.Q. Li, Y.S. Fan and M.C. Zhou, "Performance modeling and analysis of workflow", *IEEE Trans. on Systems, Man and Cybernetics*, Part A, Vol. 34, Iss. 2, March 2004, pp. 229-242.